

Advantage™ VISION:Transact® for CICS®

Reference Guide

Release 7.8



Computer Associates®

B02567-1E

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2004 Computer Associates International, Inc.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Introduction

VISION:Transact Generator and the Target Communications Monitor	1-3
The VISION:Transact Executor and Its Relationship to IMS TM	1-6
VISION:Transact Executor and Its Relationship to CICS	1-8
VISION:Transact Run Types	1-9
Definition Run – (Catalog Maintenance)	1-9
Application Generation Run – (APPGEN)	1-10
Source Statement Retrieval Run	1-13
Library Utilities	1-14
Statement Types	1-14
Run Control Group	1-14
File Definition Statements	1-15
Screen/COMMAREA* Definition Statements	1-16
Screen/COMMAREA* Control Statements	1-17
Requests/Procedures	1-17
Comments	1-17
VISION:Transact Library	1-18

Chapter 2: A VISION:Transact Generated Application System

Using the Application System	2-4

Chapter 3: Essentials of Data Definition

Chapter 4: Application Flow

A Simple Model of the Application Flow	4-2
Introduction to the Interaction	4-4
Introduction to the Screen Processing/COMMAREA Definition	4-6
Dynamic Selection of Action Codes	4-9
Managing the Application Data Flow	4-14
Selecting the Output Screen/COMMAREA Section of an Interaction	4-22
Application Linking Via Screen Control	4-26
Interactions and Screen/COMMAREA Processing Definitions	4-29
Application Flow Summary	4-30

Chapter 5: Screen/COMMAREA Layout and Screen/COMMAREA Fields

Defining Screen/COMMAREA Layouts	5-1
The SD Statement	5-1
Placing Literals and Data Fields in the Screen	5-3
Relating Screen/COMMAREA Field Designators and S0 Statements	5-4
Output Screen/COMMAREA Facilities	5-5
Screen Field Attributes	5-6
Extended Attributes – Highlighting, Color, and Programmed Symbols	5-7
Editing Screen/COMMAREA Fields	5-8
Field Descriptors for Automatically Formatted Screens	5-10
Input Screen/COMMAREA Facilities	5-11
Screen/COMMAREA Field Names	5-11
Assigning a Length to a Virtual Screen Field	5-12
Defaults for Screen/COMMAREA Field Data	5-12
Input Required in a Screen/COMMAREA Field	5-13
De-Editing Screen/COMMAREA Fields	5-13
Validating Input Screen/COMMAREA Field Data	5-14
Error Processing for the Input Screen/COMMAREA	5-19
Displaying Modified Screen Field Data	5-20
Highlighting the Field in Error	5-20
Using the MS Operator to Supply a Message for Validation Error	5-20
Using the EC and ET Operators to Override VISION:Transact Standard Error Messages	5-20
Errors in Virtual Screen Fields	5-22
Screen/COMMAREA Field Error Processing in an Input Screen Request/Procedure	5-22
Introduction to Structured Screens/COMMAREA	5-23
Screen/COMMAREA Layout and Screen/COMMAREA Fields Summary	5-23
Transaction Control Fields in a COMMAREA	5-26

Chapter 6: Structured Screens/COMMAREA And The Intermediate Structure

Structured Screens/COMMAREA	6-1
Data Accessibility to the Interaction	6-7
A Screen Synchronized Data Base View	6-8
Processing All Occurrences of a Screen/COMMAREA Synchronized Data Base	6-13
The Intermediate Structure	6-13
Segment Accessibility Rules	6-18
Looping on the Intermediate Structure	6-18
The Intermediate Structure and Output Screen/COMMAREA Processing	6-19
The Source of Screen/COMMAREA Synchronization	6-20
The Segment Operator on Conditional Screen/COMMAREA Control Statements	6-21
Intermediate Structure Summary	6-22

Chapter 7: Requests/Procedures and the Intermediate Structure

Overview	7-1
Screen/COMMAREA Requests/Procedures	7-2
Type N (or Blank) Requests/Procedures	7-2
Type 1, 2, M, and O Screen/COMMAREA Requests/Procedures	7-3
Subroutine Requests/Procedures	7-3
Request Operations and Procedure Commands	7-4
Summary of Request Operators	7-4
Summary of ASL Procedure Commands	7-5
Qualifier	7-6
Statement Label	7-6
Relational or Logical Expressions	7-7
Branching or Transfer Locations	7-8
Arithmetic Operations and Expressions	7-9
External Calls	7-9
Set Error	7-10
Message Sending	7-10
Partial Fielding	7-10
Processing Recurring (Lower Level/Subordinate) Segments	7-12
Components of a Request or Procedure	7-12
Request Components	7-12
Procedure Components	7-15

Request/Procedure Types and Data Availability	7-18
Type 1 Requests/Procedures	7-19
Type 2 Requests/Procedures	7-22
Type M Requests/Procedures	7-24
The Type O Request/Procedure	7-25
The Subroutine (Type S) Request/Procedure	7-28
Request Back Branching	7-30
Screen/COMMAREA Field Error Processing in Screen/COMMAREA Requests or Procedures	7-33
Set Error and Message Sending	7-33
Segment Actions and Data Base Availability to Screen/COMMAREA Requests and Procedures	7-35
Extending the Intermediate Structure	7-40
Virtual Screen/COMMAREA Fields	7-41
Looping	7-43
Nested Looping	7-45
Screen/COMMAREA Request/Procedure Summary	7-53

Chapter 8: Dynamic Synchronization

Direct Synchronization	8-2
Browsing Dynamically Synchronized Data Bases	8-2
Range Browsing	8-3
Generic Browsing	8-3
Browse Considerations	8-4
Dynamic and Screen/COMMAREA Synchronization	8-4
Accessing the Dynamically Synchronized Data Base	8-5
The Use of Dynamically Synchronized Data Bases	8-7
Dynamic Synchronization Summary	8-14

Chapter 9: Data Manipulation/Field Handling

Fields.....	9-1
Character String Fields.....	9-1
Date Fields.....	9-2
Zoned Decimal Fields.....	9-2
Packed Decimal Fields (Signed).....	9-2
Fixed Point Fields.....	9-2
Floating Point Fields.....	9-2
Field Handling Rules.....	9-3
Implicit Definition of Temporary Fields.....	9-6
Definition of Screen\COMMAREA Fields and Virtual Fields.....	9-7
Screen/COMMAREA Fields.....	9-7
Virtual Screen/COMMAREA Fields.....	9-8
Processing Invalid, Missing, and Overflow Fields.....	9-9
Invalid, Missing, Overflow, or Error Indicators in COMMAREA Fields.....	9-10

Chapter 10: Defining Data

VISION:Transact and IMS.....	10-3
Writing the File Definition.....	10-4
VISION:Transact Segment Actions and SSAs.....	10-7
IMS Secondary Indexing.....	10-8
IMS Fast Path Data Bases.....	10-10
VISION:Transact and DB2.....	10-12
VISION:Transact and VSAM/ISAM Files (CICS TS Only).....	10-13
Considerations for the File Definition.....	10-13
VSAM KSDS Files.....	10-15
VSAM ESDS Files.....	10-16
VSAM RRDS Files.....	10-17
VSAM Alternate Indexing.....	10-17
Defining Pseudo Files.....	10-18
Defining Working Storage, PCB Masks, and Temporary Fields.....	10-19

Chapter 11: Advanced Application Flow

VISION:Transact Screen/COMMAREA Control	11-1
Output Select Operator	11-1
Output Control Operator	11-2
Execute Request or Procedure Operator	11-4
Screen Request/Procedure Sharing	11-5
Extended Application Generation	11-5
Statement Types and Input Sequence	11-5
VISION:Transact Library Facilities	11-6

Chapter 12: Execution Tracing

Chapter 13: Introduction and Overview (DB2 UDB)

Chapter 14: A Conceptual Overview (DB2 UDB)

Introduction	14-1
Relational Data Bases as Access Methods	14-2
Logical Records Versus Physical Records	14-2
Terminology	14-3
Defining the Logical Record	14-4
Screen Synchronization and Dynamic Synchronization	14-9
Updating and Output	14-9
The Advantages of Logical Records	14-9
Summary	14-12

Chapter 15: Specifications (DB2 UDB)

Introduction	15-1
Data Base Definition	15-1
File Definition (FD) Statement	15-2
Segment Definition (LS) Statement	15-2
Field Definition (L0) Statement	15-4
Logical Relationship (LR) Statement	15-5
Logical File Assignment	15-8
File Selection (RF) Statement	15-8

Chapter 16: Details of Operation and Examples (DB2 UDB)

Introduction	16-1
Sample Data	16-4
Example 1 - Relational Tables Viewed as Separate Files	16-5
Example 2 - Use of the Logical Record	16-8
Example 3 - Suppression of Duplicate Segments	16-12
Example 4 - Tables With Hierarchically Organized Keys	16-16
Example 5 - Relational Tables Where the Keys are Not in Hierarchical Order	16-17
Example 6 - Logical Records From Normalized Tables	16-19
Examples 4, 5, and 6 - A Screen Display	16-21
Example 7 - A Different View of the Customer-Order Data Base	16-21
Example 8 - A Variant of a Different View	16-23
Example 9 - The Organization Chart	16-24
Example 10 - The Suppliers of Red Parts	16-27
Example 11 - Another SQL Comparison	16-29
Example 12 - Multiple Column Browsing	16-31
Summary	16-33

Chapter 17: Updating and Designing for Referential Integrity (DB2 UDB)

Introduction	17-1
The VISION:Transact Updating of Relational Tables	17-1
Set Operations	17-3
Creation of Tables With VISION:Transact	17-6
Referential Integrity	17-6
Designing for Referential Integrity	17-8
Maintenance of Values of Fields in Separate Tables	17-9
Modification of Keys and Foreign Keys	17-14
Insertion of Rows in a Table	17-15
Deletion of Rows From a Table	17-19
The Order of Updating Rows	17-21
Summary	17-23

Chapter 18: Helpful Hints (DB2 UDB)

Introduction	18-1
Setting Up and Running a Job for DB2.....	18-1
Some Additional Hints	18-4
Performance Considerations	18-5
Commits and Rollbacks	18-6
Creating a Logical Record	18-7
Validation Against the DB2 Catalog.....	18-7
Uniqueness of Tables	18-7
Null Fields	18-7
Performance and the Generation of Output Screens From a Browse File	18-8
Tracing SQL Calls	18-9

Appendix A: Order/Item Application Screen and Database Definition Code

Appendix B: Order/Item Application Code Design Manual

Appendix C: VISION:Transact Interaction Cycle Diagram

Glossary

Index

Introduction

VISION:Transact is a non-procedural application development system and compiler that can generate online applications that execute in either the IMS TM or CICS TS environment. Its non-procedural approach automates the programming of several functions of online systems, allowing you to devote more time to the unique requirements of your application.

VISION:Transact consists of generator and executor components. An application is compiled by the VISION:Transact generator, link edited into a load library, and installed in your online system. The executor component is a subroutine library that participates in the execution of your generated application. To the target communications monitor, the executing modules are just another application that uses the standard facilities of that online environment.

This guide explains how VISION:Transact works by describing its components and showing how they work together. It contains references to four other VISION:Transact documents: the *Advantage VISION:Transact Reference Summary*, the *Advantage VISION:Transact ASL Reference Guide*, the *Advantage VISION:Transact for CICS Environment Guide*, and the *Advantage VISION:Transact Messages Guide*. These contain all of the technical details necessary for developing applications with VISION:Transact.

Note: Starting with this release, the CICS TS version of VISION:Transact supports COMMAREA programs in addition to 3270 Terminal programs. The VISION:Transact statements used to define 3270 screen layout and content are also used to define COMMAREA layout and content. Unless specifically noted otherwise, any reference to the term “screen” hereafter in this document shall be assumed to also mean “COMMAREA”.

The power and usefulness of VISION:Transact is illustrated in [Chapter 2, “A VISION:Transact Generated Application System”](#) where we present a selection of screens from an online application system developed using VISION:Transact. This system serves as the basis of most examples in the guide.

[Chapter 3, “Essentials of Data Definition”](#) discusses the essentials of data definitions required for a VISION:Transact generated application.

Chapters 4 through 8 discuss the central concepts of the VISION:Transact system:

- Application flow, its components and controls.
- Screens, their use and definition.
- Structured screens, intermediate structure, and their relationship.
- Screen requests/procedures, their use and effects.
- Dynamically synchronized data bases, their accessibility and use.

[Chapter 9, “Data Manipulation/Field Handling”](#) details how VISION:Transact processes different data types, helping you anticipate the results of data manipulation in an online application.

[Chapter 10, “Defining Data”](#) details the process of data definition you follow to enable a VISION:Transact generated application to access various data structures.

[Chapter 11, “Advanced Application Flow”](#) covers advanced application flow including screen control, extended application generation and a brief description of VISION:Transact library facilities.

[Chapter 12, “Execution Tracing”](#) describes the options available for tracing the execution of a VISION:Transact generated application.

[Appendix , “Glossary”](#) contains a helpful glossary of many essential VISION:Transact terms and concepts, while [Appendix A, “Order/Item Application Screen and Database Definition Code”](#) contains the source statements that generated the sample application used throughout the guide. [Appendix B, “Order/Item Application Code Design Manual”](#) contains a VISION:Transact design manual for the generated sample application. [Appendix C, “VISION:Transact Interaction Cycle Diagram”](#) contains a diagram of the VISION:Transact interaction cycle.

With regard to the processing of IMS data bases, this guide assumes you are familiar with such terms as data base, segment, field, path, and hierarchical structures. On the other hand, you should need only a rudimentary knowledge of communications monitors, terminals, and online application programming.

VISION:Transact Generator and the Target Communications Monitor

[Figure 1-1](#) illustrates the steps involved in application generation with VISION:Transact. The generator component of the VISION:Transact system is used to perform this processing.

Note: The VISION:Transact generator requires use of the COMLIB subsystem load library for all application development and generation runs. The COMLIB subsystem performs all VISION:Transact library access for the VISION:Transact generator.

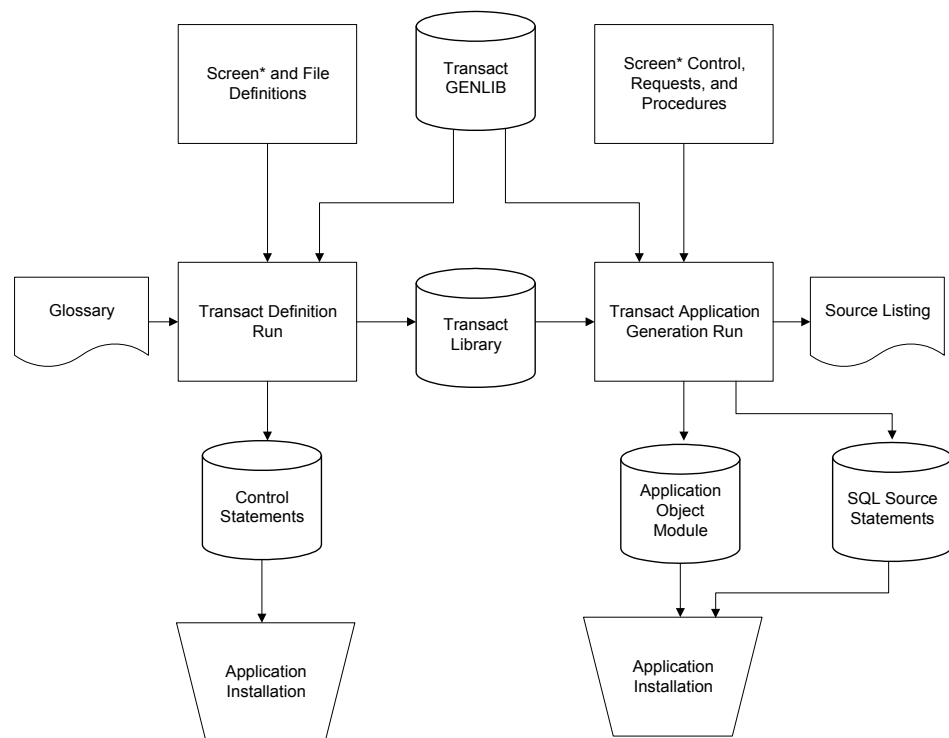


Figure 1-1 VISION:Transact and Target Communications

* "Screen" can also be COMMAREA in CICS

Note: In the previous figure, for the SQL Source Statements, if DB2 or SQL/DS databases are used, SQL source statements are output from the application generation run.

The portion of the application development process that involves the VISION:Transact generator is accomplished in three steps. First, you identify to VISION:Transact in a definition run the data structures used in the application. Second, you define your screen layouts and processing requirements for your application's display screens. Third, you formulate the processing flow and data manipulation functions of the application in a VISION:Transact application generation run. These steps are done in the MVS batch environment.

When defining files in a VISION:Transact definition run you define the data structures your application needs to access. These definitions are cataloged in the VISION:Transact library and are accessible to all other VISION:Transact generated applications. A glossary of the structures defined is output for documentation purposes.

When defining screen processing definitions in a VISION:Transact definition run you define the screen layouts and processing requirements needed by your application. This run outputs a data set of control statements used by the target communications monitor's screen formatting utilities, as well as glossaries for all items cataloged in the run and a listing of the VISION:Transact source code used as input to the run. These definitions, like the file definitions, are cataloged into the VISION:Transact library.

In a VISION:Transact application generation run (APPGEN) you identify the files to be used, the screen-to-screen flow, and the data manipulation. The primary output from an APPGEN run is an application object module ready to be link edited into an online load library. If DB2 data bases are used, then SQL source statements are output from the APPGEN run which are precompiled, assembled, and link edited with the application object module into an online load library and a bind is performed for the application plan. The source listing output during an APPGEN recommends entries to be used in the identification of the application to the communications monitor. These recommended entries may optionally be output to a data set.

You may also define screen processing definitions in an APPGEN run which are cataloged on the VISION:Transact library. If screens are included in an APPGEN, a data set is output containing control statements which are processed by the target monitor's screen formatting utilities. File definitions can be input in an APPGEN to define temporary data structures for your application, or to modify or override existing file definitions.

During an APPGEN run, you can request that execution tracing be included in your application. Execution tracing assists in the testing and debugging of your applications and is discussed in [Appendix 12, "Execution Tracing"](#). You may also request a Design Manual during APPGEN, which provides reports summarizing screen flow, data base usage, and detailed field cross references. The Design Manual is described in [Appendix B, "Order/Item Application Code Design Manual"](#).

The VISION:Transact system includes a sophisticated compiler which analyzes all input statements prior to generating an application object module. The compiler issues diagnostic messages based on conditions it detects in the application to aid in the development process. These messages are generally short explanations of some action VISION:Transact has taken or some problem VISION:Transact has detected in the source statements for a definition run or an APPGEN. A more detailed explanation of the diagnostic is located in either the *Advantage VISION:Transact Messages Guide* or *COMLIB System Messages*.

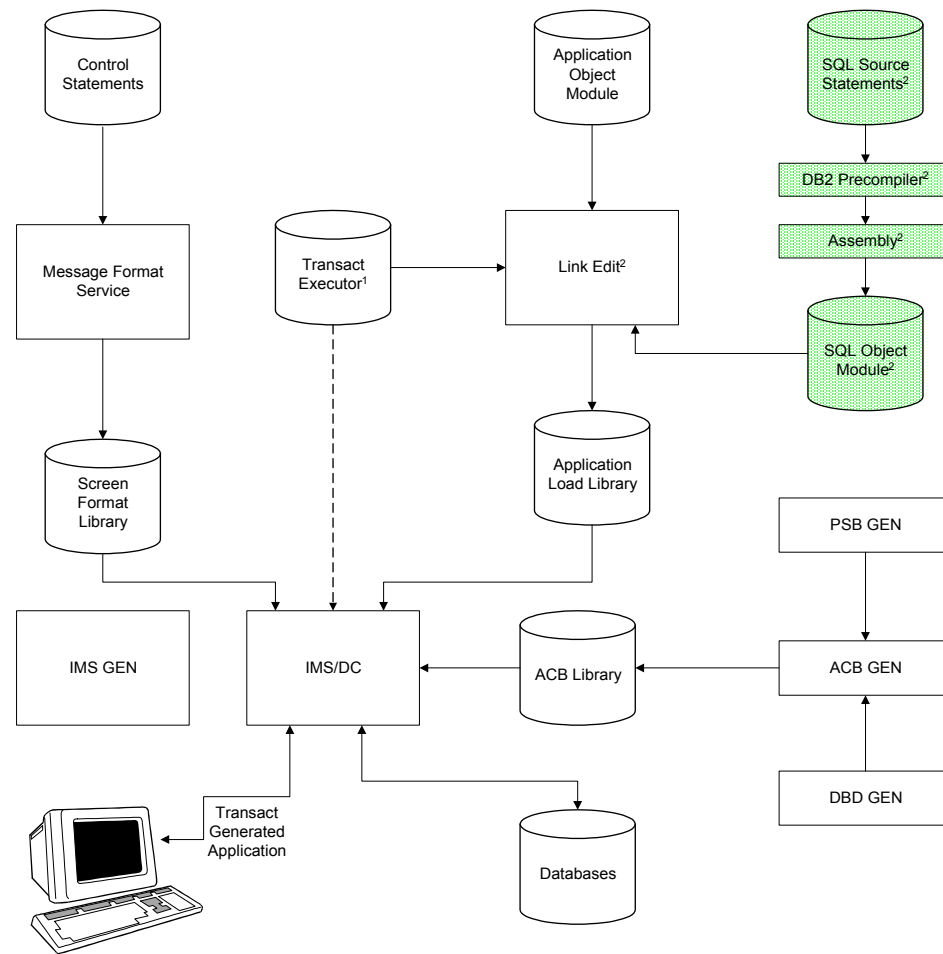
The application object module generated by the VISION:Transact compiler is input to the standard linkage editor, which creates an application load module. If DB2 data bases are used, then the SQL source statements output by the VISION:Transact generator are precompiled, assembled, and link edited along with the application object module to create an application load module; a bind is then performed for the application plan. The application load module can be placed in a load library, and it can be executed after the application has been identified to IMS TM through IMSGEN statements or to CICS through table updates.

After using the VISION:Transact generator, you are ready to add the application to your online system. The VISION:Transact executor is used in the installation and execution of your VISION:Transact generated application.

Since the VISION:Transact generator is able to compile applications for any of the supported communications monitors, its components can be discussed generically. The control statements and object module produced by the generator are communication-monitor specific, unlike the VISION:Transact generator itself. The VISION:Transact executor, however, is specific to the monitor.

The VISION:Transact Executor and Its Relationship to IMS TM

[Figure 1-2](#) depicts the elements used to execute the generated application in an IMS TM environment. The output from the definition run, the APPGEN, and the subroutines that comprise the VISION:Transact executor are used to prepare the application to execute under IMS TM.



1. **VISION:Transact** execution-time subroutines can be link edited with your application or placed in the link pack area (LPA) to be shared by all **VISION:Transact** applications or preloaded into the MP region. This is controlled by M5PARAM.
2. If DB2 data bases are used, the SQL source statements output from the APPGEN run are precompiled and assembled, and then input to the link edit along with the application object module. After the application has been link edited, the DB2 application plan must be bound.

Figure 1-2 VISION:Transact and CICS TS

The steps of application implementation depicted in [Figure 1-2](#) are the normal steps you follow in the development of an IMS TM application. These steps do not have to be adapted since the output of the VISION:Transact generation fits directly into the normal development process.

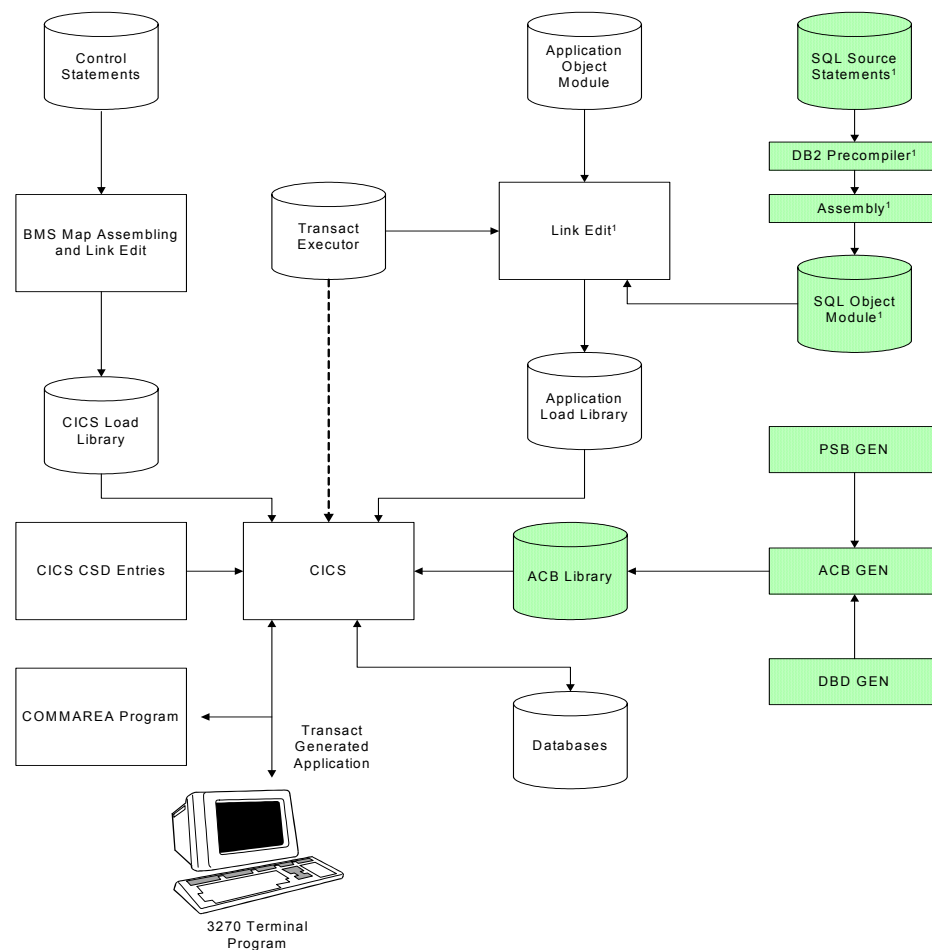
Your normal MFS processing is performed with the control statements output by a definition run containing screen definitions (or APPGEN run if screens are included) which are MFS source statements.

The application generation run outputs an application object module you link edit into a load library. If DB2 data bases are used, then the SQL source statements output from the APPGEN run are precompiled, assembled, and input to the link edit along with the application object module. The APPGEN lists APPLCTN and TRANSACT statements that you use when you identify the application to IMS TM. Recommended Program Specification Block (PSB) specifications are listed as well. The APPLCTN, TRANSACT, and PSB specifications may be optionally output to a data set.

Thus, with the output generated from VISION:Transact runs and the VISION:Transact executor you can add this new application to your IMS TM system. The MFS source statements can be processed and loaded into a screen format library, the object module(s) can be link edited into an online load library and the PSB and ACB may be generated. If DB2 data bases are used, a bind must be performed for the application plan.

VISION:Transact Executor and Its Relationship to CICS

[Figure 1-3](#) shows the elements that comprise the execution of a VISION:Transact generated application in a CICS environment. The output from the definitions, application generation, and the subroutines that comprise the VISION:Transact executor are used to prepare your application for execution under CICS.



1. If DB2 data bases are used, the SQL source statements output from the APPGEN run are precompiled and assembled, and then input to the link edit along with the application object module. After the application has been link edited, the DB2 application plan must be bound.

Figure 1-3 VISION:Transact and CICS TS

The steps of application development depicted in [Figure 1-3](#) are the normal steps you follow in the development of a CICS application. These steps do not have to be adapted since the output of the VISION:Transact generation fits directly into the normal development process.

Your normal BMS map assembly and link edit are performed with the control statements output by a definition run including screen definitions (or APPGEN run if screens are included) which are Basic Mapping Support (BMS) macro statements. Recommended Program Processing Table (PPT) entries used for table updates are output as well.

The output from the application generation (APPGEN) run is an application object module you link edit into a load library. If DB2 data bases are used, then the SQL source statements output from the APPGEN run are precompiled, assembled, and input to the link edit along with the application object module. The APPGEN output also consists of PPT and PCT entries you use to update the CICS tables. If your application references IMS data bases, recommended Program Specification Block (PSB) specifications are output and if your application accesses DB2 data bases, the RCT entry you use to update the CICS RCT table is output as well. These PPT, PCT, RCT, and PSB specifications may optionally be output to a data set. An APPGEN run also outputs BMS macro statements if screen definition statements are included.

Thus, with the output generated from VISION:Transact runs and the VISION:Transact executor you can add the application to your CICS system. The BMS Maps can be assembled and link edited into an online load library and the object module(s) are link edited into an online load library. If DB2 data bases are used, a bind must be performed for the application plan. If IMS data bases are accessed, the PSB and ACB generations must be performed.

VISION:Transact Run Types

The VISION:Transact generator has four run types: definition (catalog maintenance), application generation, source statement retrieval, and library utility. These run types determine which VISION:Transact statements may be included. Not all statements are required in each run nor are they handled in the same way in each run; how VISION:Transact processes the statements depends on the run itself. The following sections briefly describe the different run types and their purposes. For more detailed descriptions, refer to the *Advantage VISION:Transact Reference Summary*.

Definition Run – (Catalog Maintenance)

You use a VISION:Transact definition run to catalog data structures, screens, requests, and request groups in the VISION:Transact library. Once cataloged in the library, they may be retrieved for use in an application generation run. A definition run can also be used to list the names and/or contents of items cataloged on the VISION:Transact library.

Application Generation Run – (APPGEN)

The application generation run (APPGEN) is used to generate a VISION:Transact application for a particular teleprocessing environment. All elements required for a complete application may be included in an APPGEN run: file definitions, screen control, screen definitions, requests, procedures, etc.

Note: If LIBOPNA=R is specified in M5PARM, screen definitions cannot be input and individual requests cannot be cataloged in an APPGEN run. This specification can be overridden by the Library Open entry [28] on the RC statement.

[Figure 1-4](#) shows the components of an online application generated by a VISION:Transact application generation run.

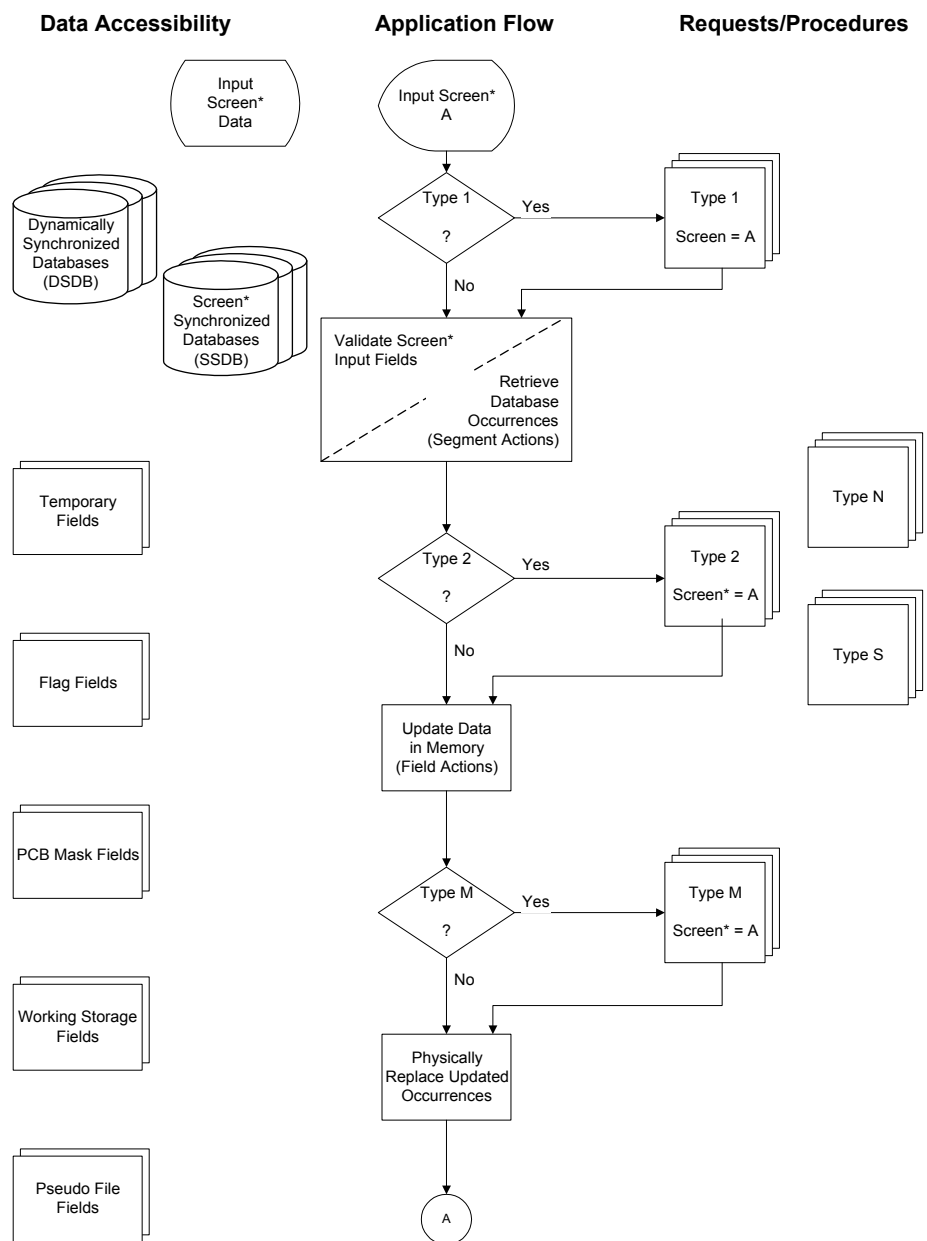


Figure 1-4 Components of a VISION:Transact Generated Application (Page 1 of 2)

* "Screen" can also be "COMMAREA section" in CICS

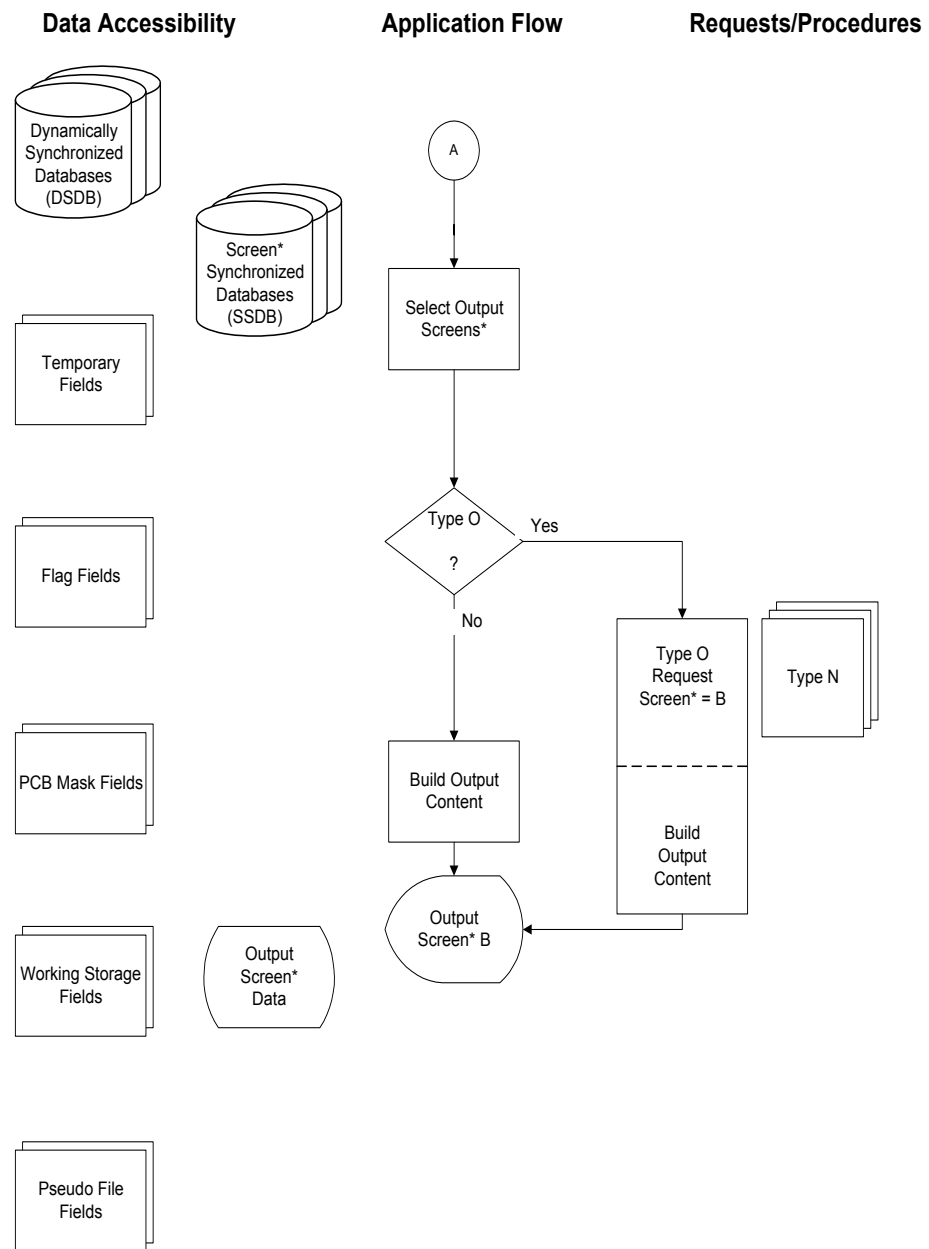


Figure 1-4 Components of a VISION:Transact Generated Application (Page 2 of 2)

* "Screen" can also be "COMMAREA section" in CICS

The components of the figure are divided into three areas of importance – data accessibility, application flow, and requests/procedures. The various data structures accessible to VISION:Transact, screen synchronized data bases (see [Chapter 4, “Application Flow”](#)), dynamically synchronized data bases (see [Chapter 8, “Dynamic Synchronization”](#)), temporary fields (local work areas defined in the APPGEN), working storage, pseudo files, and PCB mask fields (see [Chapter 10, “Defining Data”](#)) make up the components of the data accessibility area. Flag fields are system fields provided by VISION:Transact for use by your application. Flag fields are discussed in the *Advantage VISION:Transact Reference Summary* and the *Advantage VISION:Transact ASL Reference Guide*.

The online application flow starts with terminal input and typically ends with terminal output.

VISION:Transact automatically produces a formatted source statement listing every time the generator is executed. As part of the APPGEN, VISION:Transact also produces recommendations for the TP monitor and PSB statements necessary to execute the application. Sample source statement listings and TP monitor and PSB recommendations produced by VISION:Transact are shown in [Appendix A, “Order/Item Application Screen and Database Definition Code”](#).

With an RP statement, you can have VISION:Transact produce a group of reports during APPGEN which summarize screen-to-screen flow, data base usage, and field usage for your application. These reports are known as the Design Manual and are described and shown in [Appendix B, “Order/Item Application Code Design Manual”](#).

During an APPGEN run, you can request that execution tracing be included in your application using a ?T statement. Execution tracing assists in the testing and debugging of your applications and is discussed in [Chapter 12, “Execution Tracing”](#).

Source Statement Retrieval Run

A source statement retrieval run (SSR) is used to retrieve items from a VISION:Transact library and output the items in source statement form to a data set (M5SSOUT). Additional statement types may be included in a source statement retrieval run and are written directly from M5INPUT to M5SSOUT. VISION:Transact statements retrieved in this way may be used as input to a subsequent VISION:Transact run.

Library Utilities

The utility program (M5UTIL) is used to create and maintain VISION:Transact libraries. All library utilities (INIT, DUMP, REST, COND, COPY, and MERG) may be performed and may be included in a single utility run. The default library is M5LIB, but you can override the default by specifying a particular library on a utility control (UC) statement.

Statement Types

Included in each of the run types are VISION:Transact statements. Each statement has a specific function; the type of run stipulates how the statements are handled. The following is a brief discussion of the statement types. For more detailed information, see the *Advantage VISION:Transact Reference Summary*.

Run Control Group

The first group of statements input to a run are the run control group consisting of run control, item tracking, application run files, and optional run parameters or execution tracing options. Run control statements specify the type of run as well as other application specific information such as transaction identifier, load module name, teleprocessing monitor, etc.

In a definition run, the run control group specifies the teleprocessing monitor for an application as well as the primary and any multiple VISION:Transact libraries needed by the run.

In an application generation run, the run control group is used to specify the application name, transaction name, teleprocessing monitor, operating system, debug level, load module name, PSB name, application files, and special processing requirements for the application. During an APPGEN run, you may also request multiple VISION:Transact libraries to be searched, application documentation to be output, or execution tracing to be included in your generated application.

In a source statement retrieval run, the run control group is used to specify the primary and any multiple libraries needed in order to retrieve items in source statement form.

File Definition Statements

The file definition statements are used to define data structures so they can be accessed by VISION:Transact applications. (You must assign unique field names to the data defined in each file and identify the data characteristics of those fields. Each definition is given a name under which it is cataloged on the VISION:Transact library.) There are four types of data structures that can be defined to VISION:Transact using file definition statements: data bases (including relational and fast path data bases and ISAM/VSAM files), working storage, pseudo files, and PCB masks. Note that in this guide, the term "data base" is used to refer to any external file accessed from an application (other than an IMS scratch pad area or CICS temporary storage).

To define an IMS or fast path data base, each segment is assigned a name. Because VISION:Transact generates DL/I calls based on the names in the file definition, names must agree with those assigned in the Data Base Description (DBD). These include the file identification (for IMS data bases this is the DBDNAME), segment names (SEGM NAME=), and key and search fields. In addition, segment sizes must agree between the VISION:Transact file definition and the DBD.

For a DB2 relational data base, each segment in the file definition represents a relational table. Since VISION:Transact generates SQL statements based on the names in the file definition, table names and column names must agree with those defined to DB2.

When defining VSAM and ISAM data bases, the essential entry is the file identification name, which must match the file name in the File Control Table (FCT) identified to CICS.

Working storage is defined to VISION:Transact in the same manner as data bases, except that the working storage definitions must describe a single segment. At the beginning of each interaction (discussed in [Chapter 4, "Application Flow"](#)), working storage fields are initialized to the value appropriate to their data type (blanks or zeroes).

Pseudo files are defined to VISION:Transact in a way similar to data bases and may be hierarchical. The function of a pseudo file is to provide a means of building/saving data in a structured work area.

PCB masks are defined like other data areas; they define the standard IMS PCB format.

See [Chapter 10, "Defining Data"](#) for more information concerning working storage, pseudo files and PCB masks.

In a definition run, file definition statements may create, update, replace, or delete file definitions on the VISION:Transact library. File definition statements included in an APPGEN pertain to files used for that specific application only. If you input file definition statements for an item on the library, then they either replace or modify the file definition only for that application. If you input file definition statements for an item not on the library, VISION:Transact temporarily creates a new file definition for use only in that application.

For further information see [Chapter 3, “Essentials of Data Definition”](#) and [Chapter 10, “Defining Data”](#) of this guide, Chapters 3 and 4 of the *Advantage VISION:Transact for CICS Environment Guide*, and the *Advantage VISION:Transact Reference Summary* (by statement type).

Screen/COMMAREA* Definition Statements

You use VISION:Transact screen definition statements to define the layout and processing requirements for your application's display screens. Screen definitions are used to layout the fields on the screen, to indicate how those fields are used, and to describe the processing that may be initiated against the data bases using the data from the screen display. If CICS is the communications monitor, VISION:Transact outputs BMS macro statements to be assembled and link edited into a CICS load library. With the BMS macro statements, VISION:Transact lists the PPT table entries (optionally outputting them to a data set) you should use to define the BMS maps to CICS. If IMS TM is the communications monitor, VISION:Transact outputs MFS source statements to be processed by the MFS utilities and placed in an IMS format library.

Screen definition statements may be included in a definition run as well as an application generation run.⁽¹⁾ Statements included in a definition run create, delete, or replace screen definitions in the VISION:Transact library. MFS or BMS source statements are automatically generated for valid screen definitions. Screen definition statements included in an application generation run also create, delete, or replace screen definitions in the VISION:Transact library. MFS or BMS source statements are automatically generated for valid screen definitions. If included in a definition or application generation run, screen definitions are cataloged on the VISION:Transact library prior to validation. Syntax validation does occur and VISION:Transact provides messages for any syntax errors found in the screen definition.

Note: Screen definition statements can only be included in an application generation run when the library is opened for update (see the library open specification in M5PARM and on the RC statement).

Further information regarding screen definitions may be found in [Chapter 5, “Screen/COMMAREA Layout and Screen/COMMAREA Fields”](#) of this guide and the *Advantage VISION:Transact Reference Summary*.

Screen/COMMAREA* Control Statements

Screen control statements enable you to describe the screen-to-screen flow of the application and are included in an application generation run. For each input screen within the application, you can specify criteria to select the subsequent output display, to route output to alternate destinations, or to terminate the current application.

Screen control is discussed in detail in [Chapter 4, “Application Flow”](#). Advanced screen control capabilities are discussed in [Chapter 11, “Advanced Application Flow”](#).

* The terms “screen” and “COMMAREA” are interchangeable in this context.

Requests/Procedures

Often an online application includes unique data validation or manipulation prior to updating a data base or displaying a screen. The VISION:Transact vehicle for performing these functions is the *request* or *procedure*. A request or procedure is a module of code that performs data validation or manipulation at a specific point in the flow of an online application.

Requests included in a definition or application generation run may be replaced, added to, or deleted from the VISION:Transact library. Like screen definition statements, requests are not validated prior to cataloging. If a request is cataloged with errors, it must be replaced with the corrected version. Requests included in an APPGEN run can be for temporary use for that specific APPGEN; they need not be cataloged to the VISION:Transact library.

Procedures can be included only in an APPGEN run and are for temporary use for that specific APPGEN.

Requests/Procedures are described in detail in [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#).

Comments

Comments (AA statements) can be included in any run type.

VISION:Transact Library

The VISION:Transact library is used for the storage and maintenance of file and screen definitions used in the VISION:Transact system. It may also be used to store requests used in VISION:Transact applications. In addition, you may include item tracking statements for items cataloged on the library. Item tracking statements are used to assign expiration date/retention period data to items updated or created on the library as well as creator or updater identification data. See Chapters 3 and 4 of the *Advantage VISION:Transact for CICS Environment Guide* for details on storing definitions and requests on the VISION:Transact library. See Chapter 8 of the *Advantage VISION:Transact for CICS Environment Guide* for information on initialization and maintenance of the VISION:Transact library.

A VISION:Transact Generated Application System

To illustrate many VISION:Transact concepts and facilities, we have implemented an application system that maintains two data bases, a Customer data base (CUSTOMDB) and an Item data base (ITEMDB). The structure of the Customer data base is shown in [Figure 2-1](#).

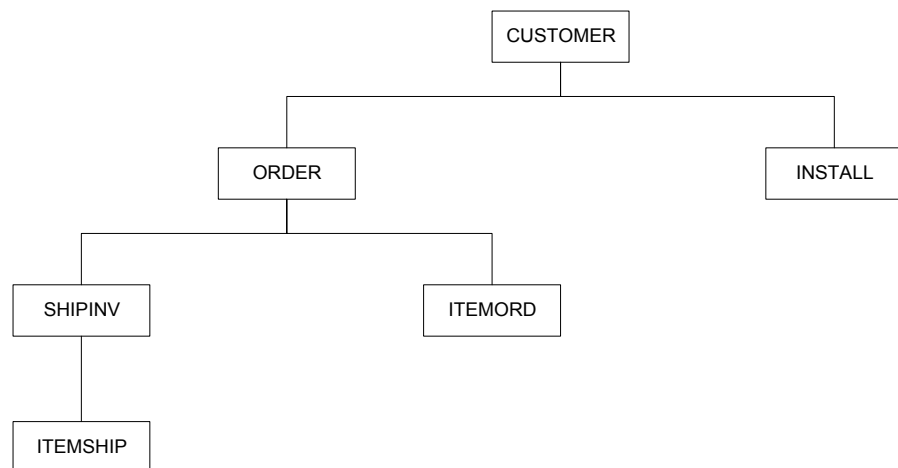


Figure 2-1 Customer Data Base

Our application system also updates and browses an Item data base, whose structure is shown in [Figure 2-2](#).

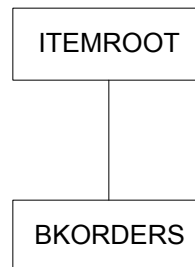


Figure 2-2 Item Data Base

The Customer and Item data bases are subsets of data bases which Computer Associates maintains. The Customer data base contains fictional information about customers' software installations, as well as their orders for both documentation and materials related to those software products. The Item data base contains detailed information about the documentation and related materials which may be ordered by the customer. We have implemented a VISION:Transact generated application system to maintain these data bases. In these examples, we are including only enough data base segments and fields within each segment to illustrate the capabilities VISION:Transact provides.

The root segment of the Customer data base contains basic customer information. The installation segment (INSTALL) extends customer data to include each Computer Associates product which is installed at each physical customer site. Thus, if a customer has two computer sites and each site has two products installed, that customer's data base record will have four installation segment occurrences.

An ORDER segment occurrence is created for each customer order.

An occurrence of the item ordered segment (ITEMORD) is created for each item which the customer orders. An item is any of the over 700 manuals, forms, binders and brochures which support our various software products.

The item number is the key of the ITEMORD segment in the Customer data base and is also the key to the root of the Item data base. Our VISION:Transact generated application system uses it to tie the two data bases together. The item number field name is ITEMORD in the Customer data base and ITEMNO in the Item data base. Different names for the item number fields in the two data bases were chosen to make the explanation easier. VISION:Transact imposes no restriction on names across data bases, and we could have used the same name.

The ITEMORD segment in the Customer data base contains the item number and the quantity of items ordered. The ITEMROOT segment in the Item data base includes the item number and the item price. Only the item numbers of those items that have been ordered are in the Customer data base, but the item numbers of all the items are in the Item data base. We use the Item data base to verify that the items ordered are carried by Computer Associates.

The VISION:Transact facility which allows our generated application system to bridge these two data bases, using the item number on the screen, is screen synchronization. Screen synchronization is discussed in [Chapter 4, "Application Flow"](#) and [Chapter 6, "Structured Screens/COMMAREA And The Intermediate Structure"](#).

Our application system also browses the Item data base. The browse function is performed using a VISION:Transact facility called dynamic synchronization, which is discussed in [Chapter 8, "Dynamic Synchronization"](#).

The system consists of the following screens:

ORDMENU4	A menu screen which invokes any of five follow-on screens, one of which is used to end the application. It allows the terminal user to enter either a customer or an item number, depending on which follow-on screen is desired.
CUSTUPD4	Allows the terminal user to update CUSTOMER or ORDER segment information, or to select an order to look at in more detail.
ORDUPDT4	Allows the terminal user to update existing ITEMORD segment information, or to add new items to an order.
ORDADD4	Allows the terminal user to add a new order for a customer, or to add items to an existing order.
ITMBRWS4	Allows the terminal user to browse items using a generic key from the menu, or to enter a new range to continue the browse, or to select an item to look at in more detail or update.
ITMBKOD4	Allows the terminal user to page through all the back-orders for an item, and to select a back-order to fill.
ITMUPDT4	Allows the terminal user to update or delete an existing item.
FILBKOD4	Allows the terminal user to fill a selected back-order.
ADDBKOD4	Allows the terminal user to back-order an item if there is not enough inventory to complete an order.
ENDSCRN4	Displays when the terminal user ends the application. This screen display shows the operator that a target communications monitor command is required to begin another application to continue the terminal session.

We have constructed this application system with a combination of single-screen and multiple-screen transactions. The variety is used to illustrate the flexibility of VISION:Transact application implementation. Since many data processing installations have standards that prohibit the inclusion of more than one screen display per online application, we have included many such single-screen transactions. The CUSTUPD4, ORDUPDT4, and ORDADD4 screen displays make up the multiple-screen transaction, while all other displays are contained in separate transactions.

The application primarily uses program function keys to control its selection of screen displays. For example, pressing PF10 returns the terminal user to the menu screen from anywhere in the system, while pressing Enter usually causes the same screen to be redisplayed.

The application system flow diagram in [Figure 2-3](#) shows all the screens used in the systems, as well as the sequence of screens available to the terminal user.

Using the Application System

Several different sequences of screens can be followed in this system. We will discuss one sequence in detail. It is indicated in [Figure 2-3](#) by boldfaced arrows.

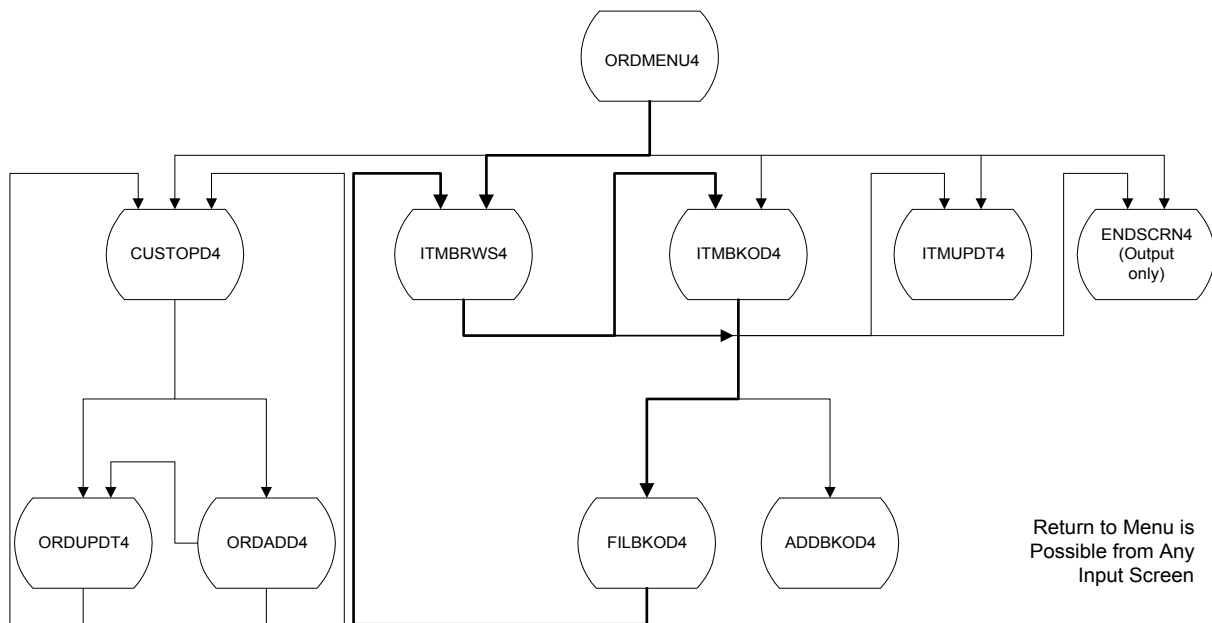


Figure 2-3 Application System Overview

To enter the application and obtain the menu screen, key in the trancode

M510

and press the Enter key. The application will then display the screen illustrated in [Figure 2-4](#). You could also key in the trancode and a screen name, if you wanted a different screen within that application displayed. For more information on initial entry to a VISION:Transact transaction, see the *Advantage VISION:Transact for CICS Environment Guide*.

```
ORDMENU4
      CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS

      PF1---CUSTOMER ADD/UPDATE
      PF2---ITEM BROWSE BY PRODUCT KEY
      PF3---BACK-ORDERED ITEM
      PF4---ITEM ADD/UPDATE
      PF5---END TRANSACTION

      ENTER CUSTOMER OR ITEM NUMBER

      (ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2 IS USED)
```

Figure 2-4 ORDER/ITEM Maintenance Application System: Menu Screen

In this execution of the application system we want to check the status of items whose item number begins with the letter K, using the first letter of the item number as a generic key. We enter a K and press PF2 to start a generic browse of the Item data base.

The ITMBRWS4 screen, shown in [Figure 2-5](#), is displayed as a result of our actions on the menu. Nothing catches our attention on this screen of items, so we press PA1 (in IMS) to retrieve the next screen of items beginning with K. (In CICS, PA1 would need to be defined as a page retrieval function key.)

```
ITMBRWS4                                NEXT SCREEN OF ITEMS=PA1
                                         `U' ACTION=ITEM UPDATE
                                         `B' ACTION=BACK-ORDERS
                                         `M' ACTION=MENU
```

PRODUCT RANGE	K	TO K999999	ITEM	ITEM	QUANTITY SHIPPED	QUANTITY	QUANTITY	
ACT			NUMBER	PRICE	ON-HAND	YTD	ORDERED	BACK
ION							YTD	ORDERED
			K31	12.00	724	327	327	0
			K32	5.00	1113	329	329	0
			K33	7.00	1302	107	107	0
			K34	7.00	0	466	625	159
			K35	3.00	665	72	72	0
			K36	3.00	1337	422	422	0
			K37	3.00	543	457	457	0
			K38	3.00	0	6742	6784	42
			K39	3.00	1983	0	0	0
			K40	3.00	110	550	550	0
			K41	3.00	1010	10	10	0
			K42	3.00	1988	87	87	0
			K43	3.00	0	700	720	20
			K45	3.00	1076	9	9	0
			K46	3.00	0	6742	6790	48

Figure 2-5 Item Browse Screen

We see that item K47 on the next screen of items, shown in [Figure 2-6](#), has back-orders against it, even though there are some of these items on hand.

ITMBRWS4 NEXT SCREEN OF ITEMS=PA1
 `U' ACTION=ITEM UPDATE
 `B' ACTION=BACK-ORDERS
 `M' ACTION=MENU

PRODUCT	RANGE	K	TO K999999	QUANTITY	QUANTITY	QUANTITY	
ACT	ION	ITEM	ITEM	SHIPPED	ORDERED	BACK	ORDERED
		NUMBER	PRICE	ON-HAND	YTD	YTD	
		K47	1.50	2000	1444	1241	283
		K48	3.00	1079	0	0	0
		K49	3.00	342	0	0	0
		K50	3.00	1005	0	0	0
		K51	3.00	0	3000	3347	347
		K52	3.00	798	96	96	0
		K53	3.00	1003	0	0	0
		K54	3.00	1276	55	55	0
		K55	3.00	203	140	140	0
		K56	3.00	24	1976	1976	0
		K57	3.00	1100	900	900	0
		K58	3.00	0	153	899	746
		K59	3.00	507	83	83	0
		K60	3.00	568	0	0	0
		K62	3.00	581	435	435	0

Figure 2-6 Item Browse Screen

In [Figure 2-7](#), we enter a B action opposite item K47 to look at the back-orders for that item.

ITMBRWS4 NEXT SCREEN OF ITEMS=PA1
 `U' ACTION=ITEM UPDATE
 `B' ACTION=BACK-ORDERS
 `M' ACTION=MENU

PRODUCT	RANGE	K	TO K999999	QUANTITY	QUANTITY	QUANTITY	
ACT	ION	ITEM	ITEM	SHIPPED	ORDERED	BACK	ORDERED
		NUMBER	PRICE	ON-HAND	YTD	YTD	
	b	K47	1.50	2000	1444	1241	283
		K48	3.00	1079	0	0	0
		K49	3.00	342	0	0	0
		K50	3.00	1005	0	0	0
		K51	3.00	0	3000	3347	347
		K52	3.00	798	96	96	0
		K53	3.00	1003	0	0	0
		K54	3.00	1276	55	55	0
		K55	3.00	203	140	140	0
		K56	3.00	24	1976	1976	0
		K57	3.00	1100	900	900	0
		K58	3.00	0	153	899	746
		K59	3.00	507	83	83	0
		K60	3.00	568	0	0	0
		K62	3.00	581	435	435	0

Figure 2-7 Item Browse Screen With Selection

The screen display in [Figure 2-8](#) shows the first of two screens of back-orders for item K47.

ITMBKOD4

```

ITEM NUMBER K47      PRICE      1.50  QUANTITY/BOX  50
QUANTITY ON HAND    2000  SHIPPED YTD    1444
ORDERED YTD        1241  QUANTITY BACK-ORDERED    283
REORDER POINT      100  PRODUCT CODE 02
*****
* SPECIFIED SCREEN OF BACK-ORDERS=SCREEN + ENTER *
* NEXT SCREEN OF BACK-ORDERS=ENTER              *
* FILL A BACK-ORDER='X' + PF8                   *
* ADD BACK-ORDER=PF9   RETURN TO MENU=PF10       *
*****
      FILL    BACK    CUSTOMER ORDER    SCREEN
      BACK    ORDER    NUMBER    NUMBER    1 OF2
      ORDERS NUMBER
          K470001 00028    28002
          K470005 01024    23444
          K470006 01025    20021
          K470007 00048    48004
          K470008 00002    20001
          K470009 00010    10555
          K470010 00028    28444
          K470011 00738    73801

```

Figure 2-8 Item Back-Order Screen

Since there are more items on hand than there are items back-ordered, we will start filling back-orders beginning with the first one. We enter an X under FILL BACK ORDERS and press PF8 to display all of the pertinent information regarding this back-order.

ITMBKOD4

```

ITEM NUMBER K47      PRICE      1.50  QUANTITY/BOX  50
QUANTITY ON HAND    2000  SHIPPED YTD    1444
ORDERED YTD        1241  QUANTITY BACK-ORDERED    283
REORDER POINT      100  PRODUCT CODE 02
*****
* SPECIFIED SCREEN OF BACK-ORDERS=SCREEN + ENTER *
* NEXT SCREEN OF BACK-ORDERS=ENTER              *
* FILL A BACK-ORDER='X' + PF8                   *
* ADD BACK-ORDER=PF9   RETURN TO MENU=PF10       *
*****
      FILL    BACK    CUSTOMER ORDER    SCREEN
      BACK    ORDER    NUMBER    NUMBER    1 OF2
      ORDERS NUMBER
      x          K470001 00028    28002
                  K470005 01024    23444
                  K470006 01025    20021
                  K470007 00048    48004
                  K470008 00002    20001
                  K470009 00010    10555
                  K470010 00028    28444
                  K470011 00738    73801

```

Figure 2-9 Item Back-Order Screen With Selection

Our selection of a back-order on the ITMBKOD4 screen causes the FILBKOD4 screen in [Figure 2-10](#) to be displayed.

```
FILBKOD4  DO YOU WANT TO FILL FILL BACK-ORDER='Y' OR
          THIS BACK-ORDER (Y OR N)?          'N' + ENTER
                                           RETURN TO MENU=PF10

          ITEM NUMBER K47
          BACK-ORDER NUMBER K470001
          ITEM NAME MARK V PRODUCT SUMM
          PRICE 1.50
          QUANTITY BACK-ORDERED 115
          QUANTITY ON HAND 2000
          CUSTOMER NUMBER 00028
          CUSTOMER NAME GEORGIA PACIFIC
          ORDER NUMBER 28002
          SHIPMENT NUMBER
          SHIPPER
          INVOICE NUMBER
          DUE DATE
```

Figure 2-10 Fill Back-Order Screen

Now we need only to enter a Y in response to the question at the top of the screen and the required information to fill the back-order, as shown in [Figure 2-11](#), and press the Enter key to fill the back-order. This will cause the deletion of the current BKORDERS segment from the item data base and the insertion of a SHIPINV segment in the Customer data base.

```
FILBKOD4  DO YOU WANT TO FILL FILL BACK-ORDER='Y' OR
          THIS BACK-ORDER (Y OR N)? y          'N' + ENTER
                                           RETURN TO MENU=PF10

          ITEM NUMBER K47
          BACK-ORDER NUMBER K470001
          ITEM NAME MARK V PRODUCT SUMM
          PRICE 1.50
          QUANTITY BACK-ORDERED 115
          QUANTITY ON HAND 2000
          CUSTOMER NUMBER 00028
          CUSTOMER NAME GEORGIA PACIFIC
          ORDER NUMBER 28002
          SHIPMENT NUMBER v345
          SHIPPER cod
          INVOICE NUMBER 65432
          DUE DATE
```

Figure 2-11 Fill Back-Order Screen With Selection

However, we neglected to enter the due date for the invoice. [Figure 2-12](#) shows a VISION:Transact generated message telling us that we haven't entered a field for which entry is required on this screen.

When one or more errors are detected by a VISION:Transact generated application, the screen is redisplayed with the field in error highlighted, the cursor positioned at the field in error, and an error message displayed on the screen. Unless you specify otherwise, this error message will be on the bottom line of the screen. VISION:Transact automatically supplies error messages for many commonly occurring error conditions. The text of many of these messages can be easily changed. Our application code checks for a variety of application-specific error conditions and uses the same method of screen redisplay shown here.

None of the data base actions mentioned with regard to this screen will take place until the entire screen passes whatever validation we have implemented for it.

```

FILBKOD4          DO YOU WANT TO FILL BACK-ORDER='Y' OR
                   THIS BACK-ORDER (Y OR N)? Y          'N' + ENTER
                                                         RETURN TO MENU=PF10

                   ITEM NUMBER K47
                   BACK-ORDER NUMBER K470001
                   ITEM NAME MARK V PRODUCT SUMM
                   PRICE          1.50
QUANTITY BACK-ORDERED      115
QUANTITY ON HAND          2000
CUSTOMER NUMBER 00028
CUSTOMER NAME GEORGIA PACIFIC
ORDER NUMBER 28002
SHIPMENT NUMBER V345
SHIPPER COD
INVOICE NUMBER 65432
DUE DATE
F01 INPUT REQUIRED BUT SCREEN FIELD IS BLANK.

```

Figure 2-12 Fill Back-Order Screen With Error

[Figure 2-13](#) shows our correction of the detected error. Now we press the Enter key again to try to fill the back-order. If the data entered on the screen is determined to be valid, the data base actions will take place.

```

FILBKOD4          DO YOU WANT TO FILL BACK-ORDER='Y' OR
                   THIS BACK-ORDER (Y OR N)? Y          'N' + ENTER
                                                         RETURN TO MENU=PF10

                   ITEM NUMBER K47
                   BACK-ORDER NUMBER K470001
                   ITEM NAME MARK V PRODUCT SUMM
                   PRICE          1.50
QUANTITY BACK-ORDERED      115
QUANTITY ON HAND          2000
CUSTOMER NUMBER 00028
CUSTOMER NAME GEORGIA PACIFIC
ORDER NUMBER 28002
SHIPMENT NUMBER V345
SHIPPER COD
INVOICE NUMBER 65432
DUE DATE 071493

```

F01 INPUT REQUIRED BUT SCREEN FIELD IS BLANK.

Figure 2-13 Error Correction

After filling the back-order, we are returned to the ITMBRWS4 screen shown in [Figure 2-14](#), in case there is something else we want to do with this item. All of the remaining back-orders can be filled in this manner. We could have chosen to fill all the back-orders from one screen display, but we wanted the terminal user to check each back-order in detail before filling it. After the desired back-orders have been filled, we will return to the menu screen (entering an M action as shown) to end our application.

```
ITMBRWS4                                NEXT SCREEN OF ITEMS=PA1
                                         `U' ACTION=ITEM UPDATE
                                         `B' ACTION=BACK-ORDERS
                                         `M' ACTION=MENU
PRODUCT RANGE  K47      TO K47      QUANTITY
  ACT      ITEM  ITEM  QUANTITY SHIPPED      ORDER      BACK
  ION      NUMBER PRICE  ON-HAND    YTD YTD    ORDERED
          m      K47   1.50   1885     1559    1241     168
```

Figure 2-14 ITMBRWS Screen

Pressing PF5 from the menu screen shown in [Figure 2-15](#) ends our application system.

```
ORDMENU4

                                CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS

                                PF1---CUSTOMER ADD/UPDATE
                                PF2---ITEM BROWSE BY PRODUCT KEY
                                PF3---BACK-ORDERED ITEM
                                PF4---ITEM ADD/UPDATE
                                PF5---END TRANSACTION

                                ENTER CUSTOMER OR ITEM NUMBER

                                (ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2 IS USED)
```

Figure 2-15 ORDMENU4 Screen

[Figure 2-16](#) shows the screen that the terminal user sees when the application ends. Screens like this one are used to let the user know that a target communications monitor command is required to continue the terminal session.

```
ENDSCRN4

                                YOU HAVE ENDED
                                YOUR APPLICATION.
                                PLEASE CLEAR THE SCREEN
                                TO CONTINUE.

*****
DATE.... DEC 27, 2003
TIME.... 10.12.13
```

Figure 2-16 Application-Ending Screen

This section is an example of some of the capabilities of VISION:Transact generated applications. The coding for the complete application is in [Appendix A, “Order/Item Application Screen and Database Definition Code”](#). The remainder of this guide shows some of the coding used to implement this application system. These examples help illustrate the concepts and facilities of VISION:Transact, and add to the capabilities used in this sample application.

Essentials of Data Definition

This section introduces the definition of data bases to VISION:Transact. In the definition of IMS data bases, three essential elements must be defined to VISION:Transact.

- The length of the segment, which facilitates path calls to access segments.
- The name of the segment and its place in the hierarchy, which permit the building of SSAs.
- The name, type, length, and scale of all key and search fields required for the application.

All IMS data base specifications must match those in the DBD for that data base.

In the definition of DB2 data bases, the following items should be noted:

- A segment in a DB2 data base represents or maps a single table or view. The name of the table or view must be specified on the LS statement.
- Segments or tables defined within a DB2 data base hierarchy are related based on the rules or criteria provided on LR statements.
- The name of each table column defined within a segment must match the name defined to DB2. In addition, the type, length, and scale defined in the file definition should match the DB2 definition to avoid field conversions.

In the definition of VSAM and ISAM data bases, a few essential points must be kept in mind.

- The file identification name on the FD statement must match the file name in the File Control Table (FCT) identified to CICS.
- Record length, or, for hierarchically structured files, segment length must agree with the actual length of retrieved data.
- For each segment type in a hierarchically structured file except the root segment, the file definition must specify either the number of times the segment is repeated, or the name of a field containing the number of segment occurrences.
- A key field must be defined which will be used to access the records.

IMS TM applications cannot access VSAM or ISAM files.

VISION:Transact also offers the facility to define other data structures. These structures are temporary fields that can be application specific or screen specific (discussed in [Chapter 7, "Requests/Procedures and the Intermediate Structure"](#)); working storage fields (discussed in [Chapter 10, "Defining Data"](#)); PCB masks, which are used for user access to PCBs (discussed in [Chapter 10, "Defining Data"](#)), and pseudo files (discussed in [Chapter 10, "Defining Data"](#)). Flag fields are predefined system fields and are discussed in the *Advantage VISION:Transact Reference Summary* and the *Advantage VISION:Transact ASL Reference Guide*.

[Figure 3-1](#) again illustrates the components of a VISION:Transact generated application but highlights by bold outline those data structures that you can define to VISION:Transact (non-shaded area).

As you progress through the guide you will see that access to this data is tied to both the application flow and request or procedure logic, represented by the shaded symbols. These concepts are discussed in the following chapters. Bold outline is used in this figure throughout the guide to emphasize the particular concepts covered in that section.

For the specific details on defining these data structures, as well as others, see [Chapter 10, "Defining Data"](#).

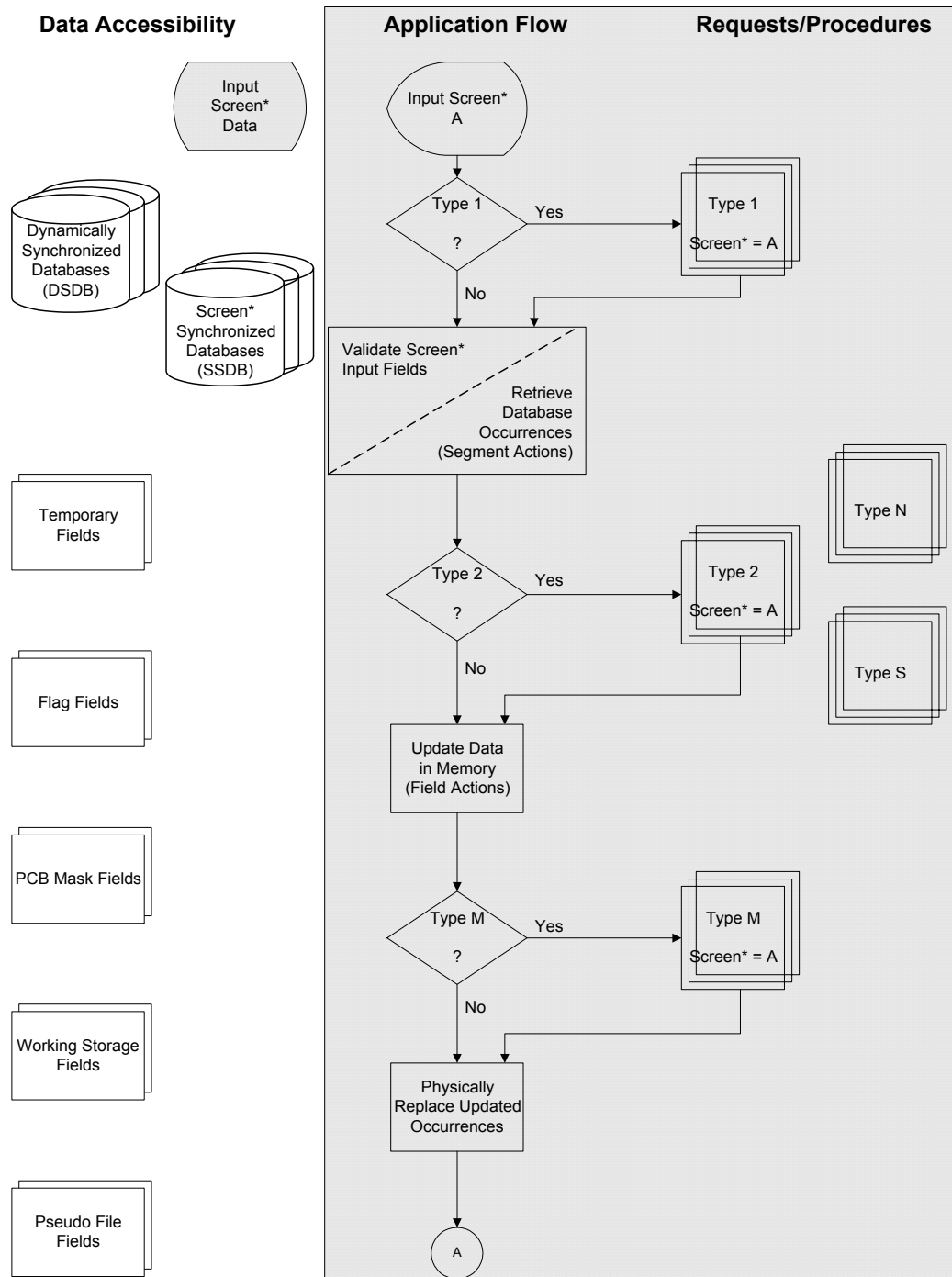


Figure 3-1 Definable Data Structures (Page 1 of 2)

* "Screen" can also be "COMMAREA section" in CICS

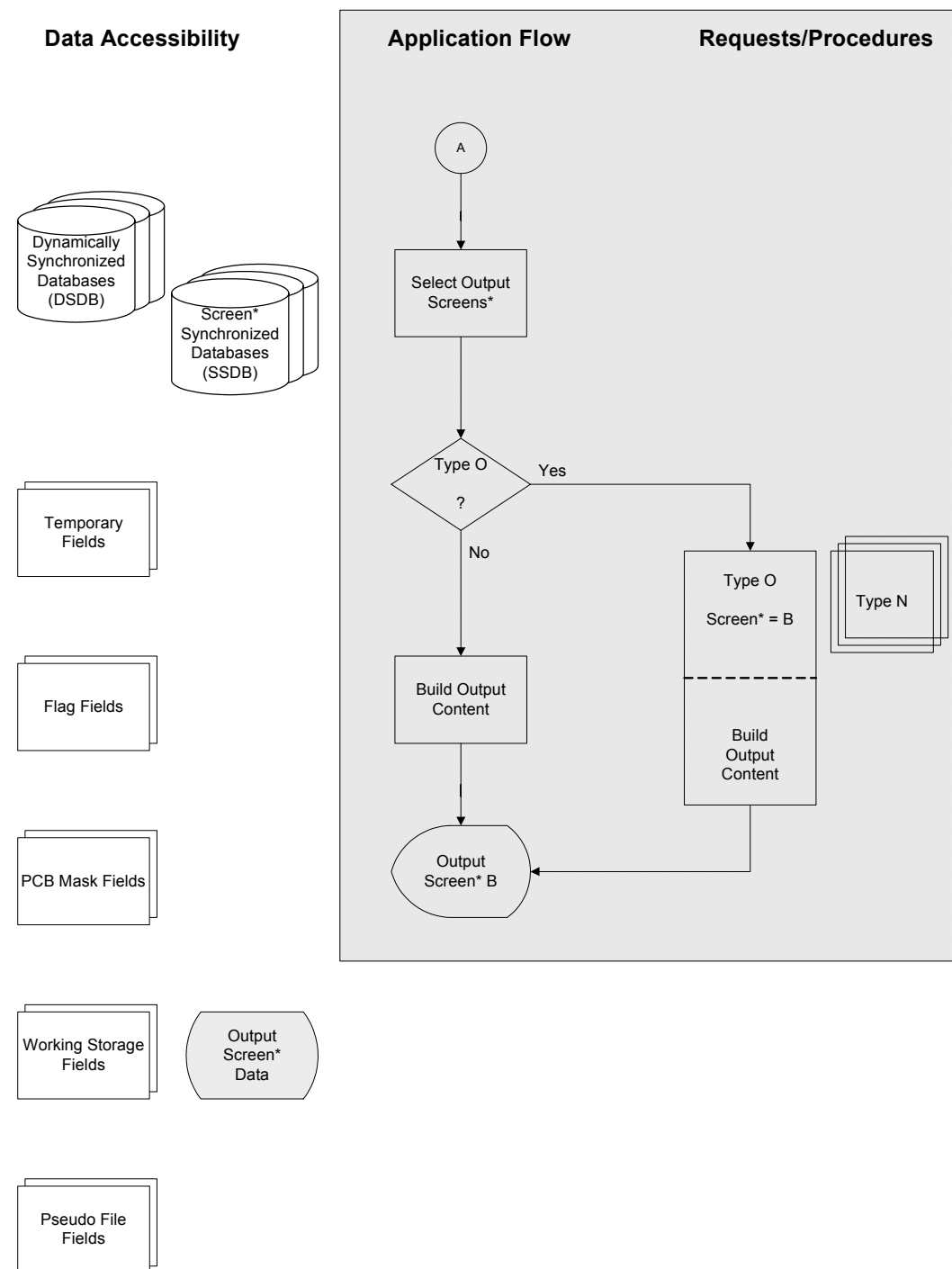


Figure 3-1 Definable Data Structures (Page 2 of 2)

* "Screen" can also be "COMMAREA section" in CICS

Application Flow

This chapter presents a simple model of the application flow, expands it, and introduces the interaction concept. As the details of the interaction area are given, the input-data/output-data processing definition is introduced. For a 3270 terminal program, this definition is referred to as a screen definition. For a CICS COMMAREA program, this definition is referred to as a COMMAREA section definition. The interaction and the screen or COMMAREA section processing definition are so interrelated that understanding both of them is necessary to understanding the flow of an application. Finally, the section discusses the VISION:Transact source code for the interaction and the screen or COMMAREA section processing definition. You must provide this source code to direct the application data flow.

Important! *The terminology throughout the remainder of this chapter uses the term "screen definition" or "screen processing definition" to identify the collection of data that flows in and out of an application. This usage is the familiar terminology relating to 3270 Terminal programs - data flows in and out of an application by using screen displays. For CICS COMMAREA programs, the term "screen definition" or "screen processing definition" is used to identify the collection of data that flows in and out of an application via the COMMAREA. Just as a 3270 Terminal program may have different screen displays for its input and output, a CICS COMMAREA program may have different sections for its inputs and outputs. Also, just as a single 3270 Terminal program interaction has one input display and one output display, the CICS COMMAREA program has one COMMAREA input section and one COMMAREA output section. When the input screen display layout for a 3270 Terminal interaction is identical to the output screen display layout, the equivalent CICS COMMAREA program would share the same COMMAREA section for both input and output.*

Please keep in mind throughout the remainder of this chapter that wherever the term "screen" is used, the concept of "COMMAREA section" can be substituted. In some places in the text, the term COMMAREA or COMMAREA section is also included for emphasis or clarification.

Execution tracing, described in [Chapter 12, "Execution Tracing"](#), can be used to output useful information concerning the flow of an application.

A Simple Model of the Application Flow

To the terminal user the application is a series of displays that pass data between the user and the program. First the application causes data to be displayed on the screen. The terminal user views the data, possibly updates it, and enters it. The application then processes the returned data which includes any updates. This application- user-application dialogue is depicted in [Figure 4-1](#).

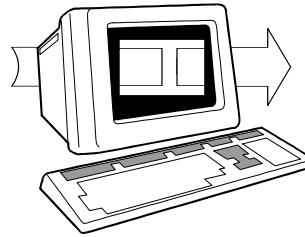


Figure 4-1 A Simple Model of an Online Application

The band entering the left side of the screen represents the data sent by the application to the terminal user. The band leaving the right side of the screen represents the data returned by the user to the application. The gap between the two bands represents the pause in the application flow as the terminal user views and possibly updates the data. The data displayed by the application may not be the same as that returned to the application, since the terminal user may update the data. Basically, the application flows through each screen, pausing for the terminal user to perform some action.

An application passes data to and from the terminal user in screen fields. A field containing data sent from the application to the terminal user is called an output screen field; it has an application field as its source. A field that contains data sent from the terminal user to the application is called an input screen field; it has an application field as its destination. Screen synchronized data bases (see [Introduction to the Screen Processing/COMMAREA Definition on page 4-6](#)) as well as temporary fields, flag fields, and working storage and pseudo files, can be used as both source and destination fields. PCBs and dynamically synchronized data bases may only be source fields since your application cannot update either of these data structures. A file defined as both screen and dynamically synchronized can have fields that are used as both source and destination fields. This distinction between source and destination is critical to your understanding of the application flow. Again, a source outputs information to a screen; a destination receives information from a screen. Collectively, the output screen fields compose the output screen content, and the input screen fields compose the input screen content. Screen fields are specified as part of the output or input screen content on the S0 statement of the screen processing definition.

[Figure 4-1](#) depicts both output and input screens inside the same display. Conceptually they are both part of the same screen. The terms screen and display are often used interchangeably in this guide. The output screen is depicted to the left of the input screen since data must be sent to the user before it can be returned. The first principle of application flow states that within one screen, the flow passes from output screen to input screen. Since most applications provide several different screens to the user, the model in [Figure 4-1](#) is expanded in [Figure 4-2](#) to include two consecutive screens displayed to the same user.

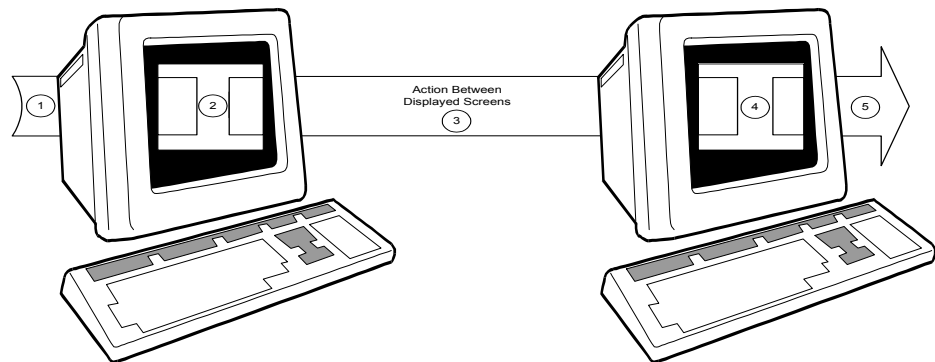


Figure 4-2 An Online Application Showing Two Consecutive Screen Displays

The dialogue associated with [Figure 4-2](#) is application-user-application-user-application. The first screen is named Screen A and the second is named Screen B. The dialogue can be expanded into the following sequence of events:

1. The application sends data to the terminal user in the A output screen content, depicted in [Figure 4-2](#) as the band entering the A output screen.
2. The terminal user views and possibly enters or updates the data in the A screen fields and returns it to the application, depicted as the pause between the A output and input screens.
3. The application processes the A input screen fields and sends more data to the terminal user in the B output screen content, depicted as the single band between the A and B screens.
4. The terminal user views and possibly enters or updates the data in the B screen fields and returns it to the application, depicted as the pause between B and input screens.
5. The application processes the B input screen content, depicted as the band leaving the B input screen.

These five events are not fully explained with the models shown above because these models do not explain the transition from the A input screen to the B output screen in event (3). The processing of screen and data base fields by the application is also still unexplained. To clarify these areas, we must discuss a concept called the interaction.

Introduction to the Interaction

The interaction represents the unit of processing time that links two consecutive screen displays. During an interaction the application processes the input screen, selects the output screen, and builds the output screen. [Figure 4-3](#) shows an interaction that starts with the A input screen and ends with the B output screen. The interaction, thus, overlaps the screen displays. The interaction together with the displays produces the application flow.

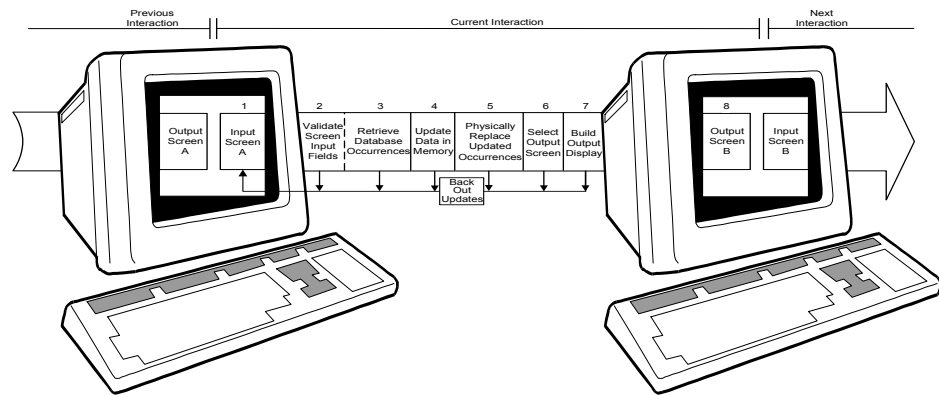


Figure 4-3 An Online Application Depicting the Organization of the Interaction

The eight basic events that take place within the time of an interaction can be broken down as follows:

1. The *current interaction* in [Figure 4-3](#) begins when the user presses the Enter or a program function (PF) key and returns the A screen data to the application.
2. The data is returned in the A input screen fields which are first validated by the application. This validation maintains the integrity of the data base, and if any input data is invalid, the screen is redisplayed to the user with a diagnostic message. In this case the A screen is redisplayed and, once the user corrects the invalid data and presses the Enter key, the interaction begins again. It is also possible that the validation process can cause early retrieval of data base segments if data base fields are used to validate screen input fields. The broken line between steps 2 and 3 in [Figure 4-3](#) indicates this possibility.
3. The validated input data can now be used by the application to retrieve data base records or segment occurrences. To accomplish this, the application uses screen fields defined to retrieve the records or segment occurrences. It is possible, even after the data is validated, that a retrieval could fail. In this case, the application redisplay the A screen with a diagnostic message concerning the unsuccessful retrieval.

4. When the proper records or segment occurrences are retrieved, the application updates destination fields in memory using the A input screen fields. The updates, however, may be incorrect and the application will redisplay the A screen with a diagnostic message.
5. After the records or segment occurrences are updated in memory, the application replaces them in the data base. If an error occurs during the physical data base update, the application redisplay the input screen and backs out any updates that have occurred. If the error occurs in an IMS TM application, the VISION:Transact generated application issues a DL/I ROLB call to perform the backout. In CICS the VISION:Transact generated application uses the ROLLBACK option of the SYNCPOINT command to backout updates of data bases that were defined as recoverable in the FCT. Note that updates are backed out any time the error cycle is invoked during an interaction (events 2 through 7 in [Figure 4-3](#)).

Note: The default action performed by VISION:Transact when the error cycle is invoked is to back out data base updates. This data base back out can be suppressed by the programmer during application generation.

Invocation of the error cycle causes the input screen to be redisplayed with a diagnostic message. This completes the primary role of the A input screen, and it is time to begin processing for the B output screen.

6. There may be other possible output screens for this application, but for now, assume that the interaction selects the B output screen as shown in [Figure 4-3](#).
7. The interaction then builds the output display based on the literals and source fields specified on the screen processing definition (see [Introduction to the Screen Processing/COMMAREA Definition on page 4-6](#)) for screen B. If an error occurs during the building of the output display (such as the application's inability to acquire enough buffer space to process selected data base segments or execution of an MS operator or MSG command), the application backs out the data base updates and redisplay the input screen with an appropriate error message.

Note: The default action performed by VISION:Transact when the error cycle is invoked is to back out data base updates. This data base back out can be suppressed by the programmer during application generation.

8. Lastly, the interaction sends this B output screen to the terminal user. This brings the discussion back to the model shown in [Figure 4-1](#) and concludes the interaction.

No explanation was given about how input screen data are validated, or how input screen data are used to retrieve data base occurrences and update them. We have not yet discussed how input and output screen fields are defined to the application. Also, the discussion about the selection of an output screen was purposely avoided. Put aside the output screen selection and concentrate on the input and output screen fields as we present the second concept of this section: the screen processing definition.

Introduction to the Screen Processing/COMMAREA Definition

The screen processing definition is a set of statements that describes the layout of a screen display used in a VISION:Transact generated application, and provides the basis for the validation and processing of that display. The statements that describe the layout of the screen and validation of its screen fields are discussed in [Chapter 5, “Screen/COMMAREA Layout and Screen/COMMAREA Fields”](#); those that describe the processing of screen field data are discussed throughout the rest of this section.

An S0 statement is used to describe the validation processing of data in a particular screen field. Most screen processing definitions include several screen fields and thus require several S0 statements. Listed below are four critical specifications of the S0 statement; three of them are discussed in this section. The edit/validation specification is discussed in [Chapter 5, “Screen/COMMAREA Layout and Screen/COMMAREA Fields”](#) along with the remaining screen processing definition specifications.

1. Input/Output.
2. Source/Destination field.
3. Action code.
4. Edit/Validation.

The Input/Output specification determines whether the screen field is used to send data to the application (input) or receive data from an application field (output) or both. Notice that the terms input and output in this context are relative to the application; that is, an input screen field is input to the application and output screen field is output from the application. Output screen fields have a single source field; however, input screen fields may have multiple (or no) destination fields. When the screen field has both a source and destination, it not only passes data to and from the screen but also from one interaction to the next. This is the most direct method of retaining data between interactions.

The source/destination specification names the particular field (data base, working storage, pseudo file, temporary, flag, or PCB mask) that supplies output screen data and/or receives input screen data. Recall that this field can be the source of data to output screen fields, or a destination for data from input screen fields. While each output screen field draws its data from a single source, VISION:Transact permits multiple destinations for input screen fields. Thus a single screen field can be used to update multiple data base fields in the same data base or multiple data bases. [Figure 4-4](#) shows coding that updates multiple data bases. Notice the two S0 statements for this screen field, one for each destination field. The first S0 statement also specifies the source field, because of the Input/Output entry of B. The second S0 statement identifies ITEMNO as a field in the data base whose qualifier has been designated as 1 (refer to [The Source of Screen/COMMAREA Synchronization on page 6-20](#) for a discussion concerning assignment of data base qualifiers).

	V				I R C E P M N				
SCREEN	STMT	SEG	S	SOURCE/DEST	/	Q	H	U	R R O U
NAME	TYPE	NO.	A	QLF FLD NAME	O	D	I	R R T D M	T

	(MULTIPLE,S0)	(1)		(ITEMORD)	(B)				(U)
	(MULTIPLE,S0)	(1)	(S)	(1,ITEMNO)	(I)				(I)

Figure 4-4 Multiple Destinations for a Single Screen Field

The action code tells VISION:Transact what is to be done with the data that the application receives in the screen field. There are two distinct types of actions, *segment* and *field*, distinguished by whether the destination is a data base key or search field or not (according to the file definition).

The following segment action codes relate the data in an input screen field to a data base record or segment occurrence and are used only if the destination input is a data base segment key or search field:

1. The *match* (M) in the action code directs the application to use the data in the screen field to retrieve an existing unique record or segment occurrence having a key or search field with the same value. This record or occurrence cannot be updated. If such a record or segment occurrence does not exist, the application redisplayes the screen with an error message. The match action may be used on a segment search field as well as a key field. This allows retrieval of a specific record or segment occurrence.
2. The *insert* (I) action code directs the application to insert a record or segment occurrence in the data base with the specified key value. The inserted record or segment occurrence can contain data specified by field actions. If this specified record or segment occurrence already exists in the data base, the screen is redisplayed with an error message.
3. The *delete* (D) action code directs the application to delete a record or segment occurrence from the data base with the specified key value. If such a record or occurrence does not exist in the data base, the screen is redisplayed with an error message.
4. The *update* (U) action code is the same as the match action code, except that the retrieved record or segment occurrence can be updated.
5. The *update/insert* (V) action code first directs the application to retrieve the record or segment occurrence with the specified key value. If the record or occurrence is not found, it is inserted. In either case the record or occurrence can be updated if field updates are specified and appropriate data is input.

6. The *repeat* (Q) action code directs the application to repeat the last segment action specified on this path of the data base. The action is inherited from the first key/search field of the record or segment or from the parent of the segment.
7. The *ignore* (X) action code directs the application to take no action against the data base record or segment occurrence with the specified key value. The only valid segment actions for dependents of that record or occurrence are X and Q. The X action may not be the only possible action provided for a key or search field; the X action must be used with dynamic actions (the IF operator).

These seven action codes manipulate the data base records or segment occurrences rather than the fields within them. The destination field named in an S0 statement containing a segment action must be defined as a key or search field in the file definition. Though they are used to retrieve, insert, delete, or ignore the records or segment occurrences and enable them for updating, they specify no actual field updates within the record or segment.

The application of one of these segment actions to a data base record or segment causes that data base to be screen synchronized. Your VISION:Transact generated application can specify segment actions only to those files defined as screen synchronized data bases (SSDB) or pseudo files. You can specify up to ten screen synchronized data bases or pseudo files for your application. [Chapter 6, "Structured Screens/COMMAREA And The Intermediate Structure"](#) describes how the action code affects the availability of data throughout the interaction. Execution tracing, described in [Chapter 12, "Execution Tracing"](#), can be used to view the segment actions selected by your application.

The following field action codes specifically update the data base, pseudo file, working storage, temporary fields, and flag fields. The action codes specify the following actions: (1) add, (2) subtract, (3) replace, (4) replace if non-blank, (5) replace with blank or zero, or (6) null action-no action is to be taken. Note that for data bases and pseudo files, segment actions must be specified in order to specify field actions for the files.

The receiving field is referred to as the destination field.

1. The add (A) action code adds the data in the screen field to the data in the destination field.
2. The subtract (S) action code subtracts the data in the screen field from the data in the destination field.
3. The replace (P) action code replaces the data in the destination field with the data in the screen field.
4. The replace if non-blank (R) action code replaces the data in the destination field with the data in the screen field only if the screen field is not blank.
5. The replace with blank or zero (B) action code requires no actual input screen data and replaces the data in a non-blank destination field with blanks if it is a character field or with zeroes if it is a numeric field.

6. The null (N) action code performs no automatic field action with data entered from the screen. It does however, allow processing of the data entered.

Thus, the screen processing definition is a powerful VISION:Transact facility that directs the flow of data through the application and its screens. The data is also validated and directed to and from the data bases according to the specification provided in the screen processing definition.

Note that a key or search field may not be changed the way a non-key data field can be updated; thus, when a segment action is used on a key or search field, the VISION:Transact generated application will automatically replace the screen data into the field when the segment or record is to be inserted, and will not replace the value on any other segment action. Key or search fields may not have field actions.

Dynamic Selection of Action Codes

By specifying action codes in your screen processing definition you have the ability to process screen input data automatically. While action codes provide a wide range of choice during programming, often the decision of which action to use is best made in light of the data entered on the screen by the terminal user. The IF operator provides this ability to dynamically choose a particular action.

The IF operator on the screen processing definition allows you to set up logical criteria that must be met for an action to be used. For simple validation the edit/validation operator sets up a comparison between the screen field and a value in the Operand B column. The IF operator, though, specifies a logical test between a "control field" and its value. If the condition specified in the logical operation is met, the action code associated with that IF operator will be applied to that occurrence of the screen field. [Figure 4-5](#) shows the coding for the dynamic actions used in the CUSTUPD4 screen.

		V	I R C E P M N					A L C								
SCREEN	SIMT	SEG	S	SOURCE/DEST	/ Q	H	U	R	R O U							
NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D	I R R T D M							

(CUSTUPD4,S0)	(	1)		(CUSTNO	)	(B)		(X)								
(CUSTUPD4,S1)								(M)	(IF) (F,ATTNID) (EQ) (C,PF006)							
(CUSTUPD4,S1)								(M)	(IF) (F,ATTNID) (EQ) (C,PF007)							
(CUSTUPD4,S1)								(U)	(IF) (F,ATTNID) (EQ) (C,ENTER)							
(CUSTUPD4,S0)	(	1)		(CUSTNAME)	(B)			(P)								
(CUSTUPD4,S0)	(	1)		(CUSTPH	)	(B)		(P)								

SCREEN	SIMT	SEGMENT	NO. OF													
NAME	TYPE	NO.	OCCURRENCES													

(CUSTUPD4,SS)	(20)		(5)													

		V	I R C E P M N					A L C								
SCREEN	SIMT	SEG	S	SOURCE/DEST	/ Q	H	U	R	R O U							
NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D	I R R T D M							

(CUSTUPD4,S0)	(20)			(I)				(N)	(NM) (ACTION)							
(CUSTUPD4,S0)	(20)			(ORDERNO	)	(B)		(X)								
(CUSTUPD4,S1)								(U)	(IF) (S,ACTION) (EQ) (C,U)							
(CUSTUPD4,S1)								(M)	(IF) (S,ACTION) (EQ) (C,X)							
(CUSTUPD4,S0)	(20)			(ORDRDATE)	(B)			(P)								
(CUSTUPD4,S0)	(20)			(ORDEPERSON)	(B)			(P)								
(CUSTUPD4,S0)	(20)			(ORDEPONUM)	(B)			(P)								
(CUSTUPD4,S0)	(20)			(ORDDUDAT)	(B)			(P)								
(CUSTUPD4,S0)	(20)			(ORDINVCN)	(B)			(P)								
(CUSTUPD4,S0)	(20)			(ORDCOMPLT)	(B)			(P)								

Figure 4-5 The Coding of Dynamically Selected Actions

Notice that the source/destination field CUSTNO has one S0 and three S1 statements that follow it. These four statements together form an Sn set, which is processed as a unit. The S0 statement specifies the default action that is applied for the set if none of the IF conditional tests are true. Each S1 statement specifies a conditional test and an action that is applied when the condition is true. A flag field containing the value of the function key depressed serves as the control field for this conditional test. Thus, the action applied to CUSTNO destination varies depending upon the function key depressed.

Control fields must be either a flag field, as used on the CUSTNO segment, or a screen field as used on the ORDERNO segment. We recommend that a single control field be used for a given destination; this avoids the possibility of two conditions being true. When two or more of these conditions are true, the action associated with the first true condition is applied to the destination field. This can result in unforeseen, possibly unfortunate, updates to your data base.

Two segment actions have particular significance as dynamically selected actions: ignore (X) and inherit (Q). In fact, the X segment action can only be used in the context of the dynamic selection of actions. Recall that the X segment action directs the application to take no action against that segment or segment occurrence from the data base. When X is selected for a particular data base segment occurrence, that occurrence is no longer accessible to the application during that interaction; it is treated as missing (see [Data Accessibility to the Interaction on page 6-7](#)) for the remainder of the interaction. When X is selected for all occurrences of a lower level data base segment, the segment is not screen synchronized. [Data Accessibility to the Interaction on page 6-7](#) explains screen synchronization and the availability of segments that have not been synchronized to the screen.

As in the example, you can allow the X to be the default action, assuring that no action is taken unless one of the conditions is true. The X segment action "turns off" the actions on all the fields of the specified destination data base segment. Thus in [Figure 4-5](#), if neither PF6, PF7, nor Enter is pressed, X is applied and no segment action is taken against that CUSTNO and no actions are taken against CUSTNAME and CUSTPH. Validation and field actions associated with destinations in data base segments for which an X segment action has been selected are not performed. However, if a screen field is input to multiple destinations (multiple Sn sets), the validation and field actions for those destinations in a data base segment with a non-X segment action, are performed. Thus, a single screen field may be validated by various criteria, depending on the selection of action codes.

An X action applied to CUSTNO "turns off" actions on the dependents of the CUSTOMER segment. If the default X is applied to CUSTNO, an X action is automatically applied to ORDERNO. A discrepancy between actions causes your VISION:Transact generated application to return a diagnostic message stating that inconsistent segment actions have been specified. Thus, your logic should assure consistency among dependent segment actions. For example, if a D action is applied to a data base segment occurrence, then all segment occurrences dependent on it should select an appropriate action, or no segment action at all. The following table shows the consistency of segment actions that must be followed in hierarchical data base segment processing.

Hierarchical Consistency of Segment Actions

D E P E N D E N T 'S	PARENT'S ACTION						
		M	U	D	I	V	X
	M	Y	Y	Note 1		Note 2	Note 3
	U	Y	Y			Note 2	Note 3
	D	Y	Y	Note 1		Note 2	Note 3
	I	Y	Y		Y	Note 2	Note 3
	V	Y	Y		Note 2	Note 2	Note 3
	X	Y	Y	Y	Y	Note 2	Y

A Y indicates that the dependent's action is consistent with the parent's action. A blank indicates it is not.

Note:

1. The existence of the dependent is validated before data base update occurs.
2. A V changes to U or I.
3. An X action is automatically forced on all dependents.

Actions are checked for consistency during application execution. You can actually see the segment actions selected by your application through the use of execution tracing which is described in [Chapter 12, "Execution Tracing"](#).

The Q action allows you to propagate an action on a segment occurrence to multiple key/search fields, or downward to dependents of that occurrence. Thus, you can dynamically apply an action to a parent segment occurrence and have that action applied to its dependent if a Q action code is specified on that dependent. [Figure 4-6](#) shows an example of the Q action code at work.

V		I R C E P M N				A L C	
SCREEN	STMT	SEG	S	SOURCE/TEST	/ Q H U R R O U	C L O E/V	OPERAND B.....
NAME	TYPE	NO.	A	QLF	FLD	NAME	O D I R R T D M
							T V N O P QLF FIELD NAME OR CONSTANT
(QEXAMPLE,S0)	(	1)		(CUSTNO	)	(B)	(Y) (X)
(QEXAMPLE,S1)							(M) (IF) (F,ATTNID) (EQ) (C,PF001)
(QEXAMPLE,S1)							(U) (IF) (F,ATTNID) (EQ) (C,ENTER)
(QEXAMPLE,S0)	(	1)		(CUSTNAME)	(B)		(R)
(QEXAMPLE,S0)	(	1)		(ORDERNO	)	(B)	(Y) (Q)
(QEXAMPLE,S0)	(	1)		(ORDDATE)	(B)		(R)
(QEXAMPLE,S0)	(	1)		(ORPERSON)	(B)		(R)
(QEXAMPLE,S0)	(	1)		(ORDPONUM)	(B)		(R)
(QEXAMPLE,S0)	(	1)		(ORDDUAT)	(B)		(R)
(QEXAMPLE,S0)	(	1)		(ORDINYN)	(B)		(R)
(QEXAMPLE,S0)	(	1)		(ORDCPLT)	(B)		(R)

Figure 4-6 The Q Action Code and the Propagation of Segment Actions

Here the action applied to CUSTNO is dynamically chosen by checking the function key pressed. And since a Q action code is specified on ORDERNO, whichever action (X,M, or U) is applied to CUSTNO is also applied to ORDERNO. Thus, the conditional tests performed against CUSTNO do not have to be repeated for ORDERNO. The Q action code thereby adds to the flexibility that the IF operator provides.

The use of multiple destination fields for a single screen field mentioned in [A Simple Model of the Application Flow on page 4-2](#) allows you to apply a different action to each destination. For example, you may want to insert into one data base a record or segment occurrence with a key field equal to the screen field value, and, in another data base, update the record or segment occurrence with the same key field value. This type of processing takes place in [Figure 4-7](#).

SCREEN	STMT	SEGMENT	NO. OF
NAME	TYPE	NO.	OCCURRENCES
(ORDADD4 ,SS)	(20)	(11)	
V I R C E P M N A L C			
SCREEN	STMT	SEG S	SOURCE/DEST / Q H U R R O U C L O E/V OPERAND B.....
NAME	TYPE	NO. A	QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT
(ORDADD4 ,S0)	(20)	(ITEMORD)	(I,V) (X)
(ORDADD4 ,S1)			(I) (IF) (F,ATTNID) (EQ) (C,ENTER)
(ORDADD4 ,S0)	(20) (S)	(1,ITEMNO)	(I) (X)
(ORDADD4 ,S1)			(U) (IF) (F,ATTNID) (EQ) (C,ENTER)

Figure 4-7 Dynamic Selection of Actions and Multiple Destinations

Two destination fields, one per Sn set, exist for this screen field: one named ITEMORD in the data base whose qualifier is a blank, and one named ITEMNO in the data base whose qualifier is the number 1. (The assignment of two Sn sets to the same screen field - S entry in V/S/A - is discussed in [Relating Screen/COMMAREA Field Designators and S0 Statements on page 5-4](#); the assignment of data base qualifiers is discussed in [The Source of Screen/COMMAREA Synchronization on page 6-20](#).) The coding specifies that if the terminal user presses any function key other than Enter, an X action is applied to both data base segments. However, if Enter is pressed, an insert action is applied to ITEMORD and an update action is applied to ITEMNO. Note that the Input Required validation is performed only if Enter is pressed, since the X action shuts off validation and field actions for any screen field whose destination is in the data base segment for which X has been selected.

Managing the Application Data Flow

You should consider three levels of data flow in your design and coding of the application: application data flow, interaction data flow, and screen processing definition data flow. In a sense a subordinate relationship exists between these different data flows, since applications are driven by interactions and interactions are driven by screen processing definitions.

[Figure 4-8](#) illustrates the three levels of data flow between three screens.

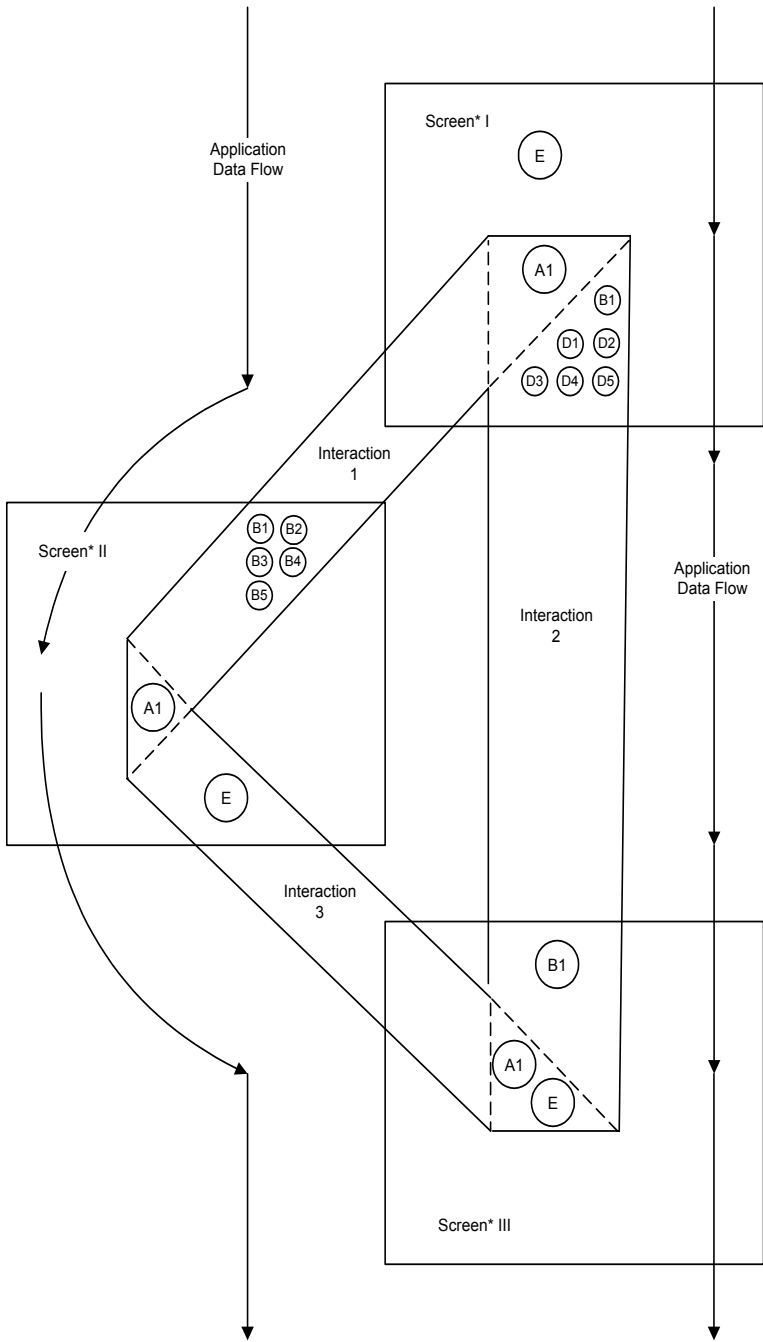


Figure 4-8 Three Levels of Data Flow in Three Screens

* "Screen" can also be "COMMAREA section" in CICS

The lettered symbols represent segment occurrences that flow through the application. The numbered symbols portray multiple occurrences of a segment. When a symbol appears within the same band in two screens, it denotes that that piece of data was passed between the screens. For example, A1 is the only data element passed from Screen I to Screen II. Screen I, however, passes both A1 and B1 to Screen III. In Screen I, B1 is an input field, but data is only entered into it when Screen I outputs Screen III. The portion of each band overlapping the screens represents either input or output screen fields depending on whether the band is leaving the screen or entering it. For example, the many occurrences of B in Screen II depict output screen fields of Screen II, while the many occurrences of D in Screen I portray input screen fields. The Input/Output specification on the screen processing definition controls this passage of data within interactions.

To illustrate the control that the screen field specifications have over the data flow, a model is provided in [Figure 4-9](#).

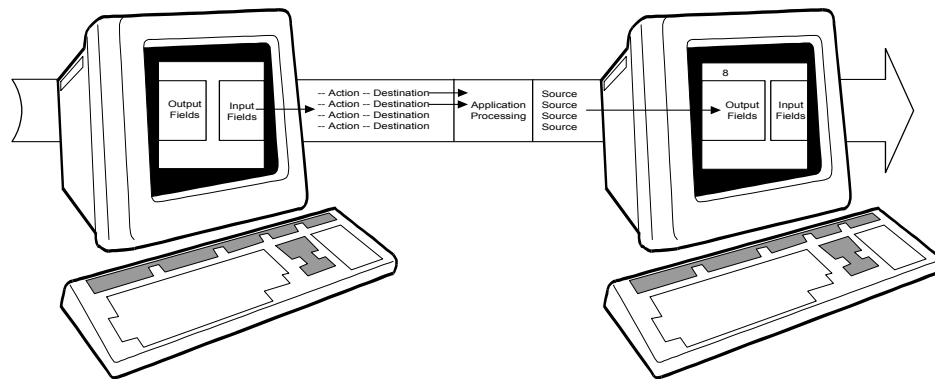


Figure 4-9 A Model of the Controlled Data Flow Between Two Consecutive Screens

The model includes the three critical screen processing definition specifications, namely source/destination, input/output, and action. The actual specifications from the two application screen processing definitions are illustrated in similar models below. These specifications are first summarized in [Figure 4-10](#).

			V	I R C E P M N				A	L	C		
SCREEN	SIMT	SEG S	SOURCE/DEST	/	Q	H	U	R	R	O	U	C L O E/V OPERAND B.....
NAME	TYPE	NO. A	QLF FLD NAME	O	D	I	R	R	T	D	M	T V N O P QLF FIELD NAME OR CONSTANT

(ORDADD4 ,S0)	(	1)	(CUSTNO)	(B)				(Y)				(X)
(ORDADD4 ,S1)												(M) (IF) (F,ATTNID) (EQ) (C,PF001)
(ORDADD4 ,S1)												(M) (IF) (F,ATTNID) (EQ) (C,ENTER)
(ORDADD4 ,S0)	(	1)	(CUSTNAME)	(O)								
(ORDADD4 ,S0)	(	1)	(ORDERNO)	(B)								(X)
(ORDADD4 ,S1)												(V) (IF) (F,ATTNID) (EQ) (C,ENTER)

SCREEN	SIMT	SEGMENT	NO. OF									
NAME	TYPE	NO.	OCCURRENCES									

(ORDADD4 ,SS)		(20)	(11)									

			V	I R C E P M N				A	L	C		
SCREEN	SIMT	SEG S	SOURCE/DEST	/	Q	H	U	R	R	O	U	C L O E/V OPERAND B.....
NAME	TYPE	NO. A	QLF FLD NAME	O	D	I	R	R	T	D	M	T V N O P QLF FIELD NAME OR CONSTANT

(ORDADD4 ,S0)	(20)		(ITEMORD)	(I,Y)								(X)
(ORDADD4 ,S1)												(I) (IF) (F,ATTNID) (EQ) (C,ENTER)
(ORDADD4 ,S0)	(20)		(ITMOTYOR)	(I,Y)								(P)

Figure 4-10 ORDADD and CUSTUPDT Screen Processing Definitions (Page 1 of 2)

V				I R C E P M N				A L C		
SCREEN	SIMT	SEG	S	SOURCE/DEST	/	Q	H	U	R R O U	
NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D I R R T D M	T V N O P QLF FIELD NAME OR CONSTANT	

(CUSTUPD4,S0)	(	1)		(CUSTNO	)	(B)			(X)	
(CUSTUPD4,S1)									(M) (IF) (F,ATTNID) (EQ) (C,PF007)	
(CUSTUPD4,S0)	(	1)		(CUSTNAME)	(O)					

SCREEN	SIMT	SEGMENT	NO. OF							
NAME	TYPE	NO.	OCCURRENCES							

(CUSTUPD4,SS)		(20)		(5)						

V				I R C E P M N				A L C		
SCREEN	SIMT	SEG	S	SOURCE/DEST	/	Q	H	U	R R O U	
NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D I R R T D M	T V N O P QLF FIELD NAME OR CONSTANT	

(CUSTUPD4,S0)	(20)			(I)					(N) (NM) (ACTION)	
(CUSTUPD4,S0)	(20)			(ORDERNO)	(B)				(X)	
(CUSTUPD4,S1)									(M) (IF) (S,ACTION) (EQ) (C,X)	

Figure 4-10 ORDADD and CUSTUPDT Screen Processing Definitions (Page 2 of 2)

An I Input/Output specification denotes that the field is an input field, while an O specifies that the field is an output field. B specifies that it is both an input and output field.

Let's suppose that the ORDADD4 screen from our sample application system is both the input and output display for interaction 1 in [Figure 4-11](#). Data is input on the ORDADD4 screen and is used to match against the CUSTOMER segment and update/insert the ORDER segment of the customer data base. The data is also used to insert the ITEMORD segment and supply the quantity ordered (ITMQTYOR).

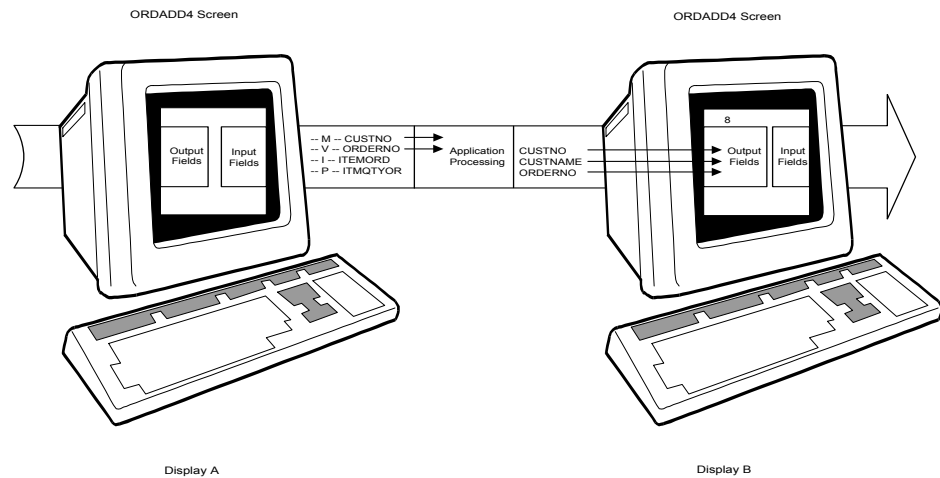


Figure 4-11 A Model of the ORDADD4/ORDADD4 Interaction

Only the input fields in the CUSTOMER and ORDER segments are specified as both input and output fields for the screen. In this way the application passes the keys of the CUSTOMER and ORDER segments and the data in those segments to the output screen of the interaction. The actions on the CUSTNO and ORDERNO segment keys cause the application to match down the data base path to ITEMORD where the user wishes to insert. ITEMORD and ITMQTYOR have an "input only" or I specification and are passed to the interaction but are not passed to the output ORDADD4 screen. CUSTNAME has an output only or O specification and is not passed from the input ORDADD4 screen to the interaction but is passed from the interaction to the output ORDADD4 screen.

Our input/output specifications determined that only the CUSTNO field and fields from the ORDER segment are both destination and source of screen data for this interaction. Fields in the ITEMORD segment are strictly the destinations of data entered on the screen and are not passed back to the screen from the interaction. Nonkey fields from the CUSTOMER segment, such as CUSTNAME, are strictly the source of information displayed on the screen and are not meant to be updated on this screen.

Next let interaction 2 in [Figure 4-12](#) involve the ORDADD4 and CUSTUPD4 screens. This second interaction having the ORDADD4 screen as an input screen is driven by a set of data base actions dynamically selected. The particular actions are selected based on the data entered by the terminal user. When this data specifies a transfer to the CUSTUPD4 screen (this happens when PF key 1 is pressed), the CUSTNO is used to match the CUSTOMER segment while no data base actions are taken against the other data base segments.

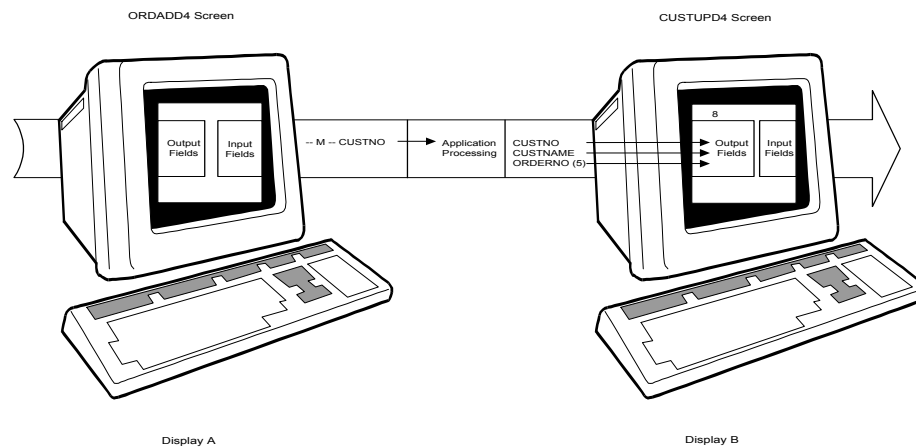


Figure 4-12 A Model of the ORDADD4/CUSTUPD4 Interaction

In the previous figure, we saw that the input fields in the CUSTOMER and ORDER segments are specified as B or "both input and output" fields for the screen. For this interaction though, the X action on ORDERNO directs the application to take no action against the ORDER segment, and so the fields in the ORDER segment are not passed to the interaction. The X action on the segment key ITEMORD also directs the application to take no action against the ITEMORD segment of the data base. All field actions within the ITEMORD segment are "turned off" as well. CUSTNAME has an "output only" or O specification and is not passed from the input ORDADD4 screen to the interaction, but is passed from the interaction to the output CUSTUPD4 screen. Though ORDERNO is not passed to the interaction from the input screen, ORDERNO is referenced on the CUSTUPD4 output. This reference causes five occurrences of ORDERNO to be passed from the interaction to the output CUSTUPD4 screen since ORDERNO is displayed in screen segment 20 which occurs five times.

Our input/output specifications combined with the dynamically selected actions determine that only the CUSTNO field is both the destination and source of screen data for this interaction (if PF1 is pressed to start the interaction). Due to the X segment action applied to the ORDER and ITEMORD segments, no fields in either segment are destinations of data entered on the screen.

Finally, let interaction 3 in [Figure 4-13](#) involve the CUSTUPD4 and ORDADD4 screens. Data is input on the CUSTUPD4 screen and is used to match against the CUSTOMER segment of the customer data base when PF7 is pressed. A match is also performed against the ORDER segment if a match (M) action is dynamically selected, but no data base action is performed if an ignore (X) action is selected.

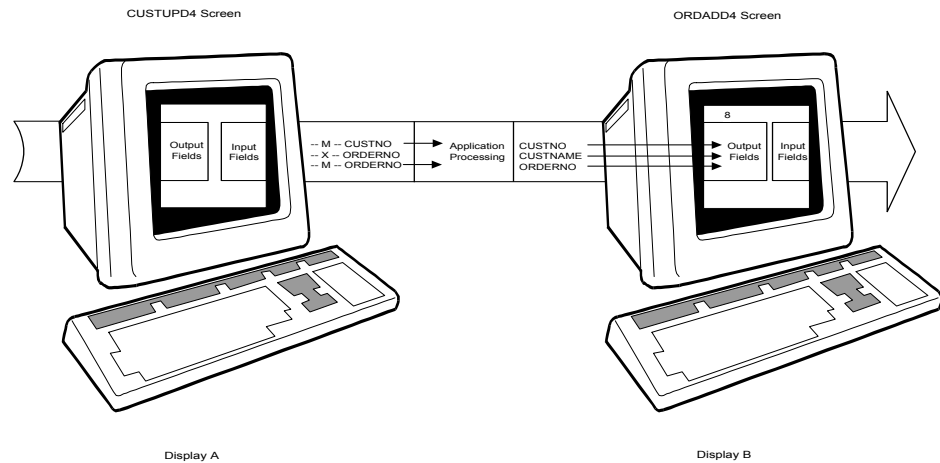


Figure 4-13 A Model of the CUSTUPD4/ORDADD4 Interaction

Here the CUSTNO and ORDERNO fields are B (both input and output) fields for the CUSTUPD4 screen. Thus, these output fields are also input fields and so are passed to the interaction, although they are ignored if an X action is dynamically selected for ORDERNO.

If you look at all three interactions together, you notice that they all pass the data base root key, CUSTNO, to the next screen. Thus, the customer number remains the same throughout a pass of these interactions, while for example, the data entered in the ITMQTYOR field is retained by the application only long enough to update the data base. These variations in the data flow are dictated by the screen processing definitions of the application system.

Besides the examples illustrated in this section, the screen processing definition also has responsibilities in regard to synchronization and structured screens. VISION:Transact synchronization is a broad concept described in [Chapter 6, "Structured Screens/COMMAREA And The Intermediate Structure"](#) and [Chapter 8, "Dynamic Synchronization"](#). Structured screens such as the CUSTUPD4 screen are discussed in [Chapter 5, "Screen/COMMAREA Layout and Screen/COMMAREA Fields"](#).

Selecting the Output Screen/COMMAREA Section of an Interaction

Screen control is the element of the application flow most visible to the terminal user. Screen control is that portion of each VISION:Transact generated application that selects the output screen during an interaction.

[Figure 4-14](#) shows the screen-to-screen flow of a few selected screens of the ORDER/ITEM application system described in [Chapter 2, “A VISION:Transact Generated Application System”](#). Screen control is the mechanism within the interaction that determines which display format is selected.

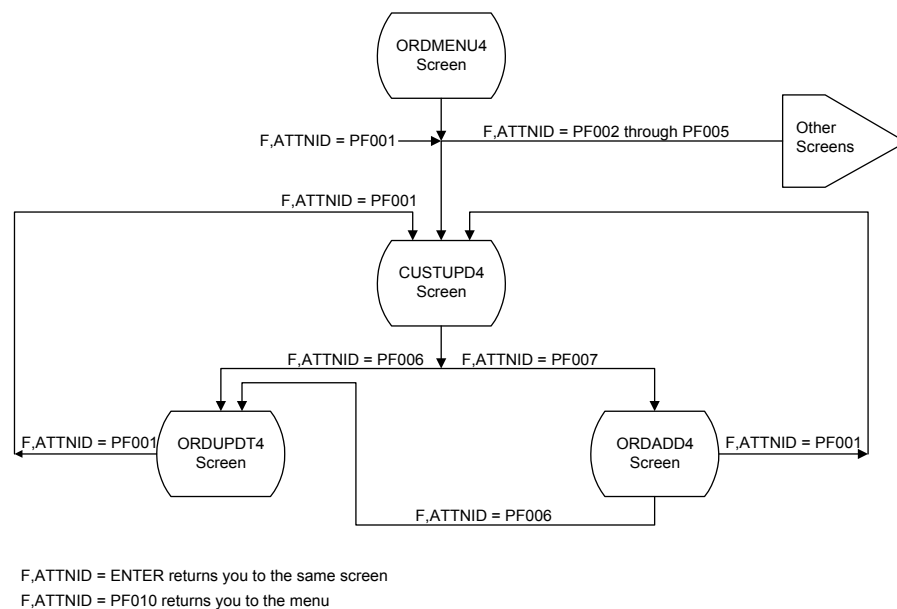


Figure 4-14 ORDER/ITEM Screen-to-Screen Flow

Screen control statements can be considered as a logical expression that states: “if the input screen to this interaction is the one named in ‘input screen name’ and if the logical expression on this (these) SC statement(s) is true, then output the screen named in ‘output screen name.’” The E (else) connector allows specification of multiple logical expressions for the same input screen. Multiple output screens (to different destinations) can be specified for a single logical condition. However, once a “true” condition has been found the application will enter the output cycle without processing any further screen control statements.

The CUSTUPD4 display in [Figure 4-14](#) may be followed by any of the four screens in the figure. [Figure 4-15](#) therefore has multiple SC statements specifying CUSTUPD4 under "input screen name." [Figure 4-15](#) contains the SC statements for all the screens discussed in [Figure 4-14](#).

INPUT SCREEN	SIMT TYPE	SEQ NO.	LOG LVL	CON CTR	OPERAND A.. QLF FIELD	OPER ATN	OPERAND B..... QLF FIELD NAME/CONSTANT	OUTPUT SCREEN
(ORDMENU4)	(SC)				(F,ATINID)	(EQ)	(C,PF001)	(CUSTUPD4)
(ORDMENU4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF002)	(ITMBRWS4)
(ORDMENU4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF003)	(ITMBKOD4)
(ORDMENU4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF004)	(ITMUPDT4)
(ORDMENU4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF005)	(ENDSCRN4)
(CUSTUPD4)	(SC)				(F,ATINID)	(EQ)	(C,ENTER)	(CUSTUPD4)
(CUSTUPD4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF006)	(ORDUPDT4)
(CUSTUPD4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF007)	(ORDADD4)
(CUSTUPD4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF010)	(ORDMENU4)
(ORDUPDT4)	(SC)				(F,ATINID)	(EQ)	(C,ENTER)	(ORDUPDT4)
(ORDUPDT4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF001)	(CUSTUPD4)
(ORDUPDT4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF010)	(ORDMENU4)
(ORDADD4)	(SC)				(F,ATINID)	(EQ)	(C,ENTER)	(ORDADD4)
(ORDADD4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF001)	(CUSTUPD4)
(ORDADD4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF006)	(ORDUPDT4)
(ORDADD4)	(SC)				(E) (F,ATINID)	(EQ)	(C,PF010)	(ORDMENU4)

Figure 4-15 Screen Control for Two Applications in the ORDER/ITEM Application System

Since ORDMENU4 is a single-screen application, it requires its own set of screen control that is included in its application generation run. The other three screens are contained in a single transaction, and so their SC statements may be grouped together for one application generation.

[Figure 4-14](#) shows that various screens may be output in response to entry of the ORDMENU4 screen; for clarity we have shown only CUSTUPD4 and its following screens in the previous diagram. The selection of an output screen display in this sample application is driven by pressing a particular function key. If PF1 is pressed, the CUSTUPD4 screen follows. Otherwise (else) a different screen follows depending on the PF key chosen. We have used the function keys as the screen control selection criteria through most of the ORDER/ITEM application system.

Note that we have used either a flag field (F,ATTNID) or a temporary field (see the complete screen control for this application system in [Appendix A, "Order/Item Application Screen and Database Definition Code"](#)) as selection criteria for screen control. These fields are used, rather than screen fields, because the input screen fields are no longer available to the application by the time the screen control executes. At this time the input display has been processed completely and the segment and field actions have been performed. The screen fields can no longer be accessed because the input screen no longer exists.

You can use screen control specifications to pass control between transactions (VISION:Transact generated applications). While the input screen must be part of the application currently being generated, the output screen can be an input screen to a different transaction (VISION:Transact generated application). This is discussed in more detail in the next subsection.

Note: Any screen referenced as an output screen will be written to the originating terminal if Operand C is blank.

Use the Connector and Logic Level specifications in screen control to specify multiple or alternative conditions which must be met for the selection of an output display format. In [Figure 4-16](#) the conditions specified in statements 001, 002, and 003 (numbers assigned in the "statement sequence" entry) must all be met for the output display format SCREEN2 to be chosen. Otherwise (statement 004), the output display chosen is SCREEN1, which happens to be the same as the input. All of these conditions must be met for the selection of the SCREEN2 format because the connector specification on statements 002 and 003 is A (for and).

INPUT	SIMT	SEQ	LOG	CON	OPERAND A..	OPER		OPERAND B.....	OUTPUT	
SCREEN	TYPE	NO.	LVL	CTR	QLF	FIELD	ATN	QLF	FIELD NAME/CONSTANT	SCREEN
(SCREEN1)	(SC)	(001)			(T,FIELDA)	(EQ)	(C,ABC		)	
(SCREEN1)	(SC)	(002)	(A)		(T,FIELDB)	(EQ)	(C,123		)	
(SCREEN1)	(SC)	(003)	(A)		(T,FIELDC)	(EQ)	(C,XYZ		)	(SCREEN2)
(SCREEN1)	(SC)	(004)	(E)							(SCREEN1)

Figure 4-16 Selection of a Single Display Format Based on Multiple Logical Conditions

In [Figure 4-17](#) the output display selected is SCREEN2 if any of the conditions specified in statements 001, 002, and 003 is true. SCREEN2 is selected under any of these conditions because the connector specification on statements 002 and 003 is O (for or).

INPUT	SIMT	SEQ	LOG	CON	OPERAND A..	OPER		OPERAND B.....	OUTPUT	
SCREEN	TYPE	NO.	LVL	CTR	QLF	FIELD	ATN	QLF	FIELD NAME/CONSTANT	SCREEN
(SCREEN1)	(SC)	(001)			(T,FIELDA)	(EQ)		(C,ABC	)	
(SCREEN1)	(SC)	(002)		(O)	(T,FIELDA)	(EQ)		(C,XYZ	)	
(SCREEN1)	(SC)	(003)		(O)	(T,FIELDB)	(EQ)		(C,123	) (SCREEN2)	
(SCREEN1)	(SC)	(004)		(E)					(SCREEN1)	

Figure 4-17 Selection of a Single Display Format Based on Alternative Logical Conditions

Use the logic level specification to combine multiple selection criteria which mix the A and O connectors. The logic level specification controls the grouping of multiple selection criteria. The use of logic levels is explained in the *Advantage VISION:Transact Reference Summary*.

There are other uses for screen control. You can use screen control to end an application by specifying an input screen name with no output screen name, as shown in [Figure 4-18](#).

INPUT	SIMT	SEQ	LOG	CON	OPERAND A..	OPER		OPERAND B.....	OUTPUT	
SCREEN	TYPE	NO.	LVL	CTR	QLF	FIELD	ATN	QLF	FIELD NAME/CONSTANT	SCREEN
(INPUT1)	(SC)				(E)	(F,ATTNID)	(EQ)	(C,PF004	)	(ENDSCRN)
(ENDSCRN)	(SC)									

Figure 4-18 Using Screen Control to End an Application

To send an output screen to more than one terminal, use the OS operator to name each output screen with its destination as shown in [Figure 4-19](#). Operand C specifies the terminal ID. When Operand C is blank, the output screen is sent to the originating terminal.

INPUT	SMT	SEQ	LOG	CON	.OPERAND A..		OPER	OPERAND B.....		OUTPUT	.OPERAND C..		
SCREEN	TYPE	NO.	LVL	CTR	QLF	FIELD	ATN	QLF	FIELD	NAME/CONSTANT	SCREEN	QLF	FLD/CONST

(INPUTA)	(SC)					(F,ATINID)	(EQ)	(C,PF008		)			
(INPUTA)	(SC)						(OS)			(OUTPUTA)			
(INPUTA)	(SC)						(OS)			(OUTPUTA)	(C,TERM	)	
(INPUTA)	(SC)						(OS)			(OUTPUTA)	(C,TERM	)	

Figure 4-19 Sending an Output Screen to Multiple Terminals

You can also send an output screen to another program/transaction. In IMS™ this causes a program-to-program message switch, where the output will be input to the transaction named in Operand C of screen control. In CICS this can be accomplished by using the OC START operator in screen control which causes the output screen to be sent to the transaction named in flag field TRANCODE. For more details on this, see the *Advantage VISION:Transact Reference Summary*.

Application Linking Via Screen Control

Often application flow becomes application system flow, in the sense that an application system contains many different transactions. This separation into multiple transactions should be done to minimize load module size, to provide transaction security or to facilitate maintenance. Therefore, the linking of these applications becomes a necessity. Application linking can be accomplished via screen control. One method is to share screens in a manner similar to [Figure 4-20](#), in which an output screen in one application is an input screen in another application, thus linking applications.

INPUT	SMT	SEQ	LOG	CON	OPERAND A..	OPER		OPERAND B.....	OUTPUT	
SCREEN	TYPE	NO.	LVL	CTR	QLF	FIELD	ATN	QLF	FIELD NAME/CONSTANT	SCREEN
(ORDMENU4)	(SC)				(F,ATTNID)	(EQ)	(C,PF001	)	(CUSTUPD4)	
(ORDMENU4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF002	)	(TIMERWS4)	
(ORDMENU4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF003	)	(TIMEROD4)	
(ORDMENU4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF004	)	(TIMUPDT4)	
(ORDMENU4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF005	)	(ENDSCRN4)	
(CUSTUPD4)	(SC)				(F,ATTNID)	(EQ)	(C,ENTER	)	(CUSTUPD4)	
(CUSTUPD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF006	)	(ORDUPDT4)	
(CUSTUPD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF007	)	(ORDADD4)	
(CUSTUPD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF010	)	(ORDMENU4)	
(ORDUPDT4)	(SC)				(F,ATTNID)	(EQ)	(C,ENTER	)	(ORDUPDT4)	
(ORDUPDT4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF001	)	(CUSTUPD4)	
(ORDUPDT4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF010	)	(ORDMENU4)	
(ORDADD4)	(SC)				(F,ATTNID)	(EQ)	(C,ENTER	)	(ORDADD4)	
(ORDADD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF001	)	(CUSTUPD4)	
(ORDADD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF006	)	(ORDUPDT4)	
(ORDADD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF010	)	(ORDMENU4)	

Figure 4-20 Linking Applications Via Screen Control

CUSTUPD4 is an output screen in the first application's screen control, while it is an input screen in the second application. Likewise, ORDMENU4 is an output screen in the second application and an input screen in the first. This sharing of screens between applications defines the process of application linking.

All application linking accomplished in the sample application system was completed in this manner. Each screen has a specific input transaction code associated with its screen processing definition. This is specified on the SD statement of the screen processing definition and is discussed in [Chapter 5, "Screen/COMMAREA Layout and Screen/COMMAREA Fields"](#). Thus, when you are using a common screen to link applications, one or several applications will output the screen, while it will be input to another application. This change of applications between output and input accomplishes the application link.

Another method for linking applications is to pass control directly to another application. Screen control can specify that the output screen be passed directly to an application rather than being sent back to the originating terminal. This method of application linking is discussed in [Chapter 11, “Advanced Application Flow”](#) and also in the *Advantage VISION:Transact for CICS Environment Guide*. This method provides easier program maintenance and reduces the size of your VISION:Transact applications.

The transaction name or trancode is used in online applications to invoke a particular application. Recall that to perform application linking, each screen has a particular input transaction code associated with it. This one- to-one relationship between input transaction code and screen is specified on the screen processing definition. When a screen is input from the terminal, the input transaction code specified on the screen's SD statement is invoked. However, you may also associate a trancode with a screen dynamically through the use of the TRANCODE flag. This flag provides you the capability to input the same screen to alternate applications.

The TRANCODE flag is an eight-byte character field that is initialized to the name of the current transaction being processed. You can manipulate this flag by request or procedure logic (discussed in [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#)) or through screen actions.

Use of the TRANCODE flag in an IMS TM application requires that it appear on the screen. Note that the flag field must appear in a singly-occurring screen segment and have a B (both) Input/Output specification and an action specified for it. MFS will map this screen field into the appropriate position in the message.

To use the flag in a CICS application, you need not name the TRANCODE flag in the screen processing definition. The content of the TRANCODE flag at output determines the transaction to invoke on subsequent input. The contents of this flag may be modified by the application at any time prior to outputting the screen.

The TRANCODE flag can be used in two ways:

1. Your application can insert into the TRANCODE flag the transaction name of the application that is to receive the next input screen. (From IMS TM you must include the TRANCODE flag on the screen, but it can be protected so users cannot modify it.)
2. You can name the TRANCODE flag as a source/destination field on the screen processing definition and allow the terminal user to enter the tran code on the input screen. For IMS TM, when the screen is entered, the transaction code entered on the screen immediately gets control. For CICS, it is the next transaction to be processed; the original transaction receives the input screen.

Interactions and Screen/COMMAREA Processing Definitions

The interaction was described earlier as the unit of processing between two consecutive screen displays. It can also be described as one execution of the VISION:Transact generated application program. To fully understand the relationship between the screen processing definition and the generated application, you must keep these related points in mind:

1. When you consider one screen processing definition (for example Screen A in [Figure 4-21](#)), you are really specifying processing that will take place over two separate interactions. The output content of the screen processing definition is processed before the terminal user sees the screen displayed (to end one interaction), while the input content of the screen is processed after the terminal user inputs the screen (to begin the next interaction).
2. When you consider one interaction (for example, the Current Interaction in [Figure 4-21](#)), you are really looking at the processing specified on two separate screen processing definitions: the input content of one screen, and the output content of a second or the same screen.

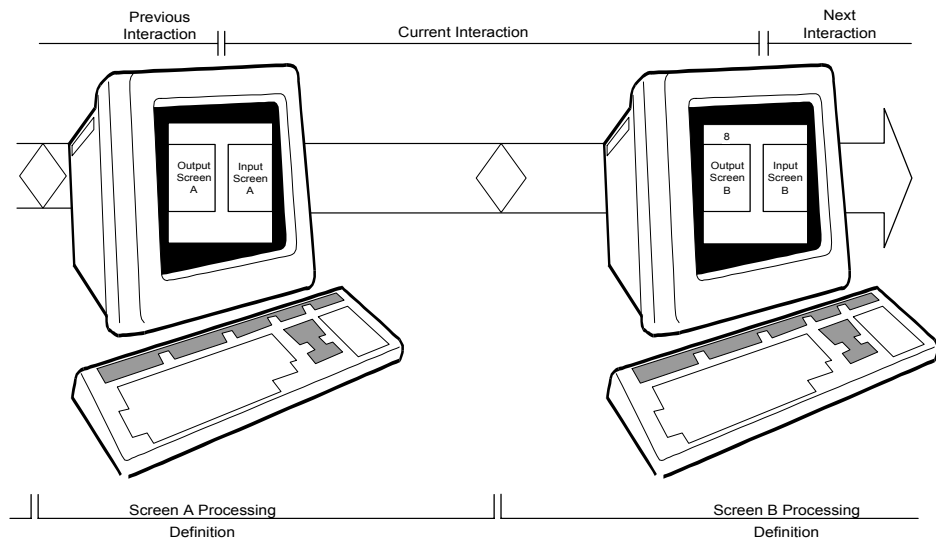


Figure 4-21 Relationship Between Screen Processing Definitions and Interactions

Application Flow Summary

[Figure 4-22](#) highlights the application flow components of a VISION:Transact generated application.

With this summary we introduce the full diagram of a VISION:Transact generated application so that you can see how each concept covered in the guide fits into the picture. The horizontal lines also appear now for the first time to help relate in a temporal manner the three areas illustrated in the diagram - data accessibility, application flow, and requests/procedures. The basic interaction, which is discussed in detail in this section and highlighted by bold outline in the figure, starts with the processing of the input screen. Processing begins with the validation of screen fields, which is discussed in detail in [Chapter 5, "Screen/COMMAREA Layout and Screen/COMMAREA Fields"](#). The validation is followed by the retrieval of data base segment occurrences. This retrieval is based on segment actions specified in the screen processing definition, and can be invoked during validation if data base fields are used for that purpose. Then field actions specified on the screen processing definition are applied against destination fields in memory. After the field actions are completed, the updated segments are replaced (possibly inserted or deleted) in the data bases, and the output screen is selected, based on screen control specifications. Finally, the output display is built based on the source fields and sent to the terminal user, ending the interaction.

For more information regarding application flow and screen control functions see [Chapter 11, "Advanced Application Flow"](#). For information on tracing the application flow and segment actions selected as well as other items, see [Chapter 12, "Execution Tracing"](#).

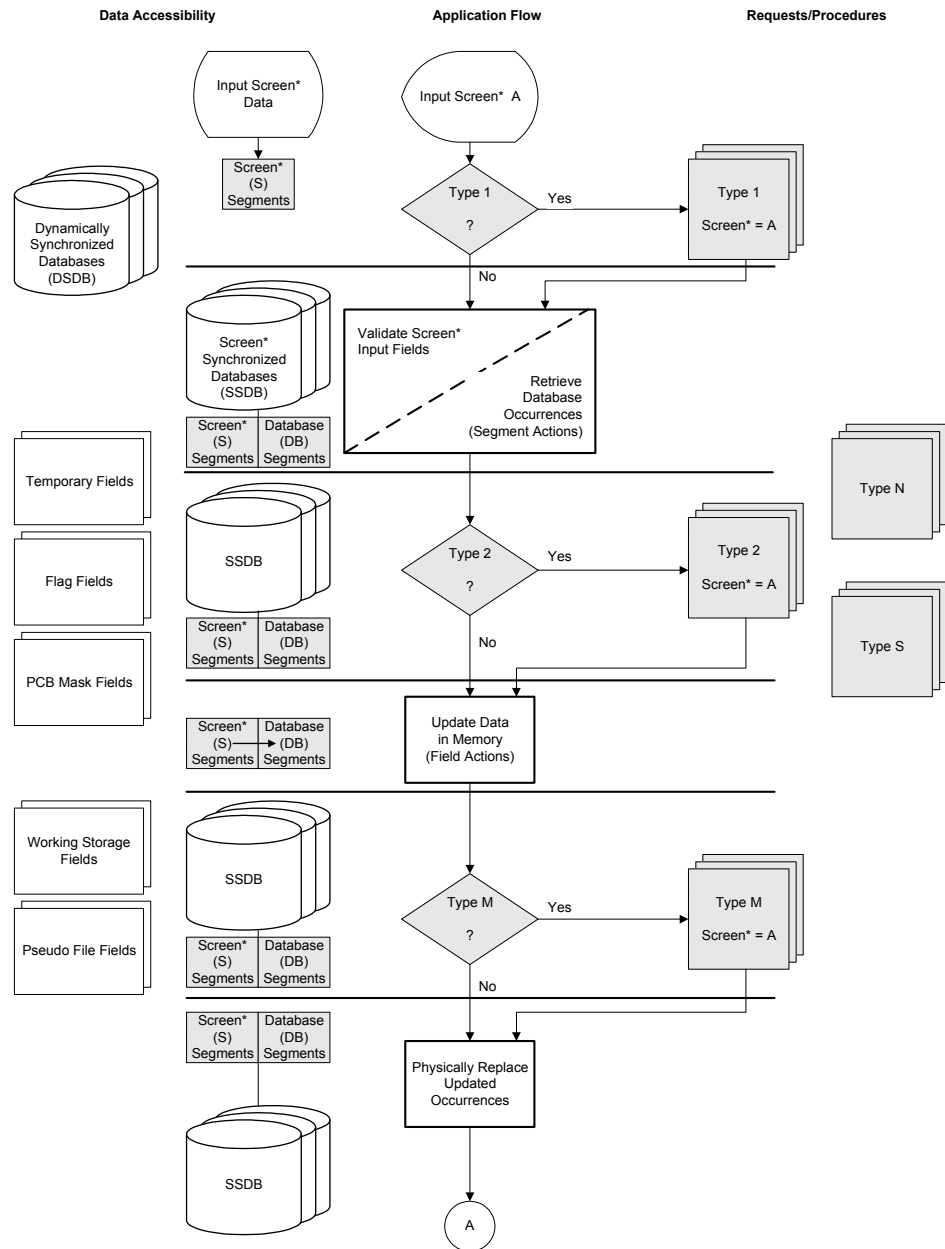


Figure 4-22 The Basic Interaction (Page 1 of 2)

* "Screen" can also be "COMMAREA section" in CICS

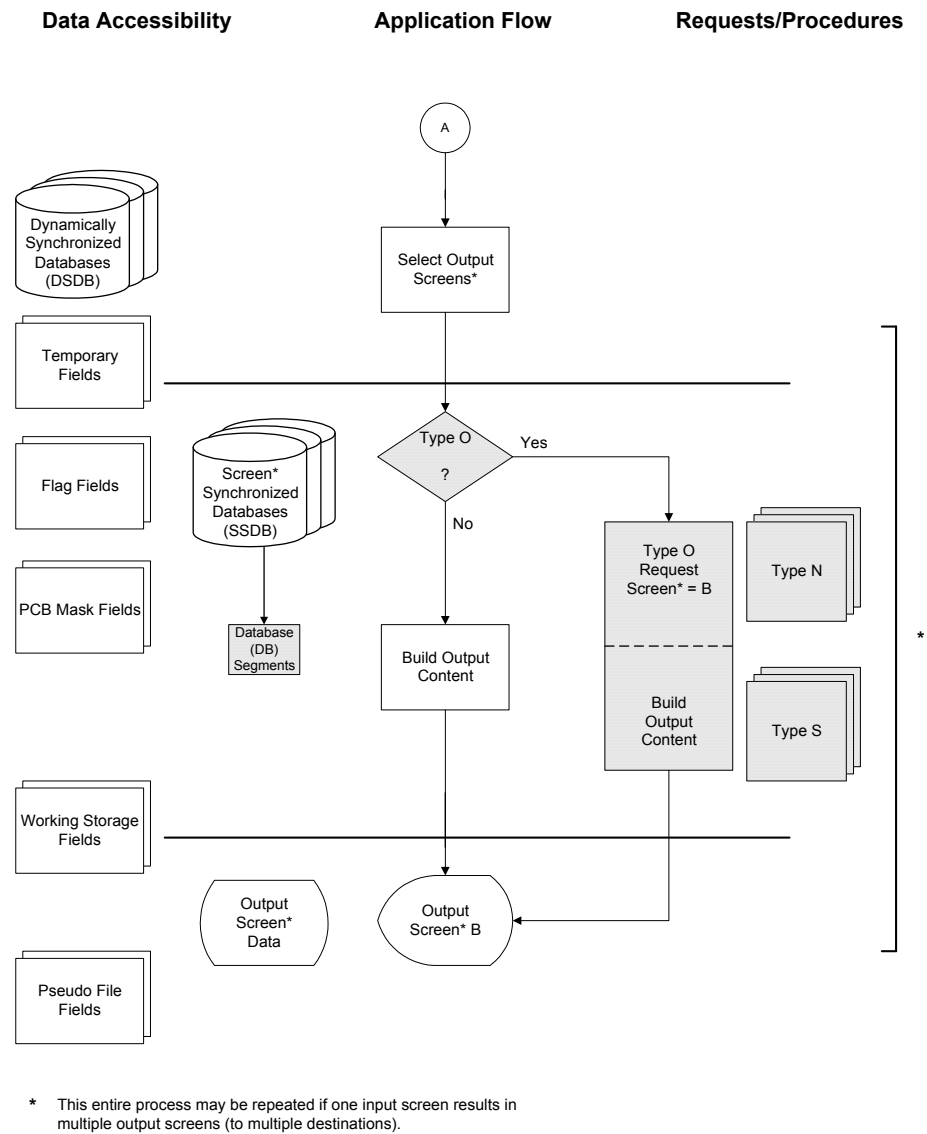


Figure 4-22 The Basic Interaction (Page 2 of 2)

* "Screen" can also be "COMMAREA section" in CICS

Screen/COMMAREA Layout and Screen/COMMAREA Fields

The formatting of screen displays or CICS COMMAREA sections is one of the most important design considerations in the development of an online application. The screen display is the vehicle for the terminal user's access to information in a 3270 Terminal program while the COMMAREA section is the vehicle for the client presentation programs access to information in a CICS COMMAREA program. Specification of the information which should be output to or input from the screen or COMMAREA section becomes one of the major influences within the flow of an application. The screen or COMMAREA section processing definition is used to define the layout of information presented to the user or client presentation program, as well as the initial data validation and editing in your online application.

Important! *Keep in mind throughout the remainder of this chapter that wherever the term "screen" is used, the concept of "COMMAREA section" can be substituted. In some places in the text, the term COMMAREA or COMMAREA section is also included for emphasis or clarification.*

Defining Screen/COMMAREA Layouts

Two statement types define the overall characteristics and layout of the screen. You use the SD statement to name the screen processing definition and provide several characteristics that apply to the layout. You use the SL statement to place literal and data fields on the screen, as well as to specify extended highlighting, color, and programmed symbols.

The SD Statement

The screen definition (SD) statement assigns general characteristics to a screen processing definition.

The Screen Name specification assigns the name under which the screen processing definition is cataloged on the VISION:Transact library. This name is later used in screen control statements that select output displays. This name is repeated on all statements pertaining to this screen processing definition.

Two characters, the field designator and the highlight designator, are the tools you use on screen layout (SL) statements to map fields which are to appear on the screen display and to set the intensity of the screen display literals. The Field Designator and Highlight Designator specifications allow you to override the system defaults for these characters which are set at installation time and are supplied as an underscore (_) and at sign (@), respectively.

The Automatic Format specification directs VISION:Transact to generate an automatic display format for all fields that do not have explicit screen layout (SL) specifications. A C in the Automatic Format entry will cause a columnar display of all fields lacking SL specifications. The *Advantage VISION:Transact Specifications Guide* contains more information about this capability.

If there are more occurrences of the source data base record or segment than will fit on one output screen display, not all data base information will be displayed. Coding an A in Output Control causes your application to output multiple screen displays, as many as are needed to display all available occurrences of the source data. The OC OUTLIM parameter in screen control can provide a limit to the number of screens generated. (See also [Chapter 11, "Advanced Application Flow"](#).) The terminal user advances from one display to the next via the PA1 key or Message Format Service (MFS) operator logical paging commands in IMS TM applications. In CICS applications, Basic Mapping Support (BMS) paging commands allow you to "page through" the set of multiple screen displays.

The Input Transaction Code specification is used to tell VISION:Transact the name of the transaction to which the screen data is input. Therefore, a screen may be used to bridge two or more transactions by having it as an output display from one or more transactions and as input to the transaction named on the screen's SD statement. You could alternatively use the TRANCODE flag to determine the proper transaction to process that screen.

Recall that in the previous section we saw that one method of performing application linking was by having each screen have a particular transaction associated with it. This one-to-one relationship between transaction and screen is specified with this entry on the SD statement.

The Screen Identifier specification is used to name the screen format to MFS or BMS.

For CICS targeted applications, VISION:Transact uses the screen identifier as the MAPSET name and the screen name as the MAP name. If multiple screens are defined in a single VISION:Transact definition run, they will be grouped into a single mapset using the screen identifier entry from the first SD statement in the run.

For IMS TM, VISION:Transact uses the screen identifier as the MOD name and appends an I to form the MID name. The screen identifier is also used for the DIF/DOF names, which are truncated to six characters if necessary, or have an F appended if the screen identifier is six characters or less. Screen identifiers for an IMS TM screen should not be exactly six bytes long since VISION:Transact uses only the first five bytes in order to make DIF/DOF names distinct from the MOD name.

Null Segment Processing is used only with structured screens (explained in [Chapter 6, “Structured Screens/COMMAREA And The Intermediate Structure”](#)). In a structured screen, as in a hierarchically structured record, there is one root segment and multiply occurring dependent segments. If there is no data displayed in a dependent screen segment occurrence when the screen is input, then the generated application will not attempt any processing for that occurrence. This specification allows you to force the application to process those null segments.

The last entry, Device ID, indicates the type of terminal or for IMS TM the group of terminals for which the Screen Definition is being generated. The device name entered here corresponds with the device name in the M5PARM Device Table. If this entry is left blank, the M5PARM default device is used by VISION:Transact. The M5PARM Device Table and the default device are shown on the VISION:Transact signon page. For more details concerning the M5PARM Device Table, refer to the *Advantage VISION:Transact Specifications Guide* and the *Advantage VISION:Transact for CICS Installation Guide*.

If the definition related to this SD statement will be used in a CICS COMMAREA program only, the Device ID entry may contain COMMAREA to indicate that the definition is intended for a COMMAREA program only and that any constraints unique to 3270 Terminal displays do not apply. Furthermore, no CICS BMS mapping statements will be generated for definitions that specify a Device ID of COMMAREA.

Placing Literals and Data Fields in the Screen

The details of a screen display are specified using screen layout (SL) statements. These statements indicate where the literals and data are to be displayed. You can code three types of characters in the text of the SL statement. One is a field designator character (defaults to _), which allocates a position on the display to input or output data. Contiguous field designators (up to 255) compose a screen field. Another character is a highlight designator (defaults to @), which surrounds a literal to indicate that the literal is to be displayed in high intensity. Lastly, literals themselves can be used; they display exactly as coded on the SL statement. The following rules must be observed when you code screen display layouts:

1. Position one line one must contain a blank or the highlight designator.

2. A string of field designators (representing one screen field) must be preceded and followed by either a blank or a highlight designator. This is to allow for the 3270 attribute byte that will be automatically inserted before and after each screen field.
3. Screen fields are laid out with a maximum of 255 field designators per field.
4. One line of the screen display must be reserved for error messages. By default, this is the last line of the display. However, you may place this message line anywhere on the screen by using the MSGLINE flag field as the source of an output screen field. In addition, you can display data in the message line screen field by moving that data into the MSGLINE flag field

For CICS COMMAREA applications, literals defined on SL statements are ignored for the COMMAREA data content and layout. The only function that SL statements serve for CICS COMMAREA applications is to define the length of a field in the COMMAREA by way of the corresponding set of contiguous field designator characters (default to _).

Relating Screen/COMMAREA Field Designators and S0 Statements

Once the screen layout is complete, it is time to assign source and destination fields to each screen field and designate whether it is input, output or both. As you saw in [Chapter 4, “Application Flow”](#), you use the S0 statement to do this. How does the generated application know which S0 statement applies to a given screen field? What happens, moreover, when one input screen field is directed to two or more destination fields in the application? The three factors that affect this process of connecting screen fields and their source and destinations are the position of the screen field in the layout, the order in which the S0 statements are input to VISION:Transact during the screen processing definition run, and the entry in V/S/A (Virtual/Same/After) on the S0 statement.

The V/S/A entry controls whether its S0 statement applies to the next screen field in the display, to the same screen field as the previous S0 statement, or to a virtual screen field. (A virtual screen field is simply a screen field that is not associated with a field designator in the screen layout. See [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#) for more details on virtual screen fields.)

VISION:Transact reads the input S0 statements and assigns them to screen fields from left to right and from top to bottom. The first S0 statement with no entry in V/S/A is assigned to the first screen field, the next S0 statement is assigned to the next screen field, and so on. If no S0 statements have entries in V/S/A, this process continues until the last S0 statement is assigned to the last screen field.

When VISION:Transact finds an S in V/S/A, then this S0 statement is assigned to the same screen field as the previous S0 statement. In this way, you assign multiple destinations to an input screen field. Another use for the S in V/S/A is to cause the screen field data to be output from one application field and then input to a different application field when the screen is subsequently input. [Figure 5-1](#) shows an example of this technique.

Notice that the first S0 statement assigns the source for the screen field data (Input/Output entry is O). The next S0 statement, with an entry of S in V/S/A, assigns the destination of the screen field data (Input/Output entry is I). When this screen is selected for output, the screen field data will come from the field CUSTNO in the data base whose qualifier is a blank. When the screen is input, the data in the screen field will be directed to the field CUSTOMER in the data base whose qualifier is 1.

When VISION:Transact finds an entry of V in V/S/A, a virtual screen field is created whose length matches the source/destination field. (If no source/destination field is specified, you must assign a length to this screen field using the LN Edit/Validation operator; see [Assigning a Length to a Virtual Screen Field on page 5-12](#)). No attempt is made to assign a virtual screen field to a string of field designators. If the next S0 statement contains an S in V/S/A, it refers to the just-created virtual screen field; if it contains a V it refers to a new virtual screen field; and if it contains a blank, it refers to the next designated screen field.

To use the entry of A in V/S/A with structured screens, see [Chapter 6, "Structured Screens/COMMAREA And The Intermediate Structure"](#).

```
-----
                                V           I R   C E P M N       A
SCREEN  STMT SEG S  SOURCE/DEST  / Q H U R R O U       C
NAME   TYPE NO. A  QLF FLD NAME O D I R R T D M       T
-----
(SCREENA ,S0) (01)           (CUSTNO ) (O)
(SCREENA ,S0) (01) (S) (1,CUSTOMER) (I)                (M)
```

Figure 5-1 Using S in V/S/A to Assign Different Source and Destination Fields to the Same Screen Field

Output Screen/COMMAREA Facilities

Several specifications on the S0 statement can affect the screen layout and screen field data before the screen is output to the terminal. This section discusses screen field attributes, output screen field editing and field descriptors for automatically formatted screens.

Screen Field Attributes

You use specifications on the S0 statement to set the normal screen field display attributes available on almost all 3270 terminals: normal, high, or dark intensity, cursor positioning, protected, and numeric lock.

Note: Attribute specifications are ignored for CICS COMMAREA applications.

You use the Highlighting specification to control the intensity with which the screen field is displayed. Different entries allow you to override the normal intensity to display the field with high intensity or to not display the screen field at all (for a password, for example). When you set the high intensity attribute for an input screen field, the characters keyed into the field are displayed in high intensity. Some 3270 terminals now allow additional highlighting features like color, underlining, and blinking. To set these attributes in a VISION:Transact generated application, see the next subsection, Extended Attributes.

Use the Cursor specification to place the cursor on a specific screen field when the screen is output.

The Protected specification allows you to inhibit keyboard input into a displayed screen field. Because output-only screen fields are automatically protected in VISION:Transact generated applications, this specification only applies to input screen fields. When the terminal user tries to enter data into a protected field, the keyboard locks and must be reset before any more input can occur.

Use the Numeric specification to allow only numeric characters to be entered into a screen field. A complete description of this specification is in [Input Screen/COMMAREA Facilities on page 5-11](#).

[Figure 5-2](#) shows examples of all of these attributes.

```
-----
              V              I R   C E P M N
SCREEN  STMT SEG S  SOURCE/DEST  / Q H U R R O U
      NAME  TYPE NO. A  QLF FLD NAME O D I R R T D M
-----
(SCREENA ,S0) (01)      (CUSTNO ) (B) (H)   (Y)
(SCREENA ,S0) (01)      (ORDERNO ) (I,Y) (Y)
(SCREENA ,S0) (01)      (ITEMNO ) (I,Y)
(SCREENA ,S0) (01)      (ITMQTYOR) (I,Y)      (Y)
```

Figure 5-2 Examples of Screen Field Attributes

Extended Attributes – Highlighting, Color, and Programmed Symbols

VISION:Transact application programmers can easily make use of the color, extended highlighting, and programmed symbol capabilities of some 3270 terminals. Color attributes allow screen fields or literals to be displayed in different colors. Extended highlighting allows a field or literal to be displayed underlined, blinking, or in reverse video (that is, characters are displayed as black on a white background, instead of the usual white on a black background). Programmed symbols are installation or application specific sets of characters that require a special set of codes to be displayed (for example, non-English alphabets).

The facility for specifying these extended attributes is the XATTR specification of the SL statement along with an identifying code in the text of the screen layout. The method of specifying extended attributes is the same for literals and screen fields.

To specify an extended attribute for a screen field or literal:

- Include an SL statement with the same line number and continuation as the SL statement containing the field or literal to which the attribute applies.
- Enter the code for the attribute type you want in the XATTR specification of the new SL statement (C for color, H for extended highlighting, or P for programmed symbols).
- Enter the corresponding code for the specific attribute in the same positions in the text of the new SL statement as the literal or field to which it applies (see the *Advantage VISION:Transact Reference Summary* for a complete list of codes).

[Figure 5-3](#) shows an example of the specifications needed for each of the extended attributes. Notice that more than one extended attribute may be specified for a field or literal.

SCREEN	SMT	CAT	GLOSS	FIELD	HILITE	AUTO	OUTPUT	TRANSACTION	SCREEN	PROC	TERMINAL
NAME	TYPE	ACTN	TYPE	DESIG	DESIG	FORMAT	CONTROL	CODE	ID	NULL	DEVICE
(SCRN	)	(SD)	(R)		(-)						
SCREEN	SMT	LINE	CONT-	EXTN	+...1....+...2....+...3....+...4....+...5....+..	<-TEXT (1-57) -					
NAME	TYPE	NO.	INUE	ATTR	..6....+...7....+...8....+...9....+...0....+...1....	<-TEXT (58-114)					
(SCRN	,SL)	(001)	(0)	(	THIS TEXT WILL DISPLAY RED	THIS WILL BE YELLOW	)				
(SCRN	,SL)	(001)	(0)	(C)	(RRRRRRRRRRRRRRRRRRRRRRR	YYYYYYYYYYYYYYY	)				
(SCRN	,SL)	(002)	(0)								
(SCRN	,SL)	(003)	(0)	(	ENTER PASSWORD -----		)				
(SCRN	,SL)	(003)	(0)	(H)	(RRRRRRRRRRRR		)				
(SCRN	,SL)	(004)	(0)								
(SCRN	,SL)	(005)	(0)	(	A RED EGCS FIELD -----		)				
(SCRN	,SL)	(005)	(0)	(C)	(RRRRRRRRRRRRRR RRRR		)				
(SCRN	,SL)	(005)	(0)	(P)	(EEEEE		)				

Figure 5-3 Specifying Color, Extended Highlighting, and Programmed Symbols

In the example, literals on line one will be displayed red and yellow. Literals on line three will display in reverse video. The field and literals on line five will both display in red. In addition, the field will be composed of extended graphics characters (see IBM 3270 programming manuals for details on extended graphics).

Contiguous attribute codes in the text of the SL statement may not be mixed. For example, a color attribute specification of BBBYBB is illegal.

For IMS TM only, to use a screen defined with extended attributes on terminals that do not support the attributes as well as terminals that do support them, the Device ID on the SD statement must specify a device group which contains a definition of each terminal type and its supported attributes. Refer to the M5PARM Device Table in the *VISION:Transact for CICS Installation Guide* for more information.

Editing Screen/COMMAREA Fields

Often, the way data is stored by the computer and the way the terminal user wants to view that data are different. For this reason, edit characters can be inserted into output screen fields. For example, a telephone number is always (in the United States, at least) a number. A VISION:Transact generated application will display those numbers in a screen field. However, a string of 10 numbers, 8187161616,

may be less clear than the standard telephone number format of (818) 716-1616. You use the ED Edit/Validation operator on an Sn statement to specify a pattern of characters that will be inserted into the output screen field data. (In [Input Screen/COMMAREA Facilities on page 5-11](#), you will see how to use the ED operator to specify de-editing of input screen fields.)

When you enter ED in Edit/Validation Operator on an Sn statement, Operand B contains the edit mask. The edit mask consists of edit insertion characters and markers to indicate where the data characters are supposed to appear in the edited field. The system delimiter is the marker character. Any characters other than the marker character, including leading and imbedded blanks, are considered insertion characters. The edit pattern cannot exceed the size of the Operand B field (24 characters), and must not exceed the size of the screen field.

[Figure 5-4](#) shows two examples of editing screen fields, one whose source field contains character-type data, and another whose source field contains numeric data.

The S0 statement for the first screen field in [Figure 5-4](#) specifies data base field NAME as its source. Using the file definition as a reference, you know that this field is 10 characters long. In addition, you know that the data in this field is stored in the format FLLLLLLLLL, where F represents the first initial of the persons first name and the Ls represent the last name. You want the name to print on the terminal in the format F. LLLLLLLLLL. Notice the edit mask in this case. It consists of one marker , followed by a period and a space (the two insertion characters), followed by nine more markers, for a total of twelve characters. The corresponding screen field must therefore have twelve field designators.

When the source field is a character type field, editing is accomplished as follows:

1. The edit mask is placed in the screen field.
2. The leftmost marker character is replaced by the leftmost character of the source field, the next marker character is replaced by the next character of the source field, and so on.
3. If there are more source field characters than marker characters, then editing stops when the last marker character is replaced. The source field is truncated on the right.
4. If there are more marker characters than source field characters, then remaining markers are replaced by blanks.

If the source field is a numeric field, it is first converted to a right justified character field whose length is the same as the number of marker characters in the mask. Leading spaces are filled with zeroes. This character field is then edited using the edit mask. A numeric source field may not contain any decimal places. In addition, a maximum of 15 markers are allowed in the edit mask when the source field for output is numeric.

VISION:Transact can format fields in a display for you if requested on the SD statement (Automatic Format entry). In this case, VISION:Transact also automatically provides the literals to describe the data in these fields. These literals are generally the name of the source or destination field listed on the first S0 statement which is assigned to the displayed field. You can specify the literals

you wish displayed with the fields that your S0 statements reference. Do this by using a DS edit/validation operator and providing the description as a character literal in the B operand. Use the C qualifier for the edit/validation field associated with the DS operator. For more details about automatically formatted displays refer to the *Advantage VISION:Transact Reference Summary*.

Input Screen/COMMAREA Facilities

Several S0 and Sn statement specifications influence the way the screen fields themselves are processed after the screen is input to the application. This section discusses the following topics with respect to input screen fields: names, length (of virtual screen fields), input required, de-editing, validation, multiple edit/validation, and default values.

Screen/COMMAREA Field Names

During the course of the interaction, you may need to refer to the data that was in the screen fields when the terminal user pressed the ENTER or program function key. VISION:Transact generated applications store this data in an area of memory called an input screen buffer. The input screen buffer is available to the interaction until the application performs the physical update to the data bases.

To reference the data in a screen field (in the screen processing definition or in a request/procedure, discussed in [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#)), use an S qualifier and the name of the screen field.

Screen fields are assigned the name of their destination field. Often this is not sufficient. For example, screen fields do not always require a destination, leaving these screen fields without a name. Multiple screen fields with the same destination cannot be distinguished on the basis of destination field name alone. Assigning a unique name to a screen field can prevent confusion.

You can assign a name to a screen field using the NM edit/validation operator with the name you are assigning to it specified as the Edit/Validation field. Consider the following statements:

```
-----
              V              I R   C E P M N   A L C
SCREEN  STMT SEG S  SOURCE/DEST  / Q H U R R O U   C L O E/V .....OPERAND B.....
NAME   TYPE NO. A  QLF FLD NAME O D I R R T D M   T V N O P  QLF FIELD NAME OR CONSTANT
-----
(CRIMENU4,S0) ( 1)      (CUSTNO ) (I,Y)              (V)
(CRIMENU4,S0) ( 1) (S) (1,ITEMNO ) (I,Y)              (M)  (NM) (ITEMNO )
```

Figure 5-5 Using the NM Operator to Assign a Name to a Screen Field

In [Figure 5-5](#), the same screen field serves as input to fields in two different data bases. By default, it receives the screen field name of the first destination field name entered (CUSTNO). By using the NM operator, we can rename the screen field to ITEMNO allowing us to reference the screen field content as S,ITEMNO during request/procedure processing (see [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#)).

Assigning a Length to a Virtual Screen Field

Screen fields receive their length from the associated field designators on the SL statement. However, a virtual screen field is a screen field that has no field designators associated with it. It exists only in the input and/or output screen buffer. If the virtual screen field has a destination field associated with it, then the length of the destination field becomes the length of the virtual screen field. If it has no destination, then you must assign a length to a virtual screen field with the LN edit/validation operator.

You can use the LN operator on either the S0 or Sn statement.

Refer to [Errors in Virtual Screen Fields on page 5-22](#) for more information regarding virtual screen fields.

Defaults for Screen/COMMAREA Field Data

Using an S0 or Sn statement, you can supply data for a screen field that the user left blank. This default technique is useful to facilitate applications which are semi-automatic to the user. The user can override the defaults you supply by keying the desired information into the screen display.

Specify defaults by using a DF Edit/Validation operator and providing the default as the Edit/Validation field with the appropriate qualifier. The default value is moved into the screen field during input validation processing if the user leaves the corresponding screen field blank. Default Edit/Validation is done prior to any other validation.

[Figure 5-6](#) uses the flag field TODAY, which contains the operating system date, as the default value for ORDRDATE.

		V	I R C E P M N				A L C		
SCREEN	STMT	SEG	S	SOURCE/DEST	/	Q H U R R O U	C L O E / V	OPERAND B.....	
NAME	TYPE	NO.	A	QLF	FLD	NAME	O D I R R T D M	T	V N O P QLF FIELD NAME OR CONSTANT

(ORDADD4 ,S0)	(1)			(ORDRDATE)	(B)		(P)	(DF)	(F,TODAY)
(ORDADD4 ,S0)	(1)			(ORDERNO)	(B)		(U)		

Figure 5-6 Using the DF Operator to Supply a Default Value for a Field

Since the screen fields receiving these default values are character string fields, the default values themselves should be character strings. This is true even if the default value is a number. Specification of the default value as a character string will prevent unnecessary data conversions by your application.

A default value specified for a screen field is not applied if the Sn set specifying the default has an X action selected for it, or if the destination segment has been assigned a segment action of X. An action of N, on the other hand, does not preclude the application of a default value to a field, though it halts all data base actions against that field.

Input Required in a Screen/COMMAREA Field

An entry of Y in the Input Required specification requires the screen field to be non-blank when the screen is input to the application. If the screen field is blank, the application will redisplay the screen with the cursor on the screen field and an error message on the message line. The error message for this condition can be the standard VISION:Transact error message or your application-specific message. See [Error Processing for the Input Screen/COMMAREA on page 5-19](#) for details on how to override standard VISION:Transact error messages.

De-Editing Screen/COMMAREA Fields

The ED Edit operator that you use to specify insertion of edit characters when the screen is output also specifies the subsequent de-editing to be performed on those input screen fields when the screen is input to the application.

De-editing is accomplished by removing the edit insertion characters and compressing the remaining data characters so that the data is left-justified in the screen field. A screen field is de-edited only if it contains all of its edit insertion characters and they are in the correct positions in the field.

If the EN operator is used on an input screen field whose destination is a character field, then

- if the screen field is blank, nothing is done
- if the screen field is not blank, its contents are right-justified, and leading spaces are filled with zeroes.

One use for the latter technique is to right justify a screen field whose destination is a character-type key field that contains numbers in ascending sequence. For example, if the key field for the record you want to display is 000025, and the EN operator is used for this screen field, the terminal user need only key two characters (25) into the screen field, rather than six (000025).

Validating Input Screen/COMMAREA Field Data

You can specify validation of input screen field data in an S0/Sn set. Two types of validation are allowed here.

- Automatic numeric validation, where VISION:Transact sets the "numeric" field attribute on terminals equipped with the numeric lock feature, and/or performs its own numeric validation, and
- Logical, application-specific validation, where you use the VISION:Transact Edit/Validation operators to compare data entered on the screen to a literal, another field or a named pattern.

You specify automatic numeric validation by an entry in the Numeric specification of the S0 statement. Different entries allow you to select either or both of the following results.

1. The generated application sets the "numeric" field attribute on those 3270 terminals that support the numeric lock feature. The terminal operator may enter only digits and the characters "-" and "." in that field. This attribute has no effect if the terminal does not support the numeric lock feature.
2. The generated application performs its own validation to see if the field is a decimal number or a floating point number.
A field is considered a decimal number if:
 - The field contains only blanks, digits, and the characters "-", ",", and "." for the sign, grouping character, and decimal point, respectively.

Note: The decimal point and grouping character may be changed during system generation. See the *VISION:Transact for CICS Installation Guide* for details.

- The field contains no imbedded blanks.
- The field contains no more than 15 digits.
- If the field contains a minus sign (-), then there must be only one and it must be the first non-blank character.
- If the field contains a decimal point (.), then there must only be one.

Note: The decimal point and grouping character may be changed during system generation. See the *VISION:Transact for CICS Installation Guide* for details.

- If the field contains grouping characters, then they must be separated by 3 digits.

Note: The decimal point and grouping character may be changed during system generation. See the *VISION:Transact for CICS Installation Guide* for details.

A field is considered to be a floating point number if:

- The field is a decimal number, or
- The field is a decimal number followed by an exponent (the character "E" followed by a one or two digit number and, optionally, a sign, "+" or "-").

If this validation fails, the screen is redisplayed with a standard VISION:Transact error message and the cursor at the beginning of the field in error. See [Error Processing for the Input Screen/COMMAREA on page 5-19](#) for details on overriding standard VISION:Transact error messages.

[Figure 5-7](#) shows several examples of automatic numeric validation.

V				I R C E P M N								A			
SCREEN	STMT	SEG	S	SOURCE/DEST		/	Q	H	U	R	O	U	C		
NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D	I	R	R	T	D	M	T

(NUMERIC ,S0) (1)				(T,FIELDA) (I)								(Y)	(R)		
(NUMERIC ,S0) (1)				(T,FIELDB) (I)								(A)	(R)		
(NUMERIC ,S0) (1) (V)				(T,FIELDC) (I)								(V)	(R)		

Figure 5-7 Examples of Automatic Numeric Validation

A Y Numeric specification, as on FIELDA, means that the numeric field attribute is set (if the terminal is so equipped) through MFS or BMS. This allows a terminal operator to enter only numeric data (as determined by the hardware) into FIELDA. In addition, at input, automatic numeric validation is performed by VISION:Transact for that field. An A Numeric specification for FIELDB means that the numeric field attribute for the terminal is set but no automatic numeric validation is performed by VISION:Transact. FIELDC is a virtual input screen

field (see [Chapter 9, “Data Manipulation/Field Handling”](#)) and does not display on the screen; its value is assigned in a type 1 request (see [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#)). A V Numeric specification does not cause the numeric field attribute to be set but automatic floating point numeric validation is performed by VISION:Transact. A Y or V specification causes automatic numeric validation of the input screen field by VISION:Transact. An A specification simply sets the numeric field attribute for the terminal. (For details on additional valid entries, see the *Advantage VISION:Transact Reference Summary*.)

You specify logical, application-specific validation by using the following Edit/Validation operators:

- EQ for equal.
- NE for not equal.
- LT for less than.
- GT for greater than.
- LE for less than or equal.
- GE for greater than or equal.

These operators compare the data entered in the displayed screen field to the data contained in the field, constant or pattern specified in Operand B. If the result of the comparison is false, the application redisplay the input screen with a diagnostic message and by default highlights the field in error. (See [Using the MS Operator to Supply a Message for Validation Error on page 5-20](#) for details on user written error messages.)

Since validation occurs during input processing, S0/Sn sets specified as output only may not specify validation criteria.

One comparison is often insufficient to validate input data. Therefore you may code as many Sn statements, each with an edit/validation operator, as necessary to perform the required input validation. You may code multiple Sn statements with the same number (S1-S9) and any combination of S1-S9 in ascending order. Each Sn statement is assigned to the same screen field as the previous S0 statement. Multiple edit/validation operators are treated as a single logical expression, and may be connected with an "and" or "or" by specifying an "A" or "O" connector on the Sn statement(s). The Logic Level specification allows you to specify the sequence in which the "and" or "or" conditions are valuated. The logical result of the connected operators is evaluated to determine the validity of the data entered in the screen field to which the S0 and Sn statements refer.

Note: You may partial field screen fields on Sn statements (n=1-9) during validation.

You can specify "or" and "and" connectors on different Sn statements for complex validation. Suppose the application requires a customer number equal to 10000 or between 1000 and 5000. You could code the following:

```

-----
              V              I R   C E P M N   A   L   C
SCREEN  STMT SEG S  SOURCE/DEST / Q H U R R O U   C   L O E/V .....OPERAND B.....
NAME   TYPE NO. A  QLF FLD NAME O D I R R T D M   T   V N O P   QLF FIELD NAME OR CONSTANT
-----
(EXAMPLE ,S0) ( 1)   (N,CUSTNO ) (B)                (U)   (EQ) (C,10000                )
(EXAMPLE ,S1)                (O,GT) (C,1000                )
(EXAMPLE ,S1)                (A,LT) (C,5000                )

```

Figure 5-8 Connectors for Complex Validation

Validation statements connected with an "A" are evaluated before those connected with an "O" unless this sequence is changed by using the Logic Level specification. The following steps summarize the validation statements shown above:

1. The data entered in the field displayed on the screen is compared with the character string 10000.
2. If the data equals 10000, the data is valid and other input data can be validated.
3. If the data doesn't equal 10000, the data is tested to determine whether it is greater than 1000 and less than 5000.
4. If the data is between 1000 and 5000, the data is valid and other input data can be validated.
5. If the data isn't between 1000 and 5000, the data is invalid and the application redisplay the input screen format with a VISION:Transact diagnostic message on the error message line. An MS operator can follow the validation to specify your own message. See [Using the MS Operator to Supply a Message for Validation Error on page 5-20](#) for details.

```

-----
              V              I R   C E P M N   A   L   C
SCREEN  STMT SEG S  SOURCE/DEST / Q H U R R O U   C   L O E/V .....OPERAND B.....
NAME   TYPE NO. A  QLF FLD NAME O D I R R T D M   T   V N O P   QLF FIELD NAME OR CONSTANT
-----
(EXAMPLE ,S0) ( 1)   (T,SELECT ) (I,Y)                (P)   (EQ) (C,A                )
(EXAMPLE ,S1)                (O,EQ) (C,M                )
(EXAMPLE ,S2)                (O,EQ) (C,V                )
(EXAMPLE ,S3)                (MS) (C,INVALID SELECTION  )

```

Figure 5-9 Required Entry and the MS Operator

In [Figure 5-9](#) above, the S0 statement directs the data to its destination using the P action code. Before this is done, however, two separate validation steps are performed. First, because input required is specified, the application confirms that there is some non-blank entry in the screen field. If not, the application receives an automatic error message from VISION:Transact. Second, because of the EQ edit validation operators, the application checks the value of the screen field. If the entry is A, M, or V, the screen field is valid and the application continues. If there is any other entry in the screen field, it is in error and the screen is redisplayed with the screen field highlighted. Because you included an MS operator immediately following the validation operators, the character string in Operand B, INVALID SELECTION, is displayed on the error message line when the screen is re-displayed.

The two previous examples showed validation against character string constants. Screen field data may also be validated against numeric constants, other fields or named patterns.

To use a numeric constant to validate a screen field, write the numeric value in Operand B with a qualifier of D. To use a data base, temporary, working storage, pseudo file or PCB mask field as a validation field, simply write the field name with the proper qualifier in Operand B after the validation operator. (See [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#) for processing considerations when using validation fields in requests or procedures.)

Validation of input screen data does not take place if the Sn set specifying the validation has an X action selected for it. If the X, D, or M action is specified for a segment, field actions and validation are not performed for that segment. An action of N, on the other hand, does not prevent the validation of a field, though it performs no actions.

You can use a named pattern to validate that the screen field data conforms to a given date or numeric format. For example, the pattern name INTEGER is used to make sure that the screen field data is an integer, and the pattern name MMDDYY is used to confirm that the data is in the given date format. To use a named pattern as a validation operand, write the validation operator EQ or NE followed by pattern name in Operand B. [Figure 5-10](#) shows an example of using named patterns on S0 and Sn statements.

		V	I R C E P M N				A L C		
SCREEN	STMT	SEG	S	SOURCE/DEST			/ Q H U R R O U	C L O E/V	OPERAND B.....
NAME	TYPE	NO.	A	QLF	FLD	NAME	O D I R R T D M	T V N O P	QLF FIELD NAME OR CONSTANT

(EXAMPLE ,S0)	(01)			(ORDRDATE)	(B)			(P)	(EQ) (P, .MMDDYY)
(EXAMPLE ,S1)									(MS) (C,WRONG DATE FORMAT)
(EXAMPLE ,S0)	(01)			(ITMQTYOR)	(B)			(P)	(EQ) (P, .INTEGER)
(EXAMPLE ,S1)									(MS) (C,MUST BE AN INTEGER)

Figure 5-10 Using Named Patterns

Notice that the first character in Operand B must be a period (.), and that the qualifier is a P. Refer to the *Advantage VISION:Transact Reference Summary* for a complete list of named patterns. [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#) shows how to use screen requests/procedures to perform more complicated validation.

In the example, if the screen field whose destination is data base field ORDRDATE is not in the format MMDDYY, the screen is redisplayed with message WRONG DATE FORMAT. If the screen field whose destination is data base field ITMQTYOR is not an integer, then the screen is redisplayed with message MUST BE AN INTEGER.

Error Processing for the Input Screen/COMMAREA

You have seen several times in this section and the previous one that the input screen is redisplayed when an error is encountered. These errors are all connected with an individual screen field. For example, if a screen field fails a validation or if a record or segment occurrence with a key field value equal to the screen field is not found, then the screen field is said to be in error. The VISION:Transact generated application will attempt to show the terminal user which screen field is in error when the screen is redisplayed by highlighting the field in error, placing the cursor at the start of the field and displaying an error message describing the error.

There are several considerations for the VISION:Transact application programmer when this occurs: what screen field data should be displayed, should the field be highlighted, what error message should be displayed, what happens if the field in error is a virtual screen field, and what happens when the error is found by a screen request or procedure, independent of the coding on the Sn set.

Displaying Modified Screen Field Data

The Display Modified specification on the S0 statement controls what data is to be displayed in that screen field when the screen is redisplayed on any error condition. If the entry is blank, the contents of the field when the user entered the screen are redisplayed. If this entry is a Y and the contents of the field have been modified, the redisplayed field will contain the modified contents. The contents of a screen field can be modified by a screen request or procedure, by de-editing, or by the default (DF) operator.

Highlighting the Field in Error

The Error Handling specification controls whether or not the screen field will be highlighted during input screen redisplay when the field is in error. By default, the field is highlighted.

Using the MS Operator to Supply a Message for Validation Error

You can use one or multiple successive MS operators on Sn statements following validation operations to override the standard error message issued by VISION:Transact when input screen field validation fails. The text (or value of fields) in operand B will be concatenated into a single error message.

Using the EC and ET Operators to Override VISION:Transact Standard Error Messages

While you use the MS operator to create error messages that are displayed when an input screen field fails validation which is specific to your application, there are other error conditions (determined by VISION:Transact) under which you may want to override the standard VISION:Transact error message with your own text. To do this, VISION:Transact provides two operators for you to use on Sn statements: EC to identify an error condition and ET to provide the text that is displayed instead of the standard VISION:Transact error message text.

The following represents a partial list of the error conditions for which you can provide your own error messages using the EC and ET operators (the complete list is in the *Advantage VISION:Transact Reference Summary*):

- Input is required but the screen field is blank (F01).
- Numeric validation is specified on the S0 statement but the screen field is not numeric (F04).
- The record or segment occurrence is not on the data base (S01).
- The record or segment occurrence to be inserted is already on the data base (S02).

The three-character code following each condition is the code you use with the EC operator to identify the condition for which you are supplying text. To use these operators, write an EC operator with an error code in the Operand B field. Follow the EC operator with one or more ET operators. Operand B of the ET operator contains the error message text. If no EC/ET combination exists for an error condition, then the standard VISION:Transact error message will be issued.

[Figure 5-11](#) shows several examples of EC/ET combinations.

V		I R C E P M N		A L C	
SCREEN	STMT SEG S	SOURCE/DEST	/ Q H U R R O U	C L O E/V	OPERAND B.....
NAME	TYPE NO. A	QLF FLD NAME	O D I R R T D M	T V N O P	QLF FIELD NAME OR CONSTANT
(SELCTCST,S0) (1)		(I,Y)		(N) (NM)	(SELECTY)
(SELCTCST,S1)				(EQ) (C,M	)
(SELCTCST,S1)				(O,EQ) (C,I	)
(SELCTCST,S1)				(MS) (C,MUST BE M OR I ONLY	)
(SELCTCST,S1)				(EC) (F01	)
(SELCTCST,S1)				(ET) (YOU MUST ENTER SOMETHING)	)
(SELCTCST,S1)				(ET) (IN THE SELECT FIELD	)
(SELCTCST,S0) (1)		(CUSTNO) (B)		(X)	
(SELCTCST,S1)				(M) (IF) (S,SELECT) (EQ) (C,M	)
(SELCTCST,S1)				(I) (IF) (S,SELECT) (EQ) (C,I	)
(SELCTCST,S1)				(EC) (S01	)
(SELCTCST,S1)				(ET) (THE CUSTOMER NUMBER IS N)	)
(SELCTCST,S1)				(ET) (OT ON THE DATA BASE	)
(SELCTCST,S1)				(EC) (S02	)
(SELCTCST,S1)				(ET) (THE CUSTOMER NUMBER IS A)	)
(SELCTCST,S1)				(ET) (LREADY ON THE DATA BASE	)

Figure 5-11 Using the EC and ET Operators

In the example, three EC/ET sets are specified for two screen fields. The first screen field is an input only field for which no action is specified. It is used to select an action to perform for the second screen field. You use the NM operator to name the screen field SELECT, the two EQ operators to validate the contents of the field, the MS operator to redisplay the screen when the validation fails, and the EC/ET set to override the standard VISION:Transact error message when the field is blank (notice the Y in Input Required).

The contents of the second screen field are used to either match on, or insert, a customer number in the CUSTOMER data base, depending on the value in screen field SELECT. The first EC/ET set (Error Code S01) will override the standard VISION:Transact error message when the M action is chosen and the record or segment occurrence cannot be found. The second EC/ET set (Error Code S02) will override the VISION:Transact message when the I action code is selected and the record or segment occurrence is already on the data base.

Errors in Virtual Screen Fields

It is possible for a virtual input screen field to be in error. For example, suppose you are using a virtual input screen field to match a data base record or segment occurrence. Further, the virtual screen field value is made up of two separate screen fields. If that record or occurrence is not found, an error has occurred, and the application will try to inform the terminal user which screen field was in error. However, in this case, a virtual screen field is in error. Because virtual screen fields do not appear on the screen, the application has no way of telling you which screen field is in error. For this reason, you can use the SE (set error) operator to associate an error in a virtual screen field with the error processing for a displayed screen field.

You enter the SE operator on an Sn statement associated with a virtual screen field. Operand B then names the non-virtual screen field whose error processing will be invoked if the virtual screen field is found to be in error.

Screen/COMMAREA Field Error Processing in an Input Screen Request/Procedure

When you use a request or procedure to procedurally validate screen field data, you have the ability to cause the input screen to be redisplayed. However, unlike error processing that is specified in an Sn set, error processing initiated from a screen request or procedure is not associated with a particular screen field. Therefore, the application cannot automatically show the terminal user what screen field is in error. You use the SE operator in a screen request or the SET command in a procedure to cause the error processing for a particular screen field to be initiated when the subsequent MS operator (request) or MSG command (procedure) is issued in the screen request or procedure. For more details on this capability, see [Request Operations and Procedure Commands on page 7-4](#).

Introduction to Structured Screens/COMMAREA

Until now our discussion of screen processing has made little reference to the display of repetitive information such as that contained in lower level data base segments or multiple records. VISION:Transact facilitates the processing of such data. The structured screen (SS) statement on the Screen Processing Definition worksheet allows you to map the structure of your data base on the screen display. The SS statement does this by allowing you to segment your display to match data base processing.

The SS statement allows you to map a structure on your display similar to a single path of a data base. This is done using a unique segment number (which defaults to 1 for root information or a single level screen format) for each level of hierarchical subordination in your display. Specify the number of occurrences of that segment type that you have formatted in your SL statements.

Often you will want to match your data base segments one-to-one with your screen segments. However it is possible to combine information from multiple data base segments into one screen segment. [Chapter 6, "Structured Screens/COMMAREA And The Intermediate Structure"](#) explains in detail the processing of segmented screen displays.

Screen/COMMAREA Layout and Screen/COMMAREA Fields Summary

[Figure 5-12](#) highlights the screen layout and validation components of a VISION:Transact generated application.

You specify the layout of screens and extended attributes with SL statements in the screen processing definition. You specify the editing and validation of screen fields with S0/Sn statements. You supply a screen's structure by the use of SS statements. A single screen definition may contain specifications for both output processing and input processing. The VISION:Transact generated application builds the output display based on your specifications of the layout and structure of the screen.

In [Figure 5-12](#) the shaded symbols to the left of the application flow represent the screen and data base segments that make up the intermediate structure, which is discussed in detail in [Chapter 6, "Structured Screens/COMMAREA And The Intermediate Structure"](#). The shaded symbols to the right of the application flow represent requests or procedures, discussed in [Chapter 7, "Requests/Procedures and the Intermediate Structure"](#).



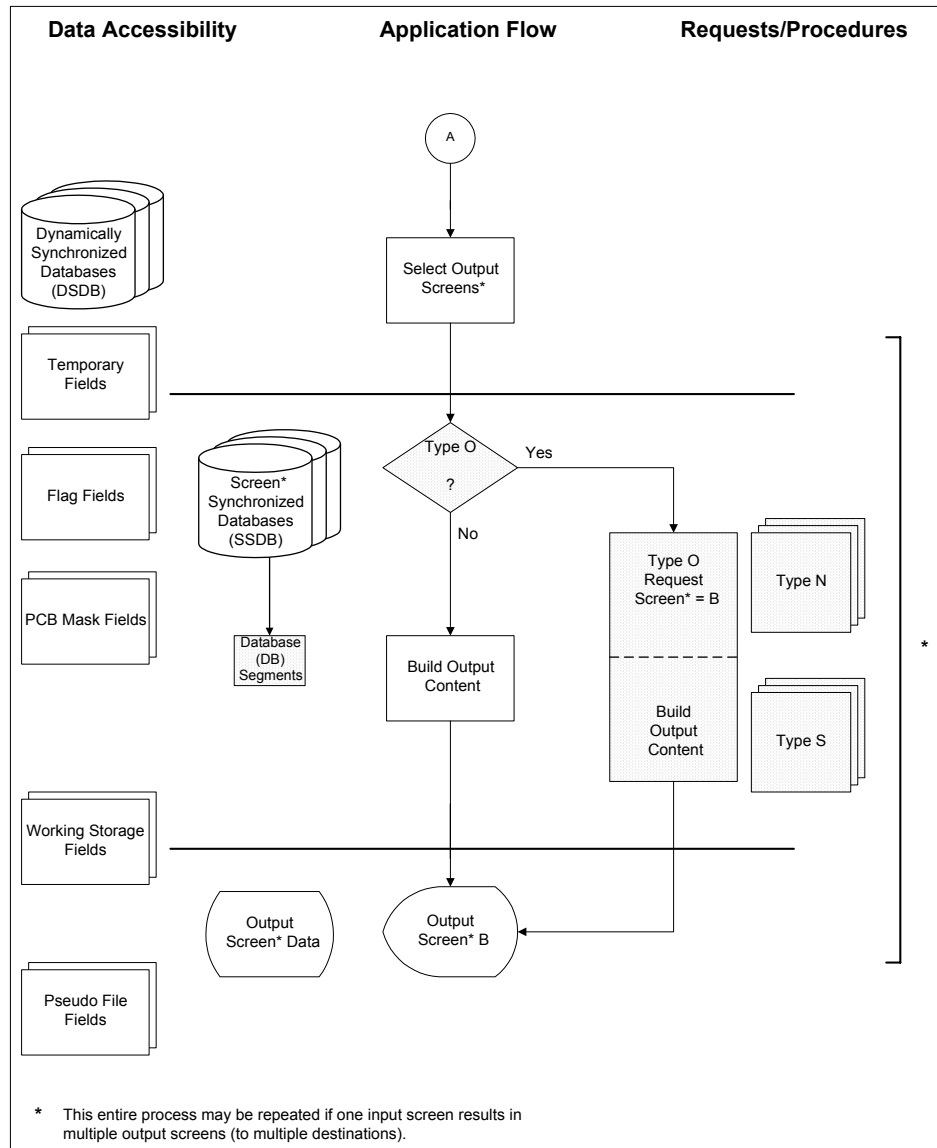


Figure 5-12 The Basic Interaction (Page 2 of 2)

* "Screen" can also be "COMMAREA section" in CICS

Transaction Control Fields in a COMMAREA

The first section of a COMMAREA contains a group of fields call the TRANSACTION-CONTROL field. This field over-defines the MSGLINE field and contains detail fields that are examined on input and used as follows:

The ATTNID field is used to set the ATTNID flag field and simulate the pressing of various 3270 keys. Values that are recognized and processed appropriately are ENTER, CLEAR PFnnn, AND PAnnn. If the field contains blanks, ENTER is assumed. The appropriate setting of this field can be used to simulate the pressing of specific 3270 keys when an application was originally designed for use with 3270 terminals or is intended to be used as both a 3270 Terminal program and a COMMAREA program. Note that when the ATTNID field value is set to CLEAR, no application program processing will occur, only the cleanup of any CICS Temporary Storage queues left over from previous interactions of a pseudo-conversation (see discussion below). This action is identical to the action taken when the CLEAR key is pressed after an interaction with an Advantage VISION:Transact Terminal program.

The CONVERSE field is used to indicate whether the COMMAREA program is to operate in a pseudo-conversation mode or not. When the field contains a value of YES, pseudo-conversational mode is assumed. Any other value will indicate that no pseudo-conversational mode is requested. Pseudo-conversational mode is required for an Advantage VISION:Transact application program when the application design requires that designated Working Storage and Pseudo File data be saved between invocations of the application program that processes the pseudo-conversation. Although one would typically not design a COMMAREA program in this manner, this functionality is provided to allow a 3270 Transact program that depends upon designated Working Storage or Pseudo File data be saved and restored between interactions of the pseudo-conversation. Such application programs can still function without modification as a COMMAREA program requiring multiple invocations simulating a pseudo-conversation.

The CONVERSATION-ID field is used to specify a unique id that will be used to uniquely identify the saved Working Storage and Pseudo File objects between invocations of the COMMAREA program participating in a pseudo-conversation. It will also be used to maintain a record in CICS Temporary Storage that identifies the current state of each pseudo-conversation. The digits making up the TCP/IP address of the work station from which the COMMAREA input originates might be a good source for a unique conversation id.

Following is the layout of a COMMAREA for a simple application showing the TRANSACTION-CONTROL field:

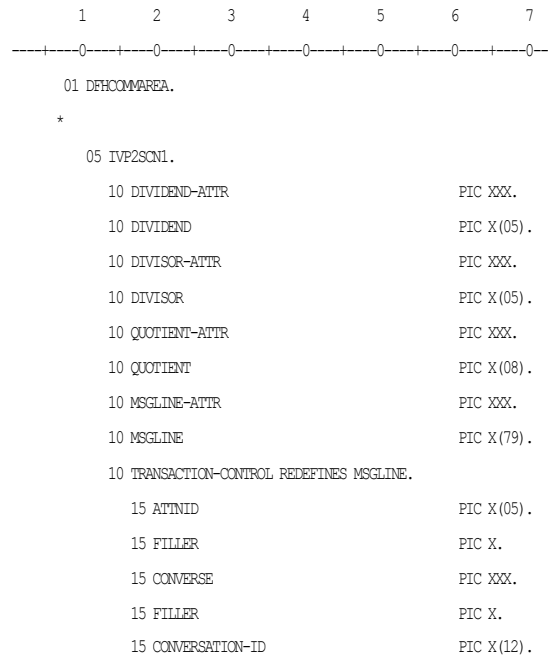


Figure 5-13 Transaction Control field layout

Structured Screens/COMMAREA And The Intermediate Structure

This section discusses structured screens, the handling of recurring segments, screen synchronized data bases, and the intermediate structure. The intermediate structure is the set of screen and screen synchronized data base (SSDB) segments available to the application during the interaction, the exact contents being determined by the timing of interaction functions and request or procedure references to data base fields. The input screen of an interaction establishes the intermediate structure. An understanding of the intermediate structure and its implications on processing the data base is prerequisite to understanding subordinate segment handling. In [The Intermediate Structure on page 6-13](#) we'll explore the building of the intermediate structure.

Structured Screens/COMMAREA

There are two types of screen formats processed by VISION:Transact: unstructured (flat) and structured (hierarchical). A flat screen contains fields which occur only once. While such a screen is suitable for many displays, it may become burdensome when dealing with a hierarchically structured data base or when displaying multiple records while browsing a file.

You can define and use screens with recurring fields and sets of fields (screen segments). The processing of structured screens is as automatic and non-procedural as the processing of unstructured screens.

The VISION:Transact structured screen is analogous to a single-leg hierarchical data base structure. The structured screen always has a singly occurring root segment. (To VISION:Transact, an unstructured screen is simply a screen with only a root segment.) Below the root segment in the screen, you may define up to eight levels of recurring segments. [Figure 6-1](#) is a conceptual sketch of a structured screen.

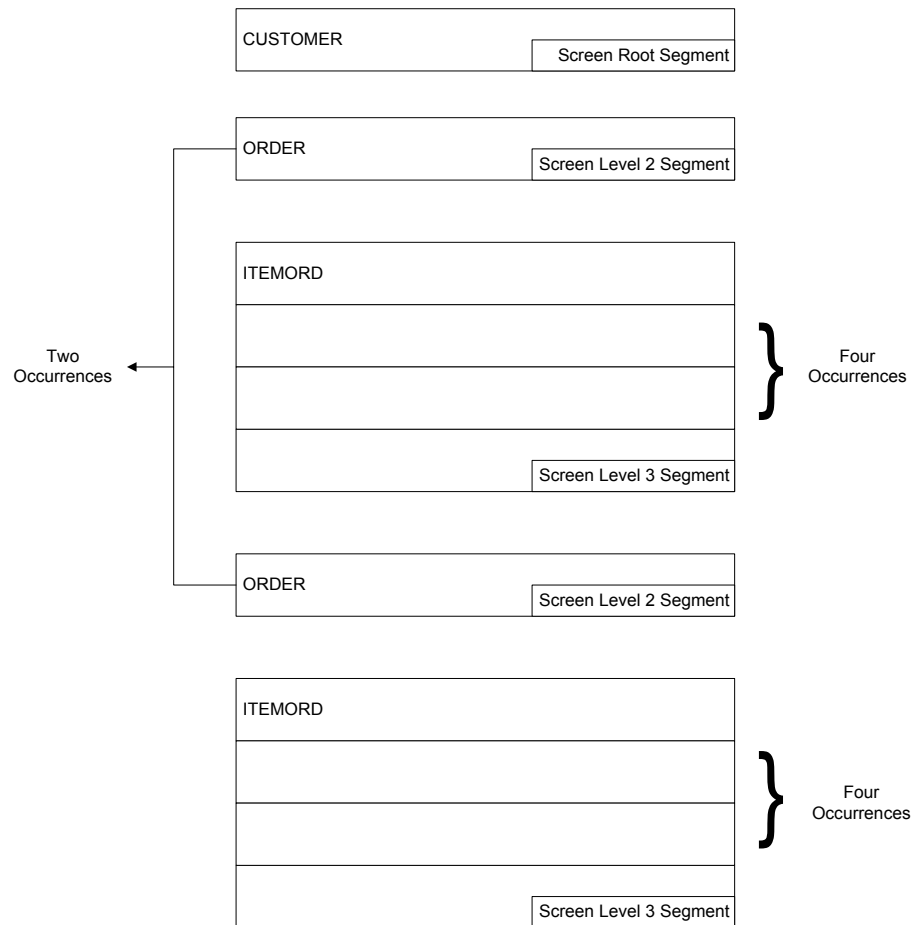


Figure 6-1 Conceptual Sketch of a Structured Screen

The screen in [Figure 6-1](#) allows the display and/or manipulation of the data for a customer and two orders from that customer. It can include data for up to four ordered items for each order. Usually your structured screen will be similar to the sketch in [Figure 6-1](#), but you are not limited to this design. A screen segment may begin anywhere on any screen line. Thus, a structured screen such as the sketch in [Figure 6-2](#) is also possible.

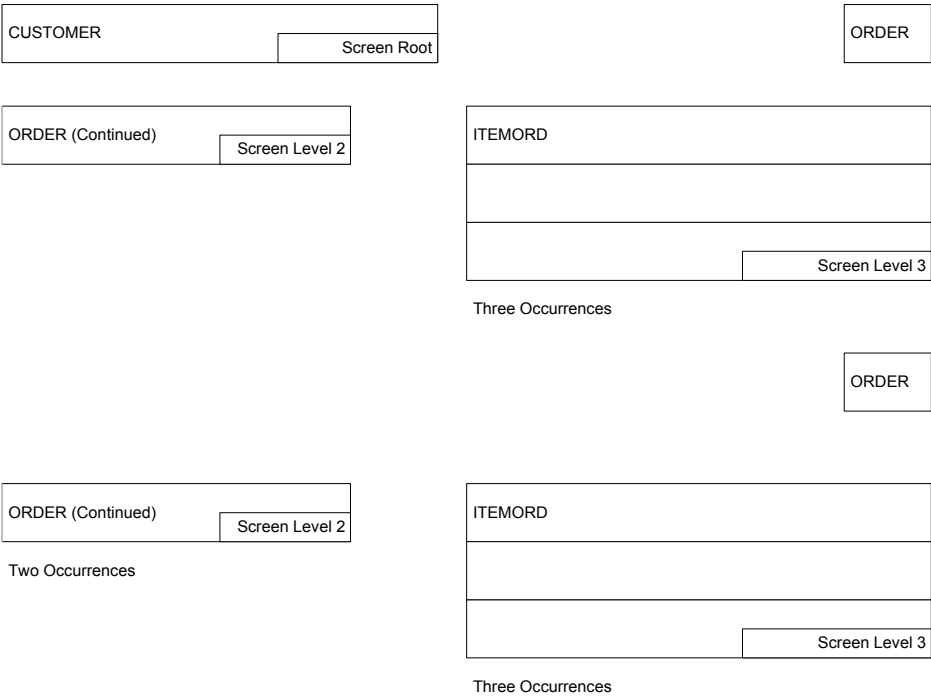


Figure 6-2 An Unusual Structured Screen

[Figure 6-2](#) is purposely an extreme example in order to suggest the breadth of structured screen design possibilities.

Another possible screen design uses the segment split concept, where fields from one screen segment can be displayed following all the fields from its dependents. [Figure 6-3](#) shows a sketch of how such a screen might look. Notice that the split segments "wrap" around all their dependents.

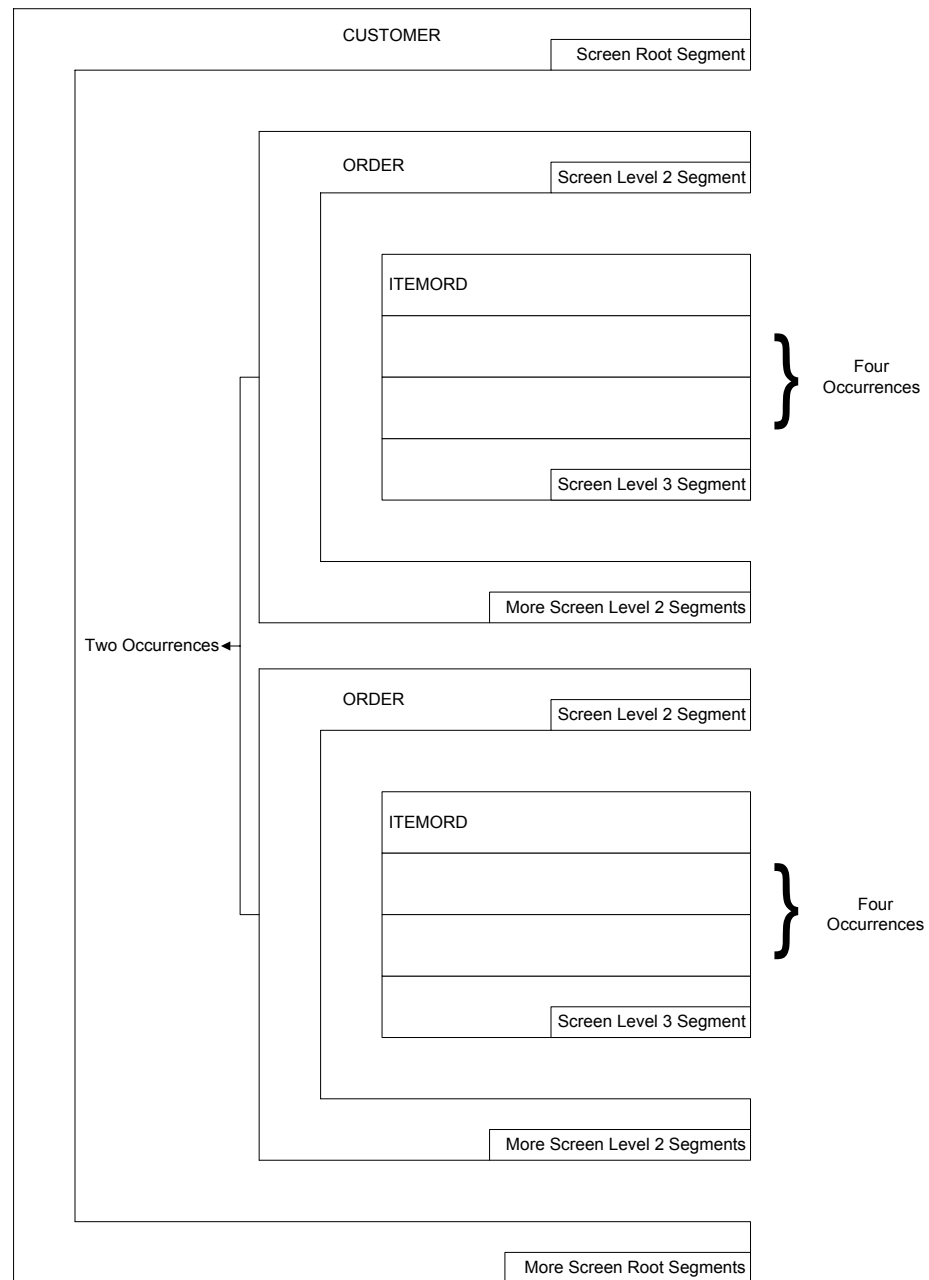


Figure 6-3 Conceptual Sketch of a Structured Screen with Split Segments

An A entry in the S0 statement V/S/A specification tells VISION:Transact that this screen segment will be split. The screen field thus designated and all subsequent screen fields in that segment will display after all dependent screen segments (to the bottom or right).

An example of a structured screen is shown in [Figure 6-4](#).

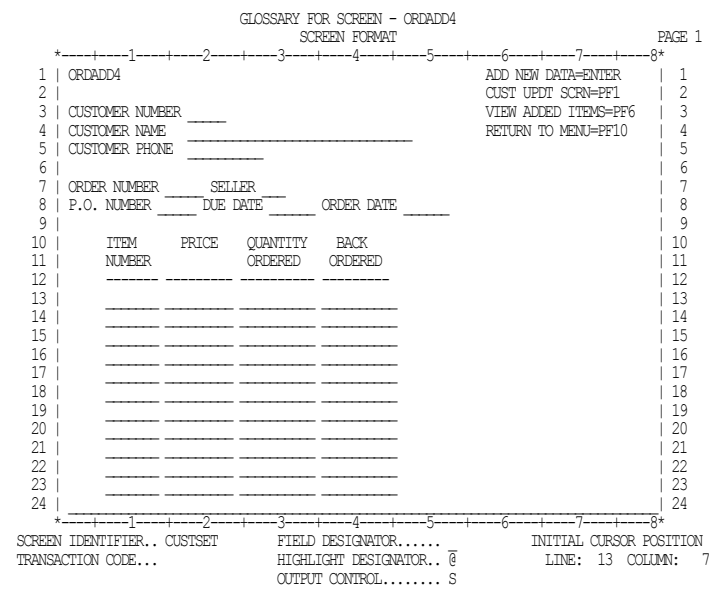


Figure 6-4 Order Add - A Structured Screen

An example of a structured COMMAREA corresponding to the above screen.

```
01 DFHCOMMAREA.
  *
  05 ORDADD4.
    10 ORDADD4-01.
      15 CUSTNO-ATTR PIC XXX.
      15 CUSTNO PIC X(05).
      15 CUSTNAME-ATTR PIC XXX.
      15 CUSTNAME PIC X(30).
      15 CUSTPH-ATTR PIC XXX.
      15 CUSTPH PIC X(10).
      15 ORDERNO-ATTR PIC XXX.
      15 ORDERNO PIC X(05).
      15 ORPERSON-ATTR PIC XXX.
      15 ORPERSON PIC X(03).
      15 ORPONUM-ATTR PIC XXX.
      15 ORPONUM PIC X(05).
      15 ORDUUDAT-ATTR PIC XXX.
      15 ORDUUDAT PIC X(06).
      15 ORDRDATE-ATTR PIC XXX.
      15 ORDRDATE PIC X(06).
      15 ORDADD4-02 OCCURS 11 TIMES.
        20 ITEMORD-ATTR PIC XXX.
        20 ITEMORD PIC X(07).
        20 ITEMPRICE-ATTR PIC XXX.
        20 ITEMPRICE PIC X(09).
        20 ITMQTYOR-ATTR PIC XXX.
        20 ITMQTYOR PIC X(10).
        20 QTYBKORD-ATTR PIC XXX.
        20 QTYBKORD PIC X(10).
    10 MSGLINE-ATTR PIC XXX.
    10 MSGLINE PIC X(79).
    10 TRANSACTION-CONTROL REDEFINES MSGLINE.
      15 ATTNID PIC X(05).
      15 FILLER PIC X.
      15 CONVERSE PIC XXX.
      15 FILLER PIC X.
      15 CONVERSATION-ID PIC X(12).
```

Figure 6-5 Order Add - A Structured COMMAREA

This screen's structure can be defined to VISION:Transact in at least the following ways.

- Screen root segment contains only data relating to the customer data base root segment called CUSTOMER. (Screen root only occurs once.)
- Screen level 2 segment occurs once and contains data relating to the customer data base ORDER segment (also at level 2).
- Screen level 3 segment occurs 12 times and contains data relating to the customer data base ITEMORD segment (also at level 3).

Or, since the screen allows for only one order:

- Screen root segment contains data relating to the customer data base CUSTOMER and ORDER segments (root and level 2).
- Screen level 2 segment occurs 12 times and contains data relating to the customer data base ITEMORD segment (level 3).

Processing efficiency is in favor of the second definition because it defines one less screen segment to process. Otherwise, it makes no difference which definition you choose. Thus, you have considerable latitude in designing screens for your VISION:Transact application. The following are a few constraints and guidelines:

- The screen structure should map closely to the structure of the data bases with which the screen interacts.
- The screen structure applies only to data fields. Literals appear only where they are coded in the screen layout.
- For input, data base segment key fields must be included for all data base segments to be accessed in the path through the lowest segment in the data base used on the screen and in the same relative sequence as they occur in the data base hierarchy. These data base key fields may be distributed over several screen segment types.
- If the screen segment is not split, the contents of two or more screen segments may not overlap. The entire root segment must be first (left-to-right, top-to-bottom) followed by the first occurrence of the level 2 segment, followed by the first occurrence of the level 3 segment...followed by the occurrences of the lowest level screen segment.
- If all input fields in a screen segment occurrence contain blanks, the segment is considered "missing." No Sn-set processing, including input-required validation, is performed on that screen segment occurrence. You can force the application to process blank screen segments by using the Null Segment Processing specification on the SD statement. Refer to the *Advantage VISION:Transact Reference Summary* for more details on this facility.
- If a structured input screen has input fields at screen segment 2, then it must also have at least one input field at screen level 1. This field may be a virtual screen field.

Data Accessibility to the Interaction

The screen processing definition for a screen display that is input to an interaction determines the portion of the data base available for the duration of that interaction. A screen processing definition containing input fields can reference data base segments along any path(s) of the data base. The segment actions for the input key/search fields, plus the key values contained in those screen fields, select the data base segment occurrences to be processed during the interaction. No other occurrences of these segment types (which are said to be synchronized to the screen) are available for the duration of the interaction. The application's view of the data base(s) during an interaction can be expanded to include dependents of segments in the previously established view, as well as segments along any other path, by request or procedure processing. Request and procedure processing are discussed in detail in [Chapter 7, "Requests/Procedures and the Intermediate Structure"](#).

A Screen Synchronized Data Base View

Consider the following example of an interaction using the CUSTOMER data base on the ORDUPDT4 screen from the ORDER/ITEM application, whose source code can be found in [Appendix A, "Order/Item Application Screen and Database Definition Code"](#). [Figure 6-6](#) is the ORDUPDT4 screen display which is input to the interaction. The display shows the details of order number 10001 for customer 00001. Four of the items displayed already exist on the order: G38, G40, G41, and G46. One item, G43, is being added to the order, and two items, G41 and G46, are being updated.

```

ORDUPDT4
CUSTOMER NUMBER 00001
ORDER NUMBER    10001
CUSTOMER NAME   TULSA TIMES

UPDATE=U ACT + ENTER
CUST UPDT SCR=PF1
RETURN TO MENU=PF10

```

ACT ION	ITEM NUMBER	ITEM NAME	PRICE	QUANTITY ORDERED	BACK ORDERED	QTY ON HAND
-	G38	1''- BINDER - A/2	1.50	25	0	125
	G40	A/2 REFERENCE MAN.	15.00	10	0	146
u	G41	A/2 CONC. & FAC.	5.00	25	0	961
u	G46	A/2 OPS. GDE.-VSE	23.00	25	0	475
u	g43			25		

Figure 6-6 Screen Input to the Interaction

[Figure 6-7](#) is a diagram of the Customer data base structure, indicating subordinate segment occurrences for record 00001.

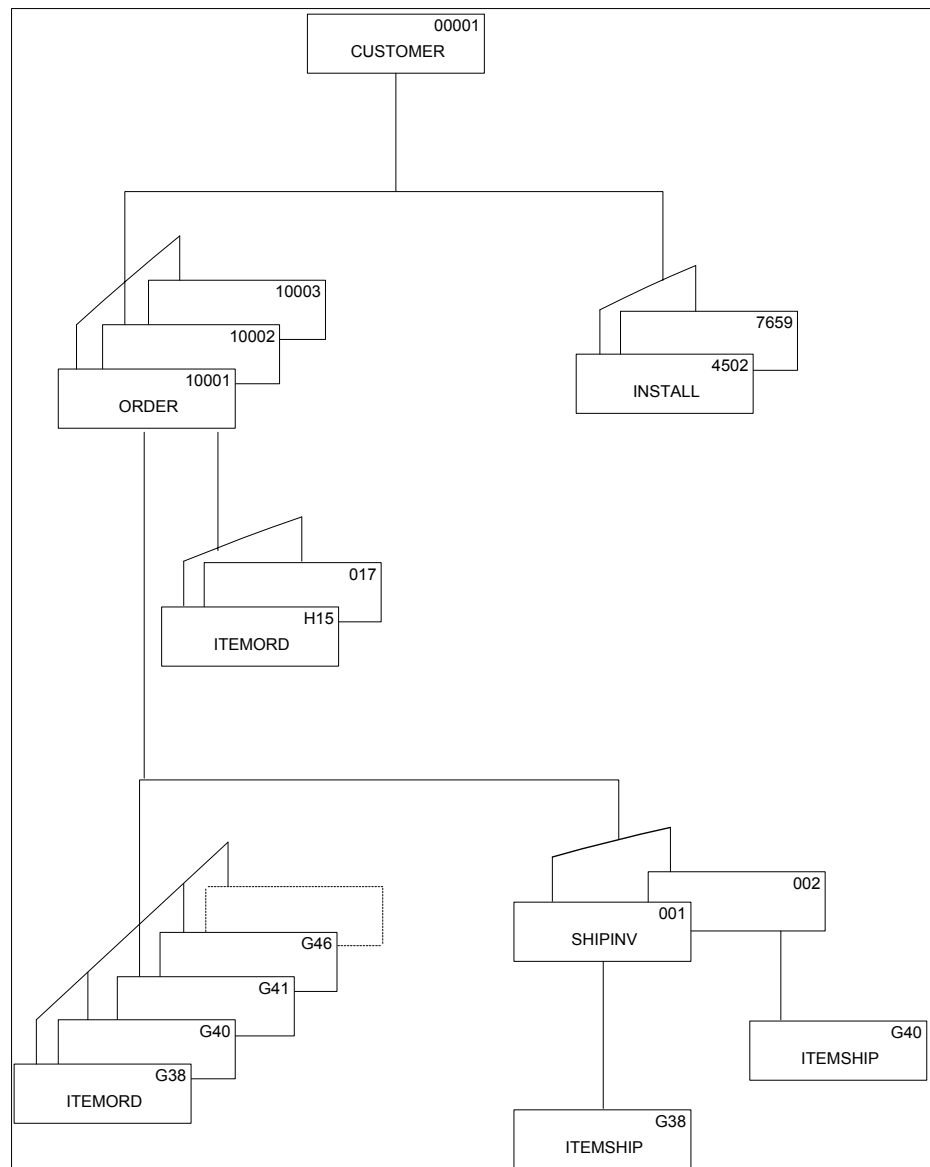


Figure 6-7 CUSTOMDB Data Base Record for Customer Number 00001

The screen content/action specifications for ORDUPDT4, the input screen for this interaction, include:

- Match (M) on the CUSTOMER segment.
- Repeat (Q) on the ORDER segment, in this instance inheriting the match (M) action from its parent CUSTOMER segment.

Either:

- ignore (X) on those ITEMORD segment occurrences for which nothing has been entered under ACTION on the screen,
- update or insert (V) on those ITEMORD segment occurrences for which a U has been entered under ACTION.

The actions performed on the ITEMORD segment are dynamically selected based on facilities described in [Dynamic Selection of Action Codes on page 4-9](#).

Remembering that the screen input to this interaction contains the data shown in [Figure 6-6](#), consider the following description of data availability to the interaction.

After successful automatic screen validation and processing, the interaction's data base view is as shown in [Figure 6-8](#).

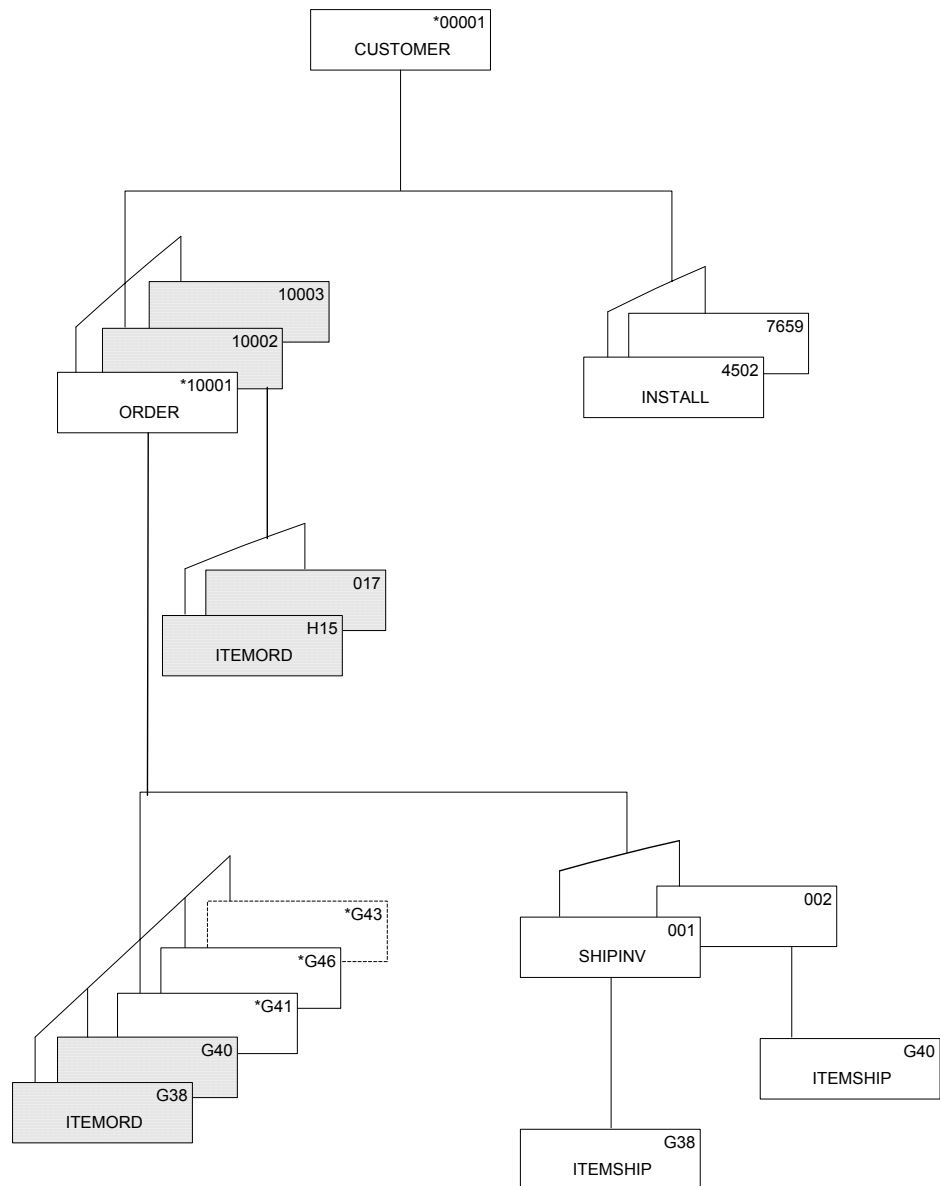


Figure 6-8 Interaction View After Input Screen Processing (Page 1 of 2)

Access Established

(segment occurrences with * in Figure 6-7)

CUSTOMER	00001	
ORDER	10001	
ITEMORD	G41	
	G46	
	G43	(new occurrence)

Potentially accessible through request processing

SHIP INV	001
ITEMSHIP	G38
SHIPINV	002
ITEMSHIP	G40
INSTALL	4502
	7659

Not accessible to the interaction

(shaded segment occurrences in Figure 6-7)

ITEMORD	G38	(for order 10001)
ITEMORD	G40	
ORDER	10002	
ITEMORD	H15	
	017	
ORDER	10003	

Figure 6-8 Interaction View After Input Screen Processing (Page 2 of 2)

The segments MARKed with an asterisk are available to the VISION:Transact generated application. Those segments are synchronized to the screen as a result of segment actions performed during the interaction.

Certain segment occurrences are shaded to indicate that they are unavailable to the interaction. These occurrences have been restricted from the application during this interaction because one of their twin occurrences or a twin of one of their parents has been synchronized to the screen. For instance, ITEMORD (G38) is unavailable because no direct match was made on it and because a direct match was made on its twin, ITEMORD (G41).

The segments that are not shaded and do not contain an asterisk are segments that are potentially available to the application during this interaction through request or procedure processing. All occurrences of a lower level segment in an SSDB record subordinate to an available parent are available to the application during an interaction if no occurrence of that segment has been synchronized to the screen.

In [Figure 6-8](#) this condition is true of the INSTALL, SHIPINV, and ITEMSHIP segments in the 00001 Customer data base record. [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#) discusses how these can become available through request or procedure logic. The next section discusses how any of the segments can be made available.

The segment number surrounded by broken lines (G43) will be inserted during this interaction.

In effect, the segment actions on the input screen processing definition function as SSDB record selection criteria for the duration of the interaction. Limiting the application's data view has the advantage of limiting the amount of looping necessary. Looping is the automatic processing of multiple occurrences of segments by the application's screen and data base manipulation.

Processing All Occurrences of a Screen/COMMAREA Synchronized Data Base

Occasionally this limiting of the application's data view is not desirable. For instance, you might want access to all of the data base record's occurrences for display on the interaction's output screen, even though some of the lower level segments in that record have been synchronized to the input screen of the interaction (i.e., update specific occurrences but output screen requires all occurrences to be displayed). In such a situation, you have several alternatives:

- You can view the data base as a separate dynamically synchronized data base (see [Chapter 8, “Dynamic Synchronization”](#)) or as another SSDB or as both an SSDB and DSDB.
- You can use the SG segment operator and its parameter SELECT ALL on a screen control statement to release the segment named there and all its dependents from the intermediate structure for output processing. See [The Segment Operator on Conditional Screen/COMMAREA Control Statements on page 6-21](#) for more information on using the SG operator in screen control.

The Intermediate Structure

The intermediate structure is the application's view of the data it will process. The intermediate structure is the in-memory organization of data from the input screen and/or data base records or segment occurrences. The data base records or segment occurrences may be screen synchronized (discussed below), dynamically synchronized (discussed in [Chapter 8, “Dynamic Synchronization”](#)), or accessed with request or procedure logic (discussed in [Chapter 7, “Requests/Procedures and the Intermediate Structure”](#)). The data in the intermediate structure may change over the course of an interaction.

The concept of the intermediate structure is important in order to understand how your VISION:Transact generated application handles references to screen and data base segments.

The screen processing definition defines the initial hierarchy and contents of the intermediate structure. The executing application, using the input screen content in conjunction with the screen processing definition, builds the intermediate structure during each interaction. For example, the ORDUPDT4 screen in [Figure 6-6](#) would establish the initial intermediate structure shown in [Figure 6-9](#). This initial intermediate structure is built from the input screen fields.

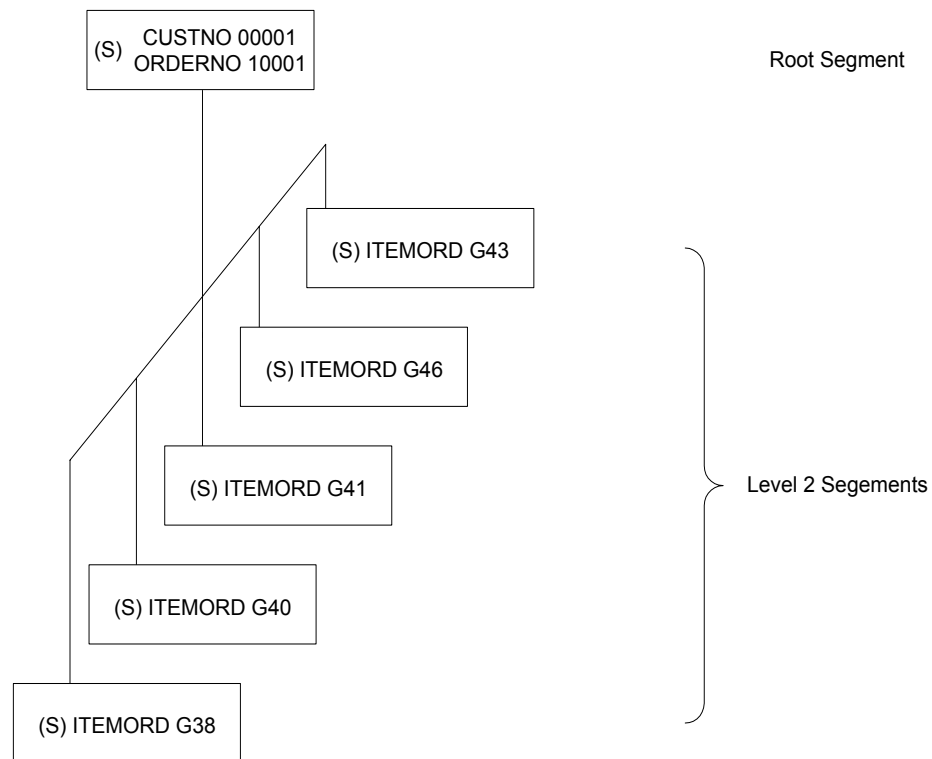


Figure 6-9 Initial Intermediate Structure

The segments illustrated in [Figure 6-9](#) are screen segments. They are distinguished by the identifier S. The names in the segments are the names of screen fields which are used in the data base read and the values shown are the key fields entered.

The next interaction function is the data base read. Because the reads are done in accordance with the S0 statements, these data base segments are synchronized to the input screen forming a composite segment.

Remember that the segment actions for this interaction include matches (M) on the CUSTOMER and ORDER segments, and either ignore (X) or update/insert (V) on different occurrences of the ITEMORD segment of the customer data base.

Not mentioned earlier in this section is the fact that the ORDUPDT4 screen also updates the Item data base. The actions performed on the ITEMROOT segment of the item data base are parallel to those performed on the ITEMORD segment of the customer data base. When no action (X action) is taken on the ITEMORD segment, none is taken on the ITEMROOT segment; when an update/insert (V) action is taken on the ITEMORD segment, an update (U) action is taken on the ITEMROOT segment.

Non-X segment actions cause matching data base segments to be retrieved. [Figure 6-10](#) shows the contents of the intermediate structure after the data bases are read.

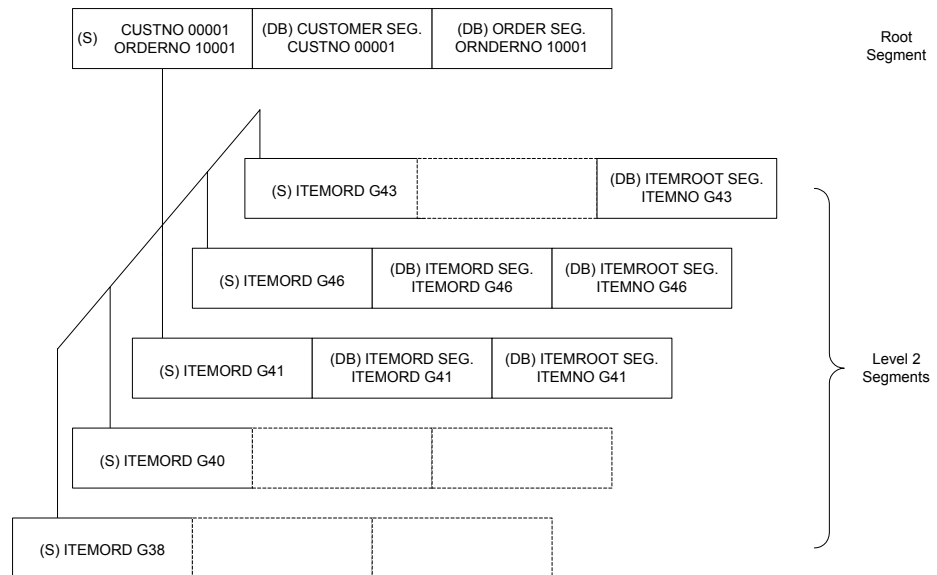


Figure 6-10 The Intermediate Structure After the Data Base Read

The intermediate structure in [Figure 6-10](#) shows the retrieved data base segments attached to their matching screen segments. The resulting intermediate structure segments are called composite segments. The identifier "(DB)" along with the segment name distinguishes the data base segments in each composite segment.

We have synchronized both the level 1 (CUSTOMER) and level 2 (ORDER) segments from the Customer data base to the root of the intermediate structure, and we have synchronized the level 3 (ITEMORD) segment from the Customer data base and the level 1 (ITEMROOT) segment from the Item data base to the level 2 segment of the intermediate structure. This synchronization establishes the limited view of the Customer data base record shown in [Figure 6-8](#). It also limits our view of the Item data base for this interaction to the three ITEMROOT segments in [Figure 6-10](#) and their subordinate segments.

There are no data base segments concatenated to the G38 and G40 screen segments because the X segment action associated with those occurrences of the ITEMORD screen field inhibits the retrieval of data base segments for those occurrences of the intermediate structure. No ITEMORD segment occurrence is concatenated to the G43 composite segment, but that data base segment will be inserted later in this interaction.

[Figure 6-11](#) illustrates the contents of the intermediate structure after an in-memory update has occurred.

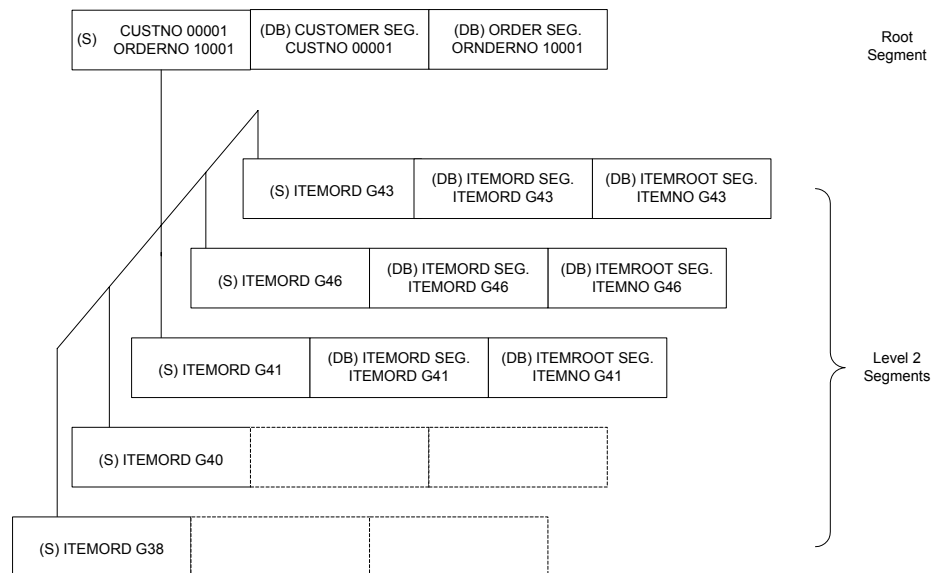


Figure 6-11 The Intermediate Structure After In-Memory Update

The root of the intermediate structure consists of data in the screen root and the data base segments synchronized to the screen root. Each lower level composite segment with a non-null (not X or N) segment action has a lower level screen occurrence concatenated with the data base occurrence for which the segment action is specified. The G43 composite segment now contains an ITEMORD segment, since the in-memory update has been completed. After the in-memory update, the application replaces (or inserts, in the case of the G43 ITEMORD segment) the updated data base segments into the data bases.

[Figure 6-12](#) shows the addressable intermediate structure for output processing which follows the replacement or insertion of updated data base segments.

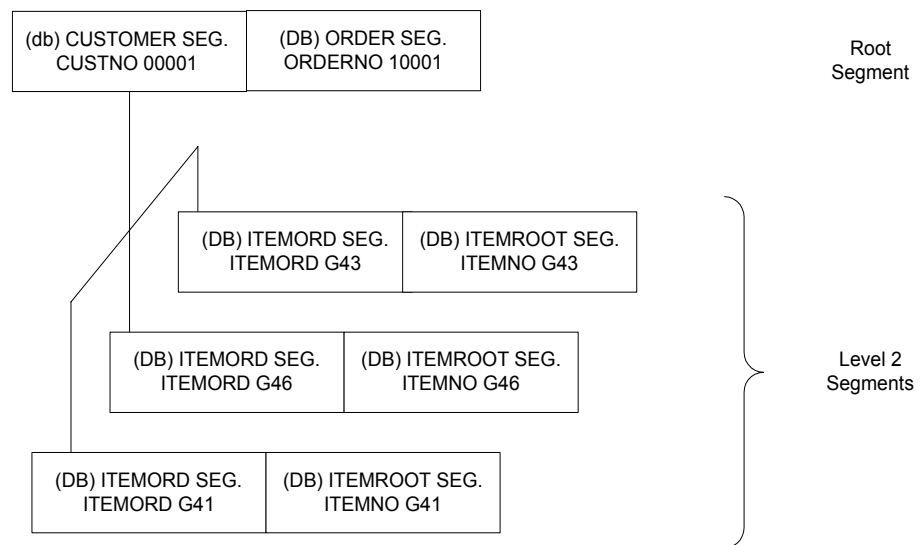


Figure 6-12 Addressable Portion of the Intermediate Structure Following Replacement/Insertion of Updated Data Base Segments

The intermediate structure now contains neither the screen segment portion of the composite segments nor the occurrences of the intermediate structure that used X actions. Screen fields are no longer available for output processing, because the input screen processing has been completed and the output screen for this interaction, which may not be the same display as the input screen, is chosen at this point, right after the updates are applied to the data bases. The segments of the intermediate structure for which X was the only segment action chosen contained only screen segments during input screen processing. Since screen fields are no longer available, the occurrences of the intermediate structure containing only screen fields now disappear. These intermediate structure occurrences would only remain available if they had active synchronized dependents (see point 6 in [The Intermediate Structure and Output Screen/COMMAREA Processing on page 6-19](#)), though their SSDB components would be "missing."

[Chapter 7, “Requests/Procedures and the Intermediate Structure”](#) describes how the intermediate structure can be manipulated and expanded at each stage of the interaction depicted in [Figure 6-9](#) through [Figure 6-12](#).

Execution tracing, described in [Chapter 12, “Execution Tracing”](#), can be used to output useful information concerning the contents of the intermediate structure.

Segment Accessibility Rules

Based upon the concepts of screen synchronization and data base segment accessibility (discussed in [Data Accessibility to the Interaction on page 6-7](#)) the following rules apply to screen-synchronized data bases:

1. The screen processing definition must perform segment actions against the SSDB in hierarchical order, i.e., parent-segment qualification must be provided before a dependent segment can be accessed.
2. A screen synchronized segment occurrence is available to the application during an interaction, based on the rules in [Segment Actions and Data Base Availability to Screen/COMMAREA Requests and Procedures on page 7-35](#). Synchronized occurrences form a subset of available occurrences.
3. If a segment occurrence is synchronized, no unsynchronized occurrences of the same segment type are available to the application during this interaction. This focuses the application on selected data base occurrences during an interaction.
4. A data base segment occurrence is available to the application only if its parent is available.
5. If an output screen display references fields in data base segments that were synchronized to the input screen, only those segment occurrences from the intermediate structure are made available for display (unless you include a screen control statement with the segment operator SG; see [The Segment Operator on Conditional Screen/COMMAREA Control Statements on page 6-21](#)).

Looping on the Intermediate Structure

VISION:Transact assumes that a reference to a field in a multiply-occurring segment (screen or data base) constitutes a reference to all available occurrences of that segment, and processes each segment occurrence in turn. Because screen and screen-synchronized segments are referenced through the intermediate structure, references to fields in those segments cause all occurrences of the composite segment in the intermediate structure to be processed.

Looping in your VISION:Transact generated application is the processing invoked by a reference to a field in a repeating segment. For instance, if you have 50 occurrences of one lower level segment and you reference a field in that segment, VISION:Transact will process all 50 occurrences. This will not cause a read if the occurrence is already in the intermediate structure.

By using the intermediate structure, multiple levels of data base segments can be processed as a single unit, as in the example in [The Intermediate Structure on page 6-13](#). If the input screen segment contains actions on more than one data base segment, both will be contained in the same composite segment of the intermediate structure and can be processed with a single pass.

Execution tracing, described in [Chapter 12, "Execution Tracing"](#), can be used to trace looping that occurs on the intermediate structure.

The Intermediate Structure and Output Screen/COMMAREA Processing

The following points illustrate the effect looping has on the intermediate structure during output screen processing.

1. Because output screen processing is based on the intermediate structure built during input processing, the content of the output screen can vary according to the structure of the input screen segments in that particular interaction.
2. If you wish to display all occurrences of a data base field on your output screen, you must take care that the input screen for that interaction does not specify segment actions for that data base segment. If it does, you must use the SELECT ALL parameter of the SG segment operation in screen control to make all occurrences available (see [The Segment Operator on Conditional Screen/COMMAREA Control Statements on page 6-21](#)).
3. Updated data base segments displayed on the output screen will be shown in the order of their preceding input, not necessarily in the order of their keys.
4. Multiple occurrences of a lower level screen segment are built for display only when the output screen processing definition references fields from structures on which looping can be initiated. Fields from a lower level segment of the intermediate structure, fields from a lower level SSDB or pseudo file segment and data bases being browsed (see [Chapter 8, "Dynamic Synchronization"](#)) are structures on which looping can be initiated. Looping during output processing is based on references to these structures, not on the hierarchy of the output screen. The number of lower level segment occurrences defined for the output screen does not start looping, but can only limit the looping during the building of the output display.
5. SSDB records are available to output processing as long as they were retrieved or inserted (and not deleted) during input processing.

- 6. The root of the intermediate structure always exists, even though the SSDB segments that the root contains might be "missing" from the composite segment during output processing due to the deleting or bypassing of an SSDB record read by input processing. Composite segments that are the parents of any SSDB segments available for output processing also exist, even if their component SSDB segments are "missing."

Execution tracing, described in [Chapter 12, "Execution Tracing"](#), can be used to trace looping that occurs on the intermediate structure.

The Source of Screen/COMMAREA Synchronization

You specify all data bases to be used and how they will be used on the application generation's RF statements. You also assign the qualifiers to be used when referring to a data base. [Figure 6-13](#) shows the RF statements for the Customer and Item data bases used in the example in this section.

RUN	STMT	FILE	FILE	SYNCHRONIZING		SYNC	SYNC
NAME	TYPE	NAME	TYPE	FIELD 1	FIELD 2	QUAL	TYPE

(M510	,RF)	(CUSTOMER)	(FILE)				(U)
(M510	,RF)	(ITEM)	(FILE1)				(U)
(M510	,RF)	(ITEM)	(FILE2)	(SYNCFLD1)	(SYNCFLD2)	(T)	

Figure 6-13 Application File Statements for Customer and Item Data Bases

Because the application will update both of these data bases, they both must be screen synchronized. This is done by an entry of U in the Screen Sync Type specification. Notice that the File Type entry assigns the qualifier. The Customer data base entry of FILE associates the blank qualifier with the screen synchronized Customer data base, while the entry FILE1 tells VISION:Transact that qualifier 1 refers to the screen synchronized Item data base.

In [Chapter 8, "Dynamic Synchronization"](#) you will see how to specify dynamic synchronization using RF statements.

The Segment Operator on Conditional Screen/COMMAREA Control Statements

You saw in [Processing All Occurrences of a Screen/COMMAREA Synchronized Data Base on page 6-13](#) that you use the SG operator to release screen synchronized data base segments from the intermediate structure. You do this to process all occurrences of a segment type during output processing. However, there may be cases where you want to release segments only when a particular screen is output. Otherwise you want the application to display only the segment occurrences that were synchronized to the screen.

The SG operator can be included in conditional screen control to control which output screens are affected by the release of synchronized occurrences. Depending on where the SG operator is placed in the SC statements, the SELECT ALL can affect:

- All possible output screens.
- Only output screens that meet some selection criteria.

[Figure 6-14](#) shows an example of each of these cases.

INPUT	SIMT	SEQ	LOG	CON	.OPERAND A..	OPER	OPERAND B.....	OUTPUT	.OPERAND C..
SCREEN	TYPE	NO.	LVL	CTR	QLF	FIELD	ATN QLF FIELD NAME/CONSTANT	SCREEN	QLF FLD/CNST
(IN1	)	(SC)				(ORDER	) (SG) (SELECT ALL	)	
(IN1	)	(SC)				(F,ATINID	) (EQ) (C,ENTER	) (OUT1	)
(IN1	)	(SC)			(E)	(F,ATINID	) (EQ) (C,PF001	) (OUT2	)
(IN1	)	(SC)			(E)			(OUT3	)
(IN2	)	(SC)				(T,SELECT	) (EQ) (C,X	)	
(IN2	)	(SC)				(ORDER	) (SG) (SELECT ALL	)	
(IN2	)	(SC)						(OUT1	)
(IN2	)	(SC)						(OUT2	) (C,ALTX
(IN2	)	(SC)			(E)			(OUT3	)
(IN3	)	(SC)				(T,SELECT	) (EQ) (C,X	)	
(IN3	)	(SC)				(ORDER	) (SG) (SELECT ALL	)	
(IN3	)	(SC)						(OUT1	)
(IN3	)	(SC)			(E)	(T,SELECT	) (EQ) (C,Y	) (OUT2	)
(IN3	)	(SC)			(E)			(OUT3	)

Figure 6-14 Using the SG Operator

In the first example, because the SG SELECT ALL precedes any relational tests for the input screen, it affects all subsequent output screens. In the second example, the SG SELECT ALL follows the relational test T,SELECT EQ CX, and therefore affects only those selected output screens that follow it (OUT1 and OUT2). In the last example, the SG SELECT ALL follows a relational test and also is followed by another relational test. In this case, the SG SELECT ALL affects only the screen that follows the first relational test (OUT1). The screens that follow the second relational test (OUT2 and OUT3) would process only screen synchronized occurrences.

Intermediate Structure Summary

[Figure 6-15](#) highlights the intermediate structure components of a VISION:Transact generated application.

The intermediate structure consists of data from the screen and screen synchronized data bases. The initial contents of the intermediate structure are determined by the input screen's screen processing definition and the data input from the screen during the execution of the application. Over the course of the interaction the contents of the intermediate structure can change. Interaction functions change the contents of the intermediate structure. Screen requests or procedures, discussed in [Chapter 7, "Requests/Procedures and the Intermediate Structure"](#), are another way to change the contents of the intermediate structure.

Execution tracing, described in [Chapter 12, "Execution Tracing"](#), can be used to output useful information concerning the contents of the intermediate structure and the status of loops on the intermediate structure.

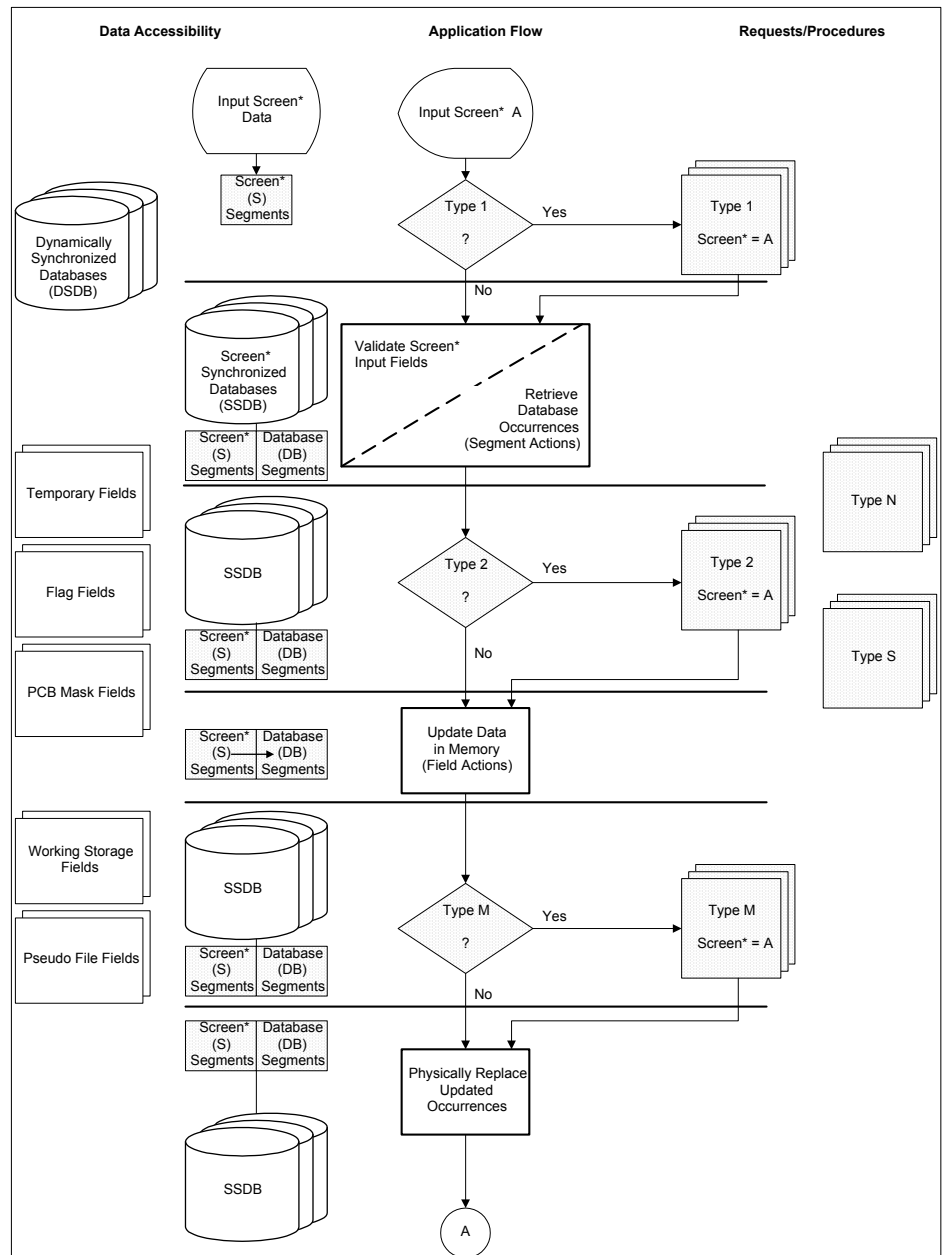


Figure 6-15 Intermediate Structure Components (Page 1 of 2)

* "Screen" can also be "COMMAREA section" in CICS

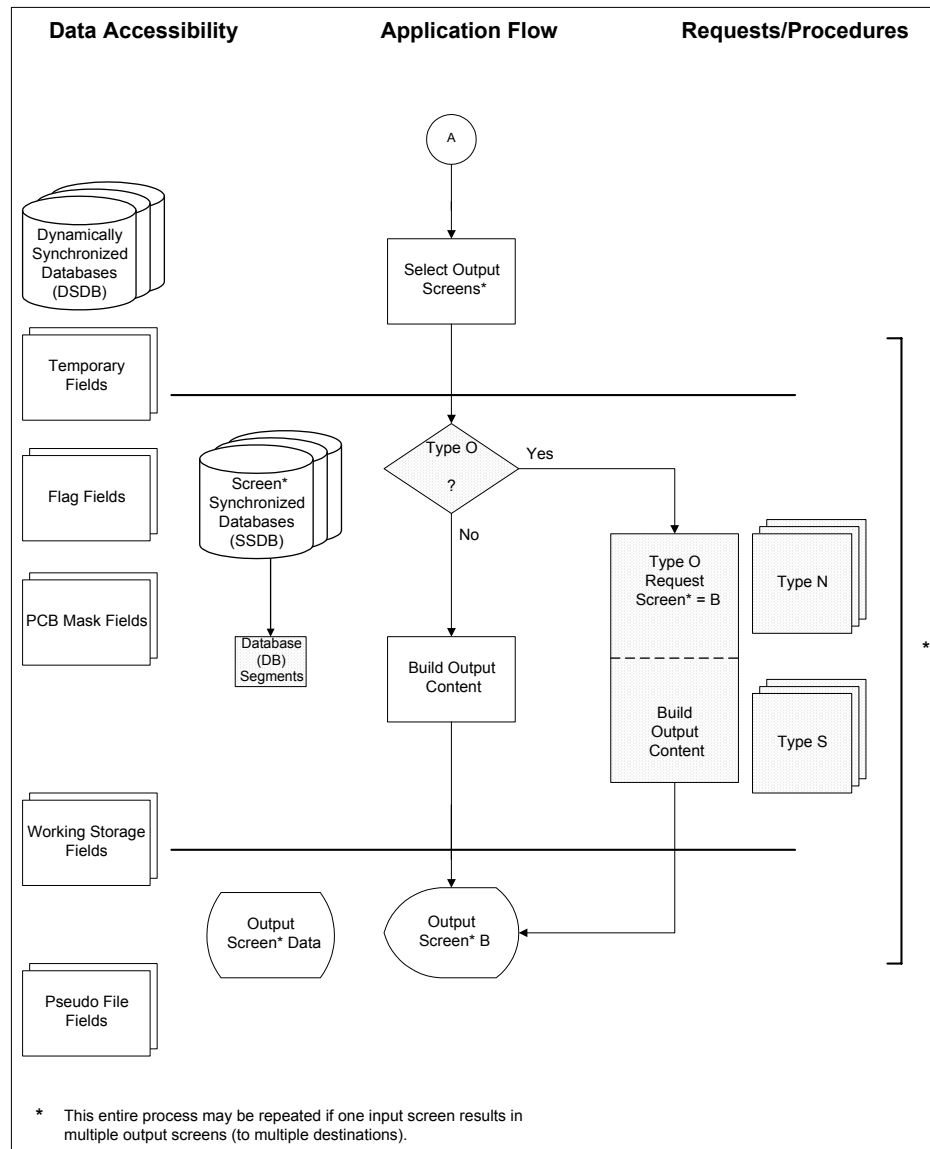


Figure 6-15 Intermediate Structure Components (Page 2 of 2)

* "Screen" can also be "COMMAREA section" in CICS

Requests/Procedures and the Intermediate Structure

Overview

The interaction cycle as we have previously discussed it has been limited to the processing and data base activities which can be specified with the screen processing definition content/action specifications and screen control. These actions as well as validation and editing of screen data in the screen processing definition are all that are required for many interactions. However, some interactions require additional processing and/or data base updating. Such requirements include the following:

- Arithmetic calculations.
- Complex inter-field validation.
- Performing validation and error handling on multiple fields in a single interaction.
- Updating data base segments which are not referenced on the input screen.
- Calling a standard routine to calculate a square root, perform a date conversion, etc.

VISION:Transact screen requests or procedures perform these and other complex processing functions. Conceptually, requests and procedures are the same, however, they do use different syntax. Requests are written using fixed form statements for procedural code (PR statements), temporary field definition (TF statements), and comments (AA statements). The *Advantage VISION:Transact Reference Summary* documents the syntax of request statements. Procedures are written using a free form language called ASL. The *Advantage VISION:Transact ASL Reference Guide* documents the syntax and structure of ASL statements.

Note: The term “screen” can also be read as “COMMAREA section”.

Each screen request or procedure (except a subroutine or type N) can be associated with a particular screen processing definition. A screen request or procedure is executed at one of four points during interactions involving the screen with which it is associated. The specific point in the interaction at which the screen request executes is determined by its type.

This section describes your ability to extend the interaction using screen requests/procedures. First, the request/procedure is introduced and its facilities are described in general terms. Then the uses of each type of screen request/procedure are described. Last, the relationship between screen requests or procedures and the intermediate structure is explored regarding data availability to screen requests, the processing of virtual fields, and looping on the intermediate structure in screen requests/procedures.

Screen/COMMAREA Requests/Procedures

VISION:Transact requests or procedures are distinguished by their type, which is specified on the ER statement shown in [Figure 7-1](#).

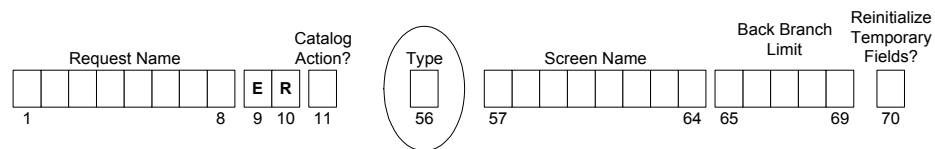


Figure 7-1 Request/Procedure ER Statement

Type N (or Blank) Requests/Procedures

If the type is N or blank, the request or procedure can contain only comments and temporary field definitions. Requests/procedures of this type are associated with the application as a whole, not with any particular screen in that application. The purpose of type N is to define temporary fields.

A temporary field, as the name suggests, is simply a place to store a data value temporarily.

Note: Another form of temporary data storage is working storage. Working storage is defined with a VISION:Transact file definition. Working storage fields may be over-defined (multiply-defined; redefined) where temporary fields may not. However, temporary fields may be defined to include an initial value, but working storage fields may not and are initialized by VISION:Transact to null values (blanks or zeroes depending on the field type). Pseudo fields are also another form of temporary data storage. Pseudo fields are defined with the file definition similar to working storage. Pseudo files may be structured whereas working storage may not; working storage may only have a root segment.

Temporary fields are initialized at the beginning of every interaction. A value placed in a temporary field is available for use for the duration of the interaction. The TFREQ keyword in M5PARM determines how temporary fields may be defined. A value of Y for the TFREQ keyword requires all temporary fields to be explicitly defined using the TF statement (requests) or FIELD command (procedures). A value of N for TFREQ allows temporary fields to be explicitly defined or implicitly defined by simply naming the temporary field in a screen request/procedure, an S0 or Sn statement (screen content/action) or an SC statement (screen control). Rules for explicit temporary field definition are in the *Advantage VISION:Transact Reference Summary* (requests) or *Advantage VISION:Transact ASL Reference Guide* (procedures). More information on the TFREQ keyword in M5PARM can be found in the *VISION:Transact for CICS Installation Guide*.

Type 1, 2, M, and O Screen/COMMAREA Requests/Procedures

Screen requests or procedures can perform validation, processing and record selection, as well as define temporary fields and comments. Each screen request/procedure is associated with the processing of a particular screen either by naming the screen on the ER statement or by naming the request or procedure on an XR or OS operator in screen control (see [Chapter 11, “Advanced Application Flow”](#)). There are four types of screen requests or procedures: 1, 2, M, and O. A type O screen request/procedure is executed during the building of the output screen display. Type 1, 2, and M are executed during input processing of the input screen.

Output screen requests or procedures, type O, are commonly used to select segments or perform calculations for display to the terminal operator, after the output screen display has been selected by the interaction's screen control.

Input screen requests or procedures, types 1, 2, and M, are commonly used to perform additional validation and to ensure the correctness of data base updates. For instance, you can use them to check which program function (PF) key was pressed by the terminal operator, to determine alternative methods of field validation, or you can use them to perform inter-data base comparisons or calculations prior to or after the update of multiple data bases. They can also be used to bypass some of the automatic processing controlled by the screen processing definition.

Subroutine Requests/Procedures

Subroutine requests or procedures, type S, are used to perform common processing that is required by more than one screen. A type S request/procedure is not associated with the processing of a particular screen. Instead, it is invoked by another request or procedure to which control returns when the type S request or procedure is finished.

Request Operations and Procedure Commands

The syntax and the available operators of the selection, calculation and processing (PR) statements for writing requests are documented in the *Advantage VISION:Transact Reference Summary*. The following table is a summary of the operators which are available for use only in a VISION:Transact request.

Summary of Request Operators

Type	Operator	Usage
Relational	EQ	Is Operand A equal to Operand B (or matches the pattern in Operand B?).
	NE	Is Operand A not equal to Operand B (or does not match the pattern in Operand B?).
	LT	Is Operand A less than Operand B?
	GT	Is Operand A greater than Operand B?
	LE	Is Operand A less than or equal to Operand B?
	GE	Is Operand A greater than or equal to Operand B?
Branching	GO	Unconditional forward branch to a statement sequence number, END, OUTPUT, SCREEN CONTROL, NEXT FUNCTION, NEXT REQUEST, SUB request-name, RETURN, or BACK.
	NS	Forward conditional branch if immediately preceding relational expression is false.
	GS	Allowable branch locations same as GO. Forward conditional branch if immediately preceding relational expression is true.
Arithmetic	+	Add Operand A to Operand B and store result in Operand C.
	-	Subtract Operand B from Operand A and store result in Operand C.
	*	Multiply Operand A and Operand B and store result in Operand C.
	/	Divide Operand A by Operand B and store result in Operand C.
Replacement	R	Move Operand A to Operand C or Move Operand B to Operand C

Type	Operator	Usage
Set Error	SE	Marks the input screen field named in Operand A for error handling if the input screen is redisplayed. Each field marked is highlighted (if error highlighting was specified in its S0 statement) when the input screen is redisplayed. The cursor is positioned at the first field in error.
Message Sending	MS	Redisplay input screen with the message specified by Operand B on the error message line (the screen field that has the MSGLINE flag as its source field); terminates the interaction. Data base updates are backed out unless an entry is specified in Operand A of the MS operator.
Execution Trace	?C ?X	Causes the contents of Operand B to be displayed within the trace output during online application execution. For more information, see Chapter 12, "Execution Tracing" .

Summary of ASL Procedure Commands

The syntax and available commands for writing procedures are documented in the *Advantage VISION:Transact ASL Reference Guide*. The following table is a summary of the commands which are available for use only in a VISION:Transact ASL procedure. ASL procedures also allow the use of the partial field built-in function. This is explained in a subsequent section.

Command	Usage
CALL	Calls another procedure (subroutine) or an external user-written module.
CASE	Begins a group of procedure statements within a DO CASE block which are to be performed when the CASE condition is true.
CONTINUE CONT	Continues by going to the end of the procedure.
DO	Either initiates a conditional loop within a procedure or begins a block of procedure statements containing groups of CASE commands. Only the first CASE block within the DO CASE block with a true condition is executed.
ELSE	Begins a group of statements within an IF or DO CASE block which are to be performed when all conditions are false.

Command	Usage
END	Terminates a DO , IF , or DO CASE block.
FIELD FLD	Defines a temporary field within a procedure.
GO	Jumps to a specified statement within a procedure.
IF	Begins a block of procedure statements that are performed when the condition is true on the IF command.
LEAVE	Provides a way to exit a DO loop immediately.
LET	Sets the value of a field equal to the value of a field, constant, or arithmetic expression.
MSG	Terminates the interaction and displays a message when screen field validation fails.
RETURN RET	Returns control to the calling procedure.
SET	Specifies the screen field to be processed by error handling.
TRACE	Displays literals or fields during application execution.
TRANSFER	Causes control to be transferred to a specific point within the processing cycle such as next function, screen control, or output.

Qualifier

The qualifier is a one-character code used to further identify the named field which immediately follows the qualifier. Thus, 1.CUSTNO specifies the customer number field (named CUSTNO) in the data base whose file type on the RF statement is FILE1, while T.CUSTNO specifies the temporary field named CUSTNO. This facility allows you to use the same field name to identify fields in different files or storage locations (e.g., temporary storage, PCB mask, multiple data bases, etc.) since the qualifier differentiates them. A qualifier of blank (no qualifier) or N refers to the data base whose file type is FILE.

Statement Label

In requests, you use the statement label field to label the PR statements. The label can then be used as a branching location (explained below). VISION:Transact processes PR statements in input sequence.

In procedures, a label identifies a specific statement (or command). It is optional at the beginning of a procedure statement and must be immediately followed by a colon (:). The label can then be used as a branching location in a GO command (explained below).

Relational or Logical Expressions

A relational or logical expression in a request is one or more contiguous PR statements, each of which uses one of the relational operators. The connector (position 15) between any of two such statements is assumed to be OR unless you explicitly specify O (or) or A (and). Be aware, however, that the CONREQ keyword in M5PARM can make the connector entry mandatory. (See the *VISION:Transact for CICS Installation Guide* for information on M5PARM.)

Use the logic level in requests for complex expressions. The number in the logic level entry specifies the number of sets of parentheses which would surround the connector in normal Boolean notation. The evaluation of VISION:Transact logical expressions follows the normal Boolean logic rules. That is, expressions connected by AND are grouped before those connected by OR, and expressions between parentheses are grouped from the innermost parentheses to the outermost. Every logical expression in a request is followed by a non-select (NS) or go select (GS) branching instruction; if you do not specify the NS or GS operator, the VISION:Transact compiler automatically inserts NS END, where END is the bottom of this request. (Non-select means: if preceding logical expression is not true, go to the specified branching location.)

In procedures, logical expressions are used in IF, CASE, and DO commands. A logical expression consists of relational expressions or list expressions connected by logical operators. A relational expression compares FIELD A to a constant, field, or arithmetic expression using EQ, NE, LT, GT, LE, or GE operators. A LIST expression lists the specific values to test. There are three Boolean logical operators: AND, OR, and NOT. Parentheses may be used to specify the order of evaluation. For more information and examples, see the *Advantage VISION:Transact ASL Reference Guide*.

An EQ or NE operator in requests or procedures may also reference predefined date or numeric patterns for validation. When used, the value in Operand A or Field A is checked for an equal or not equal condition for the pattern named in Operand B or Field B. The patterns include various date formats as well as numeric, integer and floating point patterns. Refer to the *Advantage VISION:Transact Reference Summary* or *Advantage VISION:Transact ASL Reference Guide* for more details.

Branching or Transfer Locations

In addition to coding an NS or GS instruction after a logical expression in a request, you may specify GO (go to) at any point in your request PR statements. Branching to a statement label within a request must be in forward (top-to-bottom) direction. You may specify branching to any following PR statements as long as it is not in the middle of a logical expression or a continuation of a CALL statement.

In procedures, the GO command can be used to cause a forward branch to a labeled statement in the current procedure. The GO command may only branch forward to a subsequent labeled statement; it may not branch backward to a prior labeled statement. The labeled statement must be in the current procedure.

You may specify branching to the end of a request by specifying END or NEXT REQUEST as a branching location. You may specify branching to the end of a procedure using the CONTINUE command. Type O requests or procedures have an additional branching or transfer-to location of output.

The use of branching location SUB request-name (requests) or CALL command (procedures) causes a branch to a subroutine request or procedure. The use of branching location RETURN (requests) or the RETURN command (procedures) is limited to subroutine requests or procedures; it specifies immediate return to the calling request or procedure. Subroutines are described later in this section.

Branching or transfer-to location SCREEN CONTROL (valid in type 1, 2, M, or S requests or procedures) specifies an immediate branch to the screen control function of the interaction, skipping all other requests and procedures of this type and the automatic functions controlled by the screen processing definition. It also sets the flag field GOSC to 1, 2, or M to indicate the request or procedure type that issued the branch.

Branching or transfer-to location NEXT FUNCTION causes the application to skip all remaining requests and procedures of the same type and resume processing at the next automatic function. If executed in a type O request or procedure, processing stops for the current output screen and the screen is sent to its destination.

In requests only, the branching location BACK specifies a backward branch to the top of the current request, thus initiating a request loop. Request backward branching is described later in this section. Note that ASL procedures can initiate procedure loops through the use of the DO command (refer to the *Advantage VISION:Transact ASL Reference Guide*).

Arithmetic Operations and Expressions

In requests, arithmetic is performed by coding an arithmetic operation (+, -, *, /), which requires the specification of all three PR statement operands. The arithmetic operation is performed on the fields specified in Operands A and B and the result is placed in the field specified in Operand C.

In procedures, arithmetic is performed by coding arithmetic expressions using the operators + (add), - (subtract), * (multiply), and / (divide). For example, $A-B*C+D$. As in conventional algebraic notation, operations within an arithmetic expressions are executed according to a specific hierarchy, basically from left to right. However, multiplication and division are performed prior to addition and subtraction unless this order is overridden by parentheses. For example, $(A-B)*(C+D)$.

Procedures may use arithmetic expressions within logical expressions specified in IF, CASE, and DO commands (e.g., $A \text{ GT } B+C*D$). Arithmetic expressions may also be specified as the source-expression of a LET command (e.g., $\text{LET } A=B+C*D$). For more information, refer to the *Advantage VISION:Transact ASL Reference Guide*.

External Calls

At any point in a request or procedure, you may write a call statement to do the following:

- Make use of an external subroutine.
- Access additional data bases.

In requests only, you perform external calls from a VISION:Transact generated application with either a CALL, PLI, or DLI PR statement. Use the DLI statement to communicate with Data Language/I modules through the VISION:Transact data base interface. For IMS TM only, use the PLI statement to communicate with subroutines written in the PL/I language. Use the CALL statement to communicate with all other programs external to your VISION:Transact generated application. The specifications for writing a call on a VISION:Transact PR statement are in the *Advantage VISION:Transact Reference Summary*.

In ASL procedures, use the CALL command to branch to an external subroutine written in another language or perform your own DL/I calls. The specifications for writing ASL calls are in the *Advantage VISION:Transact ASL Reference Guide*.

Two flag fields, CSTATUS and RETURNCD, are used in conjunction with calls. These flags are discussed in the *Advantage VISION:Transact Reference Summary* or the *Advantage VISION:Transact ASL Reference Guide*.

Set Error

The SE operator (requests) or SET command (procedures) allows you to mark a screen field for error handling while you perform input screen field validation in the request or procedure. Then, when an MS operator (requests) or MSG command (procedures) is issued later in the request or procedure, screen fields thus marked will have their error handling invoked. Each time the set error is executed the flag field ERROR is incremented by one. See [Screen/COMMAREA Field Error Processing in Screen/COMMAREA Requests or Procedures on page 7-33](#) for an example of this.

Message Sending

Often, you need a means of invoking the automatic VISION:Transact error routines. The MS operator (requests) or MSG command (procedures) does the following:

1. It allows you to code your own error message that will be displayed on this input screen's designated error line.
2. It lets you control whether or not data base updates are to be backed out.
3. It invokes the routines to redisplay the input screen and terminate the interaction.

These routines are the same as those invoked when automatic validation (specified on the Sn statements) detects an invalid condition. Thus, consistency is maintained between automatic and procedural validation.

Partial Fielding

Partial fielding allows you to specify processing of only a part of a character string field. This can be accomplished by specifying fixed positions of a field or using the partial field flags. The partial field specifications on the PR statement are used to specify partial fielding in requests. The PF function in ASL is used to specify partial fielding in procedures and is documented in the *Advantage VISION:Transact ASL Reference Guide*.

[Figure 7-2](#) is an example of a request which partial fields specific positions of a screen field. Statement 040 replaces the first position of the screen field ITEMNO into the first position of T,SYNCLFD2. The last six positions of T,SYNCLFD2 are replaced with 9's. The value of T,SYNCLFD1 and T,SYNCLFD2 are used to range browse the ITEM data base. (See [Chapter 8, "Dynamic Synchronization"](#) for details on range browse.)

* REQUEST NAME - MENUFPCK *									
* INPUT STREAM REQUEST *									

STMT	CAT							REQ	SCREEN
TYPE	ACT	REQUEST DESCRIPTION						TYP	NAME

(ER)							(1) (ORDMENU4)		
COMMENT	*****	THIS RQST SAVES THE FUNCTION KEY PRESSED FOR LATER						*****	
COMMENT	*****	VALIDATION BY SCREEN DEF AND SETS UP THE ITEM BROWSE						*****	
COMMENT	*****	WHEN APPROPRIATE						*****	

STMT	FIELD	FIELD	FLD	DEC	OUTPT	EDIT	INITIAL		
TYPE	NAME	LENGTH	TYP	PLC	EDIT	LGTH	VALUE	COMMENTS	

(TF) (SYNCFD1) (7) (C)									
(TF) (SYNCFD2) (7) (C)									

STMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....		RESULT....	PARTIAL FIELD
NUMB	NO.	LEV	CTR	QLF FIELD NAME	ATN	QLF	FIELD NAME OR CONSTANT	QLF FIELD NAME	STRT NUMB
(1)	(010)			(F,ATINID)	(R)			(S,PFKEY)	
(2)	(020)			(F,ATINID)	(EQ)	(C,PF002	)		
(3)	(030)				(R)	(S,ITEMNO	)	(T,SYNCFD1)	
(4)	(040)				(R)	(S,ITEMNO	)	(T,SYNCFD2)	(1) (1) (C)
(5)	(050)				(R)	(C,999999	)	(T,SYNCFD2)	(2) (6) (C)

Figure 7-2 Partial Fielding in Request Processing

Dynamic partial fielding involves setting and/or testing the value of a partial field flag. There are eight flag fields that can be used in partial field operations or functions: PSTART, PNUMBER, LSTART, LNUMBER, MSTART, MNUMBER, RSTART, and RNUMBER. The field names can be abbreviated to PS, PN, LS, LN, MS, MN, RS, and RN, respectively. These fields are four-byte, fixed point, integer fields that are initialized to zero at the start of the interaction.

PR statements (requests), the PF function (procedures), and SC statements may use the two-character abbreviations in the Partial Field Start and Partial Field Length specifications. PR and SC statements as well as procedure statements may also use the flag field names or abbreviations as operands. Dynamic partial field flags cannot be used as entries for Partial Field Start or Partial Field length specifications on S0 or Sn statements; the field names may be used as Operand A or B.

[Figure 7-3](#) is an example of dynamic partial fielding using a type 1 request and a subroutine. In the type 1 request VALIDATE, statement 120 sets the Partial Start flag (PSTART) to 30. Statement 130 replaces the screen field CUSTNAME into T,SCANFLD. The next statement invokes the subroutine SCANLEFT. In the subroutine, statement 100 begins with the leftmost position of T,SCANFLD and checks for a blank. The scan continues until the back branch counter has been decremented 30 times or the PSTART flag equals 0. If the branch to RETURN is taken in statements 120 or 150, processing returns control to the calling request (VALIDATE) at statement 150. Only those positions of S,CUSTNAME that are non-blank are replaced into T,MSG (statement 150) followed by the rest of the message (statement 170).

* REQUEST NAME - VALIDATE *									
* INPUT STREAM REQUEST *									

STMT TYPE	CAT ACT	REQUEST DESCRIPTION					REQ TYP	SCREEN NAME	BACK BR RNT CONTROL TMP
(ER)							(1)		
COMMENT	*****	VALIDATE CUSTNO EQUAL TO '2' IN THE FIRST POSITION					*****		
COMMENT	*****	PLACE CUSTNAME IN MESSAGE AFTER TRUNCATING BLANKS					*****		
STMT TYPE	FIELD NAME	FIELD LENGTH	FLD TYP	DEC PLC	OUTPT EDIT	EDIT LGTH	INITIAL VALUE	COMMENTS	
(TF)	(SCANFLD)	(30)	(C)						
(TF)	(MSG)	(74)	(C)						
STMT NUMB	SEQ NO.	LOG LEV	CON CTR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR
(1)	(100)			(S,CUSTNO)	(EQ)	(C,2)		(1)	(1) (A)
(2)	(110)				(NS)	(END)			
(3)	(120)				(R)	(D,30)	(F,PSTART)		
(4)	(130)				(R)	(S,CUSTNAME)	(T,SCANFLD)		
(5)	(140)				(GO)	(SUB SCANLEFT)			
(6)	(150)				(R)	(S,CUSTNAME)	(T,MSG)	(1)	(PS) (C)
(7)	(160)			(F,PSTART)	(+)	(D,1)	(F,PSTART)		
(8)	(170)				(R)	(C,CUSTNO CANNOT BE 2)	(T,MSG)	(PS) (32)	(C)

* REQUEST NAME - SCANLEFT *									
* INPUT STREAM REQUEST *									

STMT TYPE	CAT ACT	REQUEST DESCRIPTION					REQ TYP	SCREEN NAME	BACK BR RNT CONTROL TMP
(ER)							(S)		(30)
COMMENT	*****	SCAN FOR THE FIRST BLANK POSITION					*****		
STMT NUMB	SEQ NO.	LOG LEV	CON CTR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR
(1)	(100)			(T,SCANFLD)	(EQ)	(C)		(PS) (1)	(A)
(2)	(120)				(NS)	(RETURN)			
(3)	(130)			(F,PSTART)	(-)	(D,1)	(F,PSTART)		
(4)	(140)			(F,PSTART)	(EQ)	(D,0)			
(5)	(150)				(GS)	(RETURN)			
(6)	(150)				(GO)	(BACK)			

Figure 7-3 Dynamic Partial Fielding Using a Type 1 Request and a Subroutine

Processing Recurring (Lower Level/Subordinate) Segments

The first *executed* reference to a field in a subordinate segment of the intermediate structure automatically starts a loop whose span is from that reference to the end of the request or procedure. This eliminates much of the procedural code which is usually necessary when processing subordinate segments.

Components of a Request or Procedure

Request Components

The request components and event sequence is visually presented in [Figure 7-4](#).

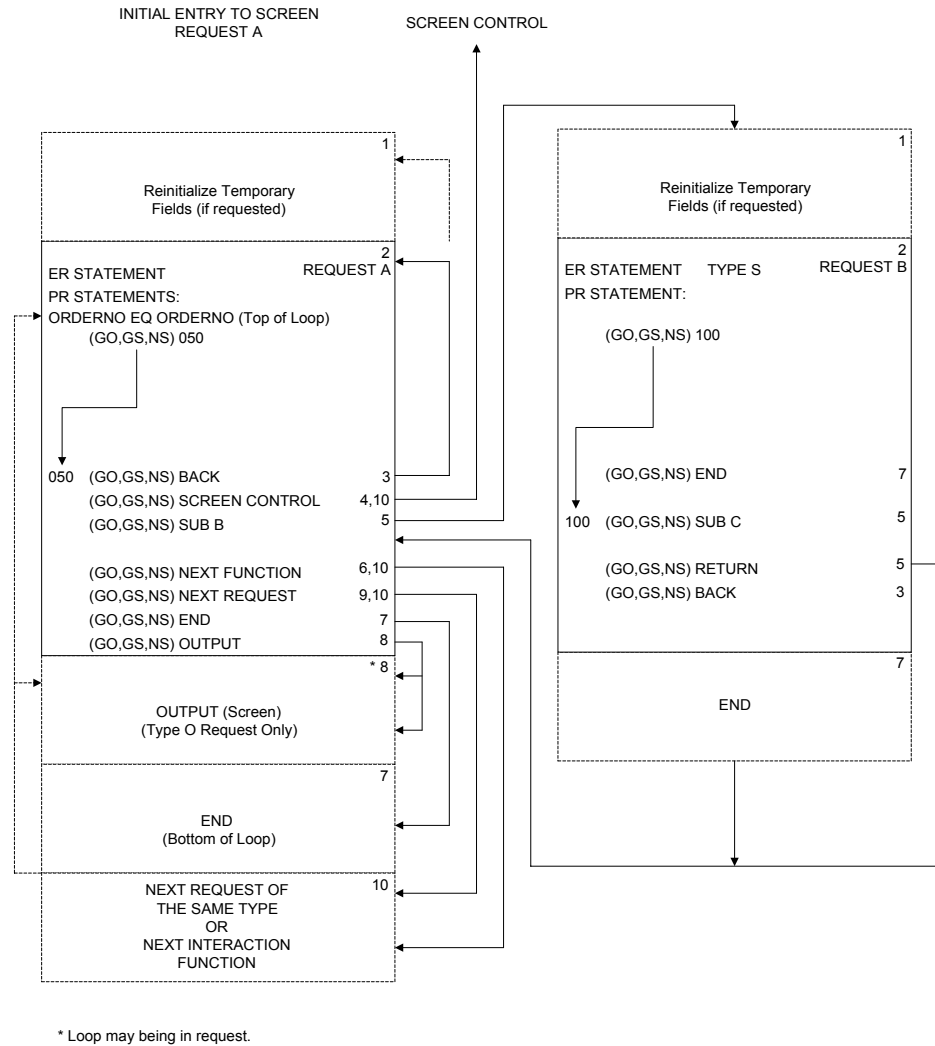


Figure 7-4 Branching Operators and Flow of Control in a Request

In the figure, the parts of the request outlined with dashes represent functions performed automatically by your generated application.

The sequence of events depicted in [Figure 7-4](#) occurs as follows:

1. If you specified that temporary fields are to be reinitialized, they are re-initialized first.

2. Your request logic, as defined in PR statements, now executes. Screen synchronized data base (SSDB) field references in PR statements address the intermediate structure (see [The Intermediate Structure on page 6-13](#)).
3. Branching to location BACK causes a backward branch to the top of the current request, thus creating a request loop. The PR statements within the loop execute until either your PR logic directs a branch out of the loop or the back branch limit (specified on the ER statement) is reached. When this limit is reached processing continues at the statement following the back branch.
4. Branching to location SCREEN CONTROL causes an immediate branch to the screen control function of the interaction. All other requests and procedures of the same type and automatic functions of the screen processing definition are bypassed.
5. Branching to location SUB request-name invokes the subroutine request (or procedure) named request-name. A subroutine inherits loops on segment occurrences from the request or procedure that invokes it. During subroutine execution, more loops may be started but, upon return to the invoking request or procedure, any looping started by the subroutine is terminated and only those loops that were in progress when the subroutine was called will continue. When the subroutine completes processing or when it branches to location RETURN, control returns to the invoking request or procedure at the statement following the branch.
6. Branching location NEXT FUNCTION causes the application to bypass remaining requests and procedures of the same type and resume processing at the next automatic function. If executed in a type O request, processing stops for the current output screen and the screen is sent to its destination.
7. The END logic determines whether all loops have been exhausted. These loops may have been started by request logic, Step 2 above, or output screen processing, Step 8. If any loop is still in progress, the application gets the next occurrence of the segment being looped on, and returns to the point at which the loop began.

Note: Loops begun in output screen processing always loop back to the beginning of output screen processing.

8. Branching to the location OUTPUT, or merely completing the Type O request execution without branching to END, causes the formatting of an occurrence of all screen segments from the current occurrence of the named fields. Loops may be started by the output screen formatting processes at the end of your Type O request. This looping is based on the source field, not on the output screen structure.
9. The output looping will end when (1) no more occurrences of the segment being looped on are available, (2) a NEXT REQUEST or NEXT FUNCTION branch is executed within the output request, or (3) the output screen occurrences have been filled and an A was not specified for Output Control on the SD statement.
10. The cleanup logic performs housekeeping to ensure the modularity of the next request or procedure of the same type or the next interaction function. The

next request/procedure or interaction function will operate independently upon the intermediate structure, initiating its own actions, including any fresh looping which may be begun. Branching to the location NEXT REQUEST, NEXT FUNCTION, or SCREEN CONTROL invokes this cleanup logic immediately, bypassing the END logic in Step 7.

Procedure Components

The procedure components and event sequences are visually presented in [Figure 7-5](#).

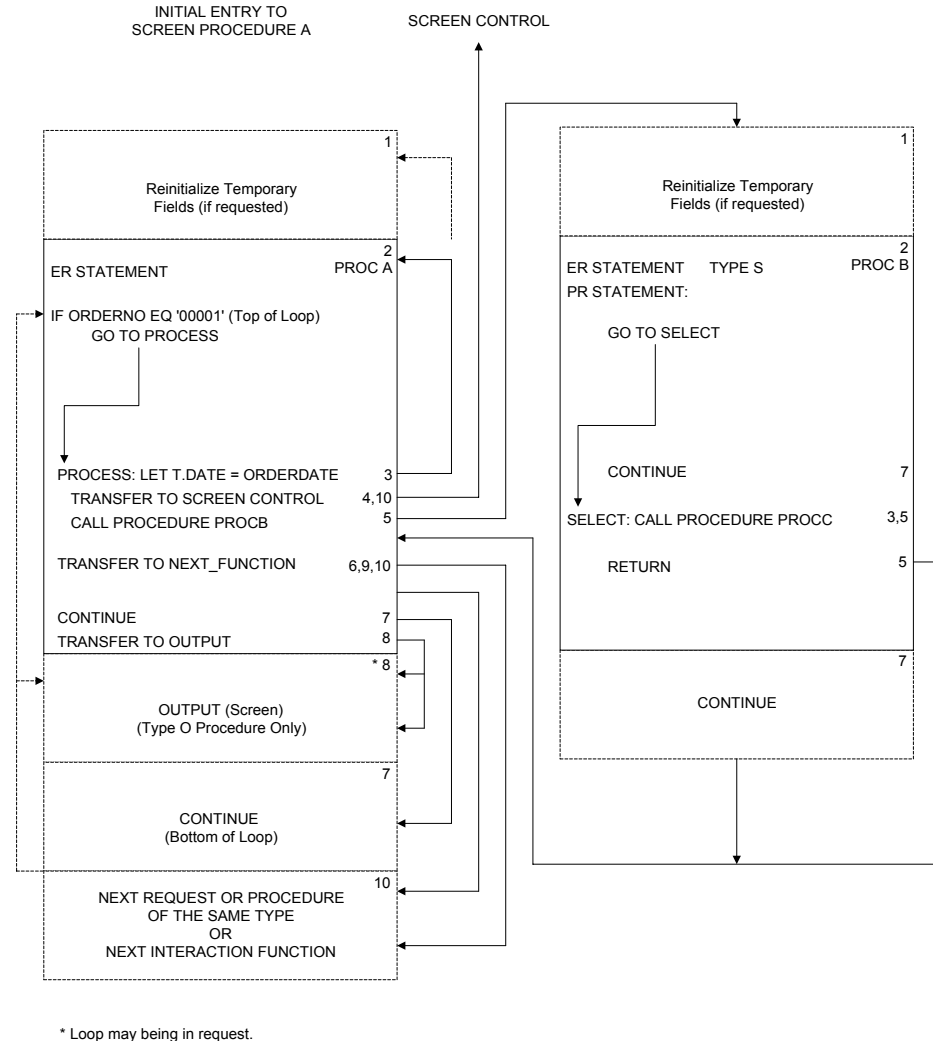


Figure 7-5 Branching Operators and Flow of Control in a Procedure

In the figure, the parts of the procedure outlined with dashes represent functions performed automatically by your generated application.

The sequence of events depicted in [Figure 7-5](#) occurs as follows:

1. If you specified that temporary fields are to be reinitialized, they are re-initialized first.

2. Your procedure logic, as defined by ASL statements, now executes. Screen synchronized data base (SSDB) field references in procedure statements address the intermediate structure (see [The Intermediate Structure on page 6-13](#)).
3. A GO [TO] command branches forward to a subsequent labeled statement in the current procedure.
4. Transferring to location SCREEN CONTROL causes an immediate branch to the screen control function of the interaction. All other requests and procedures of the same type and automatic functions of the screen processing definition are bypassed.
5. Calling a subroutine request or procedure invokes the subroutine named in the CALL statement. A subroutine inherits loops on segment occurrences from the request or procedure that invokes it. During subroutine execution, more loops may be started but, upon return to the invoking request or procedure, any looping started by the subroutine is terminated and only those loops that were in progress when the subroutine was called will continue. When the subroutine completes processing or when it transfers to RETURN, control returns to the invoking request or procedure at the statement following the branch.
6. Transferring to location NEXT FUNCTION causes the application to bypass remaining requests and procedures of the same type and resume processing at the next automatic function. If executed in a type O procedure, processing stops for the current output screen and the screen is sent to its destination.
7. The CONTINUE logic determines whether all loops have been exhausted. These loops may have been started by procedure logic, Step 2 above, or output screen processing, Step 8. If any loop is still in progress, the application gets the next occurrence of the segment being looped on, and returns to the point at which the loop began.

Note: Loops begun in output screen processing always loop back to the beginning of output screen processing.

8. Transferring to the location OUTPUT, or merely completing the type O procedure execution without issuing a CONTINUE command, causes the formatting of an occurrence of all screen segments from the current occurrence of the named fields. Loops may be started by the output screen formatting processes at the end of your type O procedure. This looping is based on the source field, not on the output screen structure.
9. The output looping will end when (1) no more occurrences of the segment being looped on are available, (2) a transfer to NEXT FUNCTION is executed within the output procedure, or (3) the output screen occurrences have been filled and an A was not specified for Output Control on the SD statement.
10. The cleanup logic performs housekeeping to ensure the modularity of the next request or procedure of the same type or the next interaction function. The next request/procedure or interaction function will operate independently upon the intermediate structure, initiating its own actions, including any fresh looping which may be begun. Transferring to the location NEXT FUNCTION

or SCREEN CONTROL invokes this cleanup logic immediately, bypassing the CONTINUE logic in Step 7.

Request/Procedure Types and Data Availability

The screen requests or procedures are type 1, type 2, type M, and type O. [Figure 7-5](#) shows when each screen request or procedure type extends an interaction. Each screen request or procedure is only executed when the input or output screen of the interaction is its assigned screen. While multiple type 1, type 2, and type M screen requests and procedures can be assigned to the input screen, only one type O screen request or procedure can be assigned to a single output screen.

The VISION:Transact extended interaction cycle is composed of the primary events shown in [Figure 7-6](#).

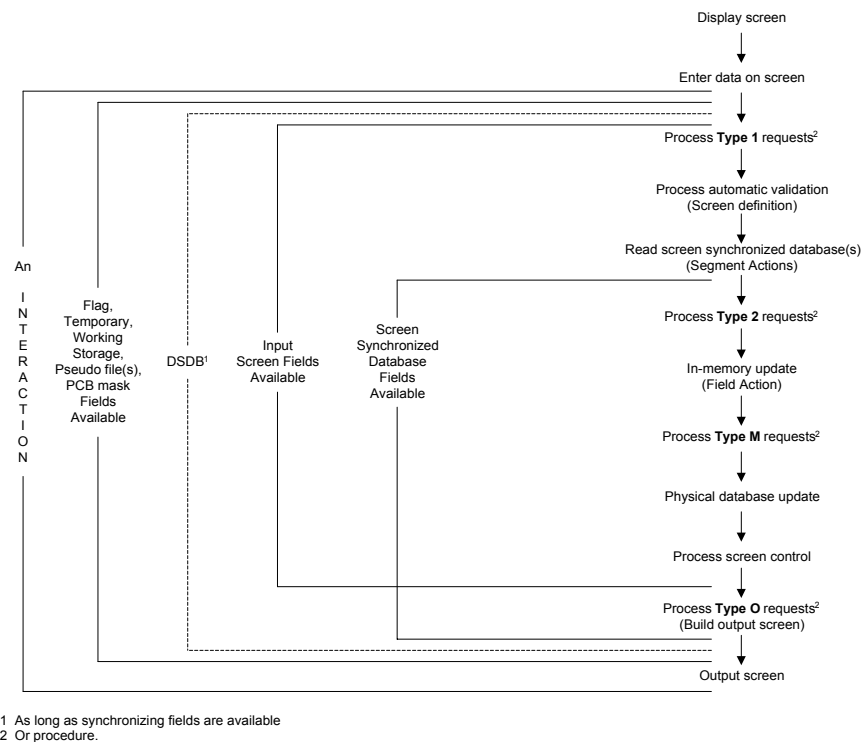


Figure 7-6 The Interaction

[Figure 7-7](#) shows where screen requests/procedures fit into the interaction cycle introduced in [Chapter 4, “Application Flow”](#).

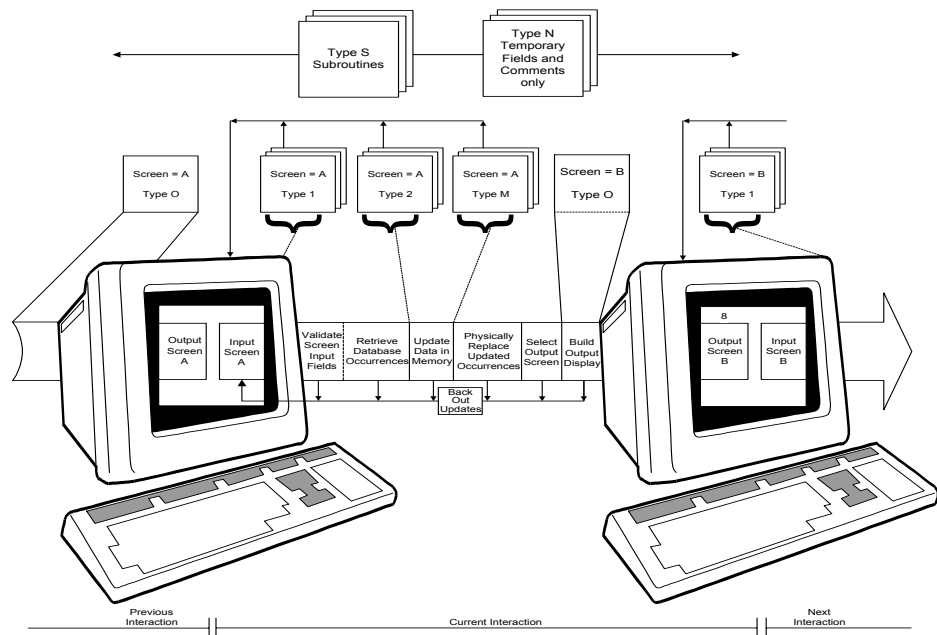


Figure 7-7 Extended Interaction Cycle

Execution tracing, described in [Chapter 12, “Execution Tracing”](#), can be used to output trace information concerning the execution of your requests and procedures such as when they are entered and exited.

Now let's examine screen requests and procedures in detail, using samples of each request type from the VISION:Transact generated application system described in [Chapter 2, “A VISION:Transact Generated Application System”](#), whose complete source code is in [Appendix A, “Order/Item Application Screen and Database Definition Code”](#).

Type 1 Requests/Procedures

Type 1 requests and procedures are executed after successful screen input and initial program start-up processing such as screen name validation, PCB selection (in applications that reference IMS data bases), saved working storage and/or saved pseudo file retrieval (IMS TM - SPA, CICS - Temporary Storage), and input screen size verification. As a result they can manipulate and validate fields on the screen prior to the automatic validations and defaults specified in the input screen's definition. The following fields can be referenced in a type 1 request or procedure:

1. Screen fields (using an S qualifier).

2. Temporary fields (using a T qualifier).
3. Working storage fields (using a W qualifier).
4. Flag fields (using an F qualifier).
5. PCB mask fields (using a V qualifier).
6. Dynamically synchronized data base (DSDB) fields that are synchronized against fields listed in (1) through (5) or (7) (using the qualifier of the DSDB). Dynamic synchronization is explained in [Chapter 8, “Dynamic Synchronization”](#).
7. Pseudo file fields (using the qualifier of the pseudo file).

Type 1 requests and procedures cannot reference SSDB fields since, as shown in [Figure 7-5](#), a VISION:Transact application does not read from those data bases until after type 1 requests and procedures have executed.

At type 1 time, the input screen data has been read into memory. While the actions specified in the screen processing definition have not yet been performed, any de-editing you specified on input screen fields has been completed. Only fields referenced with an S qualifier reflect data entered on the input screen. Also, default values specified on the screen processing definition have not yet been provided.

Type 1 requests/procedures can procedurally validate input screen fields. They can also assist you by displaying your own application-specific error messages coded via the MS operator (requests) or MSG command (procedures).

A type 1 request or procedure can also be used to initialize/load a pseudo file record (see [Chapter 10, “Defining Data”](#)).

[Figure 7-8](#) shows a type 1 request written using fixed form statements.

```

*****
* REQUEST NAME - OURINI *
* INPUT STREAM REQUEST *
*****

-----
STMT CAT                                REQ SCREEN BACK BR RNT
TYPE ACT REQUEST DESCRIPTION          TYP NAME CONTROL TMP

-----
(ER)                                     (1) (ORDUPDT4)
COMMENT ***** THIS REQUEST VALIDATES THE FUNCTION KEY PRESSED *****
COMMENT ***** BY THE TERMINAL OPERATOR *****

-----
STMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD
NUMB NO. LEV CTR QLF FIELD NAME ATN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OPR

-----
( 1) (010) (F,ATTNID ) (EQ) (C,ENTER )
( 2) (020) (O) (F,ATTNID ) (EQ) (C,PF001 )
( 3) (030) (O) (F,ATTNID ) (EQ) (C,PF010 )
( 4) (040) (GS) (060 )
( 5) (050) (MS) (C,INVALID KEY PRESSED - RETRY )
COMMENT ***** *****
COMMENT ***** IF PF1 OR PF10 HAVE BEEN PRESSED, MAKE SURE *****
COMMENT ***** ACTION IS BLANK SO THAT ITEMORD SEGMENT IS *****
COMMENT ***** REMOVED FROM THE INTERMEDIATE STRUCTURE *****
COMMENT ***** *****
( 6) (060) (F,ATTNID ) (NE) (C,ENTER )
( 7) (070) (R ) (C,S ) (S,ACTION )

```

Figure 7-8 A Type 1 Request

This type 1 request for the ORDUPDT4 screen has two functions. First, in statements 010-050, the request validates the function key pressed by the terminal operator, checking the ATTNID flag (F,ATTNID) for the values which trigger segment actions in the screen processing definition for ORDUPDT4. If a wrong key has been pressed, statement 050 redisplay the input screen with an appropriate error message. If an acceptable key has been pressed, statement 040 branches to statement 060, where a screen field is cleared (by replacing c, b- a blank character string-into S,ACTION) if something other than the Enter key was pressed. You can combine multiple functions such as these in a single input screen request, or you can split the functions into different requests in a more modular fashion.

Type 2 Requests/Procedures

Type 2 requests and procedures execute after type 1 requests and procedures and validation specified in the screen processing definition is performed and after all SSDBs are read based on segment actions. Type 2 requests/procedures execute before field actions are performed. The following fields can be referenced by type 2 requests or procedures:

1. Screen fields (using an S qualifier).
2. SSDB fields (using the qualifier of the SSDB).
3. Temporary fields (using a T qualifier).
4. Working storage fields (using a W qualifier).
5. Flag fields (using an F qualifier).
6. PCB mask fields (using a V qualifier).
7. DSDB fields (using the qualifier of the DSDB).
8. Pseudo file fields (using the qualifier of the pseudo file).

Unlike type 1 screen requests or procedures, type 2 requests or procedures can reference SSDB fields since those data bases have now been read (refer to [Figure 7-6](#)). Type 2 requests or procedures are very useful for verifying that the proper data base segments are available to the remainder of the interaction and to validate non-key screen fields. Type 2 requests or procedures can use the MS operator (requests) or MSG command (procedures) to send diagnostic messages to the terminal user, abort the update, and terminate the interaction.

At type 2 time, destination fields specified by the input screen's definition do not reflect the entries made on the input screen; field actions have not been performed. As in type 1 requests/procedures, temporary, working storage, and SSDB fields are not yet updated by input screen entries. You can however, reference the screen fields themselves to access data entered on the input screen. At this point, SSDB segments which will be deleted are still available while segments which will be inserted are not yet available, and are considered missing (unequal in compares). At type 2 time, any default values specified in the screen processing definition will have been applied to screen fields for which no entry was made on the screen.

Figure 7-9 shows a type 2 request written using fixed form statements.

```

*****
* REQUEST NAME - OUR2N2 *
* INPUT STREAM REQUEST *
*****

STMT CAT                                REQ SCREEN BACK ER RNT
TYPE ACT REQUEST DESCRIPTION            TYP NAME CONTROL TMP

(ER)                                     (2) (ORDUPDT4)
COMMENT ***** THIS REQUEST CONTROLS THE UPDATE OF THE QUANTITY-ON-HAND *****
COMMENT ***** FIELD IN THE ITEM DATA BASE. *****

STMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD
NUMB NO. LEV CTR QLF FIELD NAME AIN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OPR

( 1) (010)          (F,ATTNID ) (EQ) (C,ENTER )
( 2) (015)          (A) (S,ACTION ) (EQ) (C,U )
( 3) (020)          (NS) (END )
COMMENT *****
COMMENT ***** IS THIS ITEM ALREADY ON THIS ORDER? *****
COMMENT *****
( 4) (030)          (ITEMORD ) (EQ) (ITEMORD )
( 5) (040)          (NS) (100 )
COMMENT *****
COMMENT ***** IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED? *****
COMMENT *****
( 6) (050)          (S,ITMQTYOR) (EQ) (ITMQTYOR )
( 7) (060)          (GS) (END )
COMMENT *****
COMMENT ***** IF QUANTITY ORDERED HAS BEEN CHANGED, ADD BACK THE *****
COMMENT ***** OLD AMOUNT BEFORE SUBTRACTING THE NEW AMOUNT *****
COMMENT *****
( 8) (090)          (ITMQTYOR) (+) (1,QTYONHND ) (1,QANONHND)
( 9) (100)          (1,QANONHND) (-) (S,ITMQTYOR ) (T,QANONHND)
(10) (110)          (T,QANONHND) (GE) (D,0 )
(11) (120)          (GS) (190 )
(12) (125)          (ITEMORD ) (EQ) (ITEMORD )
(13) (126)          (NS) (140 )
(14) (130)          (1,QANONHND) (-) (ITMQTYOR ) (1,QANONHND)
(15) (140)          (T,QANONHND) (*) (D,-1 ) (T,QANONHND)
(16) (150)          (R ) (C,QUANTITY ORDERED IS ) (T,MSG ) (1) (19) (C)
(17) (160)          (R ) (T,QTYONHND ) (T,MSG ) (21) (07) (C)
(18) (170)          (R ) (C,MORE THAN QUANTITY ON HAND ) (T,MSG ) (31) (31) (C)
(19) (175)          (S,ITMQTYOR) (SE)
(20) (180)          (MS) (T,MSG )
COMMENT *****
COMMENT ***** STATEMENT 190 (AS WELL AS 090 AND 130 ABOVE) *****
COMMENT ***** UPDATES THE ITEM DATA BASE PROCEDURALLY. *****
COMMENT *****
( 21) (190)          (R ) (T,QANONHND ) (1,QANONHND)

```

Figure 7-9 A Type 2 Request

The request in [Figure 7-9](#), which is executed when the ORDUPDT4 screen is input to an interaction, reflects many of the characteristics of type 2 processing, including:

- The state of the intermediate structure. The ORDUPDT4 screen processing definition specifies a V action on the ITEMORD segment of the CUSTOMER data base, meaning that the terminal operator may be updating an existing ITEMORD segment, or he may be inserting a new one. Statements 030/040 and 125/126 cause alternative processing based on this distinction. Comparing a field (ITEMORD) to itself, as in these statements, is a common VISION:Transact coding technique to check for the existence of a segment. If the data base segment is being inserted, it will not exist in the composite segment at type 2 time, prior to the in-memory update. When the segment is missing from the composite segment, the NS branch will be taken, bypassing some processing statements which are executed only for the update of existing ITEMORD segments.

- The fact that field updates haven't happened yet. Statements 050/060 compare the screen field S,ITMQTYOR to the data base field ITMQTYOR to see if the quantity ordered for this item, which was already on the order (determined above in statements 030/040), has been changed. If the quantity ordered has not been changed, the remainder of the request is bypassed by statement 060.
- Your ability to perform error processing. Statements 090/120 determine whether there is enough quantity-on-hand to fill the order for this time. If there is not, statements 150/180 build an error message, MARK the screen field S,ITMQTYOR for error handling, and redisplay the input screen.
- Your ability to update SSDB fields in screen requests. Statements 090, 130, and 190 update the quantity-on-hand field 1,QANONHND by adding, subtracting, or replacing other fields into 1,QANONHND.

Type M Requests/Procedures

Type M requests or procedures are executed after type 2 requests/procedures and after the application updates the SSDB(s) and other destinations in memory (all field actions), but before the SSDB updates are actually written to the data base(s). The following fields can be referenced by type M requests or procedures:

1. Screen fields (using an S qualifier).
2. SSDB fields (using the qualifier of the SSDB).
3. Temporary fields (using a T qualifier).
4. Working storage fields (using a W qualifier).
5. Flag fields (using an F qualifier).
6. PCB mask fields (using V qualifier).
7. DSDB fields (using the qualifier of the DSDB).
8. Pseudo file fields (using the qualifier of the pseudo file).

Though type M requests or procedures can reference SSDB fields as can type 2 requests/procedures, at type M time all fields reflect the input screen updates. In this way type M requests or procedures are useful to perform additional verification on the updated segments before they are actually written to the data base. At this point, deleted segments are no longer available (flagged as missing) while inserted segments are available.

Type M requests or procedures can use the MS operator (requests) or MSG command (procedures) to re-display the input screen with an error message back to the terminal and terminate the interaction. Since the interaction is terminated via the MS operator or MSG command, the in-memory updates are nullified and as a result, type M screen requests or procedures are useful in preventing improper updates from actually reaching the screen synchronized data bases.

[Figure 7-10](#) shows a type M request written using fixed form statements.

* REQUEST NAME - FILLINV *									
* INPUT STREAM REQUEST *									

STMT	CAT					REQ	SCREEN	BACK	BR RNT
TYPE	ACT	REQUEST DESCRIPTION					TYP	NAME	CONTROL TMP

(ER)					(M) (FILEBKOD4)				
COMMENT	*****	THIS REQUEST SUPPLIES THE INVOICE DATE AND AMOUNT.					*****		

STMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....	...RESULT...	PARTIAL FIELD	
NUMB	NO.	LEV	CTR	QLF FIELD NAME	ATN	QLF FIELD NAME OR CONSTANT	QLF FIELD NAME	STRT	NUMB OPR

(	1)	(010)		(F,ATINID)	(EQ)	(C,ENTER			)
(	2)	(020)	(A)	(S,CHOICE)	(EQ)	(C,Y			)
(	3)	(030)			(NS)	(END			)
(	4)	(040)			(R)	(F,TODAY		(INDATE)	
(	5)	(050)		(S,ITMPRICE)	(*)	(S,QTYBKORD		(INAMT)	

Figure 7-10 A Type M Request

The request in [Figure 7-10](#) takes advantage of the availability of newly inserted segments at type M time. The FLLBKOD4 screen inserts a SHIPINV segment occurrence into the customer data base when the terminal operator presses the Enter key after entering a Y in the screen field S,CHOICE. The FILLINV request provides the invoice date and amount following the in-memory insertion of the new SHIPINV segment into the intermediate structure.

The Type O Request/Procedure

A type O request or procedure is executed after screen control has selected the output screen. The type O request or procedure is defined against the output screen. Unlike the other types of screen requests/procedures, the type O request or procedure usually involves selection calculation or editing for display rather than validation and verification. The following fields can be referenced by a type O request or procedure:

- 1. SSDB fields (using the qualifier of the SSDB).
- 2. Temporary fields (using a T qualifier).
- 3. Working storage fields (using a W qualifier).
- 4. Flag fields (using an F qualifier).

5. PCB mask fields (using a V qualifier).
6. DSDB fields (using the qualifier of the DSDB).
7. Pseudo file fields (using the qualifier of the pseudo file).

Type O requests/procedures cannot reference screen fields. They may include an MS operator (requests) or MSG command (procedures) which re-displays the input screen with a message and specifies whether or not data base updates should be backed out. The SE operator (requests) or SET command (procedures) may not be used since screen fields are not allowed.

A subroutine request or procedure that references a screen field is invalid if invoked by a type O request or procedure.

[Figure 7-11](#) shows the unique relationship between the type O request or procedure and the output screen. There can be only one type O request or procedure for a given screen because type O executes as part of output screen formatting. Type O requests or procedures may branch directly to screen formatting by specifying the keyword, OUTPUT, as a branch-to (requests) or transfer-to (procedures) location. Thus a type O request can select specific segment occurrences to be displayed by branching/transferring to OUTPUT to display this occurrence, or to END (requests)/CONTINUE (procedures) to bypass the output of this occurrence.

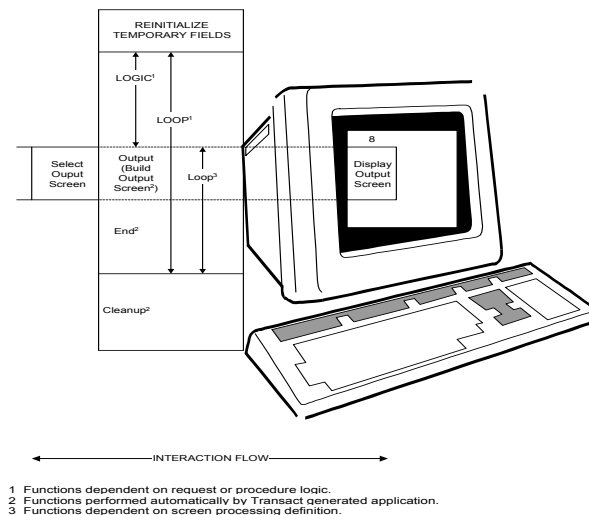


Figure 7-11 The Relationship Between a Type O Request/Procedure and the Output Screen

[Figure 7-12](#) shows a type O request written using fixed form statements.

```

*****
* REQUEST NAME - SCRNSEL *
* INPUT STREAM REQUEST *
*****

```

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK CONTROL	BR RNT	TIME
(ER)				(O) (ITMBKOD4)			
COMMENT	*****	THIS REQUEST SELECTS THE DESIRED OCCURRENCES OF BACK-	*****				
COMMENT	*****	ORDERS FOR SCREEN DISPLAY BASED ON T,SCREENNO.	*****				

STMT NUMB	SEQ NO.	LOG LEV	CON CTR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR
(	1)	(010)			(R)	(D,0	)	(T,COUNTOCC)	
(	2)	(020)		(T,SCREENNO)	(*)	(D,8	)	(T,LASTOCC)	
(	3)	(030)		(T,LASTOCC)	(-)	(D,7	)	(T,FIRSTOCC)	
(	4)	(040)		(1,BKORDRNO)	(EQ)	(1,BKORDRNO	)		
(	5)	(045)			(NS)	(OUTPUT	)		
(	6)	(050)		(T,COUNTOCC)	(+)	(D,1	)	(T,COUNTOCC)	
(	7)	(060)		(T,COUNTOCC)	(GE)	(T,FIRSTOCC	)		
(	8)	(070)			(NS)	(END	)		

Figure 7-12 A Type O Request

The ITMBKOD4 screen is a structured display of data from the item data base. Data from the root data base segment is displayed in the screen root, and data from as many as eight occurrences of the lower level BKORDERS segment is displayed in the second level of the structured screen. A paging capability is designed into the processing of the ITMBKOD4 screen, so that the terminal user can move his display forward and backward through occurrences of the BKORDERS segment, eight at a time.

The type O request in [Figure 7-12](#) selects appropriate occurrences of the BKORDERS segment for the ITMBKOD4 screen display, using the value in the temporary field T,SCREENNO. T,SCREENNO is passed from the input screen processing of the interaction. It contains either a 1, if the input screen to this interaction was some screen other than ITMBKOD4, or a selected screen number, if ITMBKOD4 was the input screen to this interaction. Statements 010/030 calculate the first occurrence to be displayed on the screen.

Statement 040 brings the BKORDERS segment into the intermediate structure. [Extending the Intermediate Structure on page 7-40](#) discusses extending the intermediate structure in this manner. Depending on the input screen to the interaction, the BKORDERS segment either has not been referenced previously, or it had an X segment action applied to all occurrences processed by the input screen. Since statement 040 brings this lower level segment into the intermediate structure, it causes the retrieval of all BKORDERS segments subordinate to the ITEMROOT (item) segment that was synchronized to the input screen of the interaction.

The remainder of the request is contained in the loop begun by statement 040, including the OUTPUT portion in which the output screen content of the ITMBKOD4 screen processing definition is built. Statements 050/070 increment a counter, T,COUNTOCC, by one for every occurrence of the BKORDERS segment. When that counter is greater than or equal to the calculated first occurrence to be displayed, the occurrence is selected for display. When the counter value is less than the first occurrence to be displayed, the occurrence is bypassed, because the branch to END is taken.

NS END is the default conditional branch assumed for any logical expression, and is explicitly coded in statement 070 only to clarify the processing performed by this type O request.

Statement 045, NS OUTPUT, is indicative of the third characteristic of request processing associated with [Figure 7-4](#). The NS branch will only be taken if the field 1,BKORDRNO is missing. Since statement 040 brings the BKORDERS segment into the intermediate structure, this missing condition can only occur if there are no occurrences of the BKORDERS segment (for its parent ITEMROOT segment, which is already synchronized to the intermediate structure) to process. When this condition occurs, no request loop is started and the NS OUTPUT branch causes the OUTPUT processing to execute once. The output screen content will contain data from the ITEMROOT segment and hyphens (-) where data from fields in the BKORDERS segment would ordinarily have displayed. If the output processing was not executed at least once (say, if statement 045 was not coded at all and the implicit NS END branch was executed instead), the output screen content would contain literal data only; no field data would appear on the output display.

The Subroutine (Type S) Request/Procedure

A subroutine (or type S) request or procedure is called by another request or procedure to perform a set of functions. A subroutine is invoked by using the branching operators GO, GS, or NS with Operand B consisting of branching location SUB request name (requests) or by the CALL PROCEDURE command (procedures). A subroutine returns control to the calling request or procedure at the statement following the GO/NS/GS SUB or CALL PROCEDURE command. Subroutine requests or procedures are not associated with a specific screen. You define a subroutine on the ER statement by entering an S in the Type specification and leaving the Screen Name specification blank. Such a request or procedure is not invoked automatically by VISION:Transact; it is invoked only from another request or procedure.

An individual subroutine may be invoked by any number of requests/procedures within an application and by any type, including another subroutine (up to 16 levels of nested subroutines are allowed). However, recursive invocation of a subroutine is not permitted; i.e., no subroutine may be specified which could invoke itself with a GO, GS, or NS SUB operation (requests) or CALL PROCEDURE command (procedures) either directly or indirectly. For example, subroutine A may not invoke subroutine B if subroutine B could also invoke subroutine A.

If the GO, GS, or NS SUB statement (requests) or CALL PROCEDURE command (procedures) is the last statement in a request or procedure, then, upon return from the subroutine, normal end-of-request/procedure processing is performed. Implicit return from a subroutine occurs when the subroutine is through processing. Explicit return from a subroutine is specified by coding GO, GS, or NS RETURN (requests) or the RETURN command (procedures) to force immediate return to the invoking request or procedure. RETURN is valid only in a subroutine request or procedure.

A subroutine inherits loops on segment occurrences from the invoking request or procedure. During subroutine execution, more loops may be started but any looping started by the subroutine is terminated upon return to the invoking request or procedure, and only those loops that were in progress in the calling request or procedure will continue.

The fields that can be referenced in a subroutine depend on the type of the calling request or procedure. For example, if a type 1 request or procedure calls a subroutine, the subroutine cannot reference SSDB fields as they are not available to type 1 requests/procedures.

Figure 7-13 shows a type S request written using fixed form statements.

```

*****
* REQUEST NAME - ORDRTOTL *
* INPUT STREAM REQUEST *
*****

```

SIMT	CAT	REQ	SCREEN	BACK	BR	RNT
TYPE	ACT	TYP	NAME	CONTROL	TMP	
(ER)						(S)
COMMENT *****	THIS REQUEST COMPUTES THE TOTAL COST OF THE ITEMS	*****				
COMMENT *****	ON AN ORDER.	*****				

SIMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD
NUMB	NO.	LEV	CTR	QLF FIELD NAME	ATN	QLF FIELD NAME OR CONSTANT	QLF FIELD NAME	SIRT NUMB OPR
(1)	(010)				(R)	(D,0	)	(T,ORDERTOT)
(2)	(020)			(ITMPRICE)	(*)	(ITMPTYOR	)	(T,ITEMTOT)
(3)	(030)			(T,ORDERTOT)	(*)	(T,ITEMTOT	)	(T,ORDERTOT)

```

*****
* REQUEST NAME - CALLSUB *
* INPUT STREAM REQUEST *
*****

```

SIMT	CAT	REQ	SCREEN	BACK	BR	RNT
TYPE	ACT	TYP	NAME	CONTROL	TMP	
(ER)						(O) (ORDUPDT4)
COMMENT *****	SELECT AN ORDER AND CALL SUBROUTINE ORDRTOTL	*****				
COMMENT *****	TO COMPUTE COST OF THE ITEMS FOR THAT ORDER.	*****				

SIMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD
NUMB	NO.	LEV	CTR	QLF FIELD NAME	ATN	QLF FIELD NAME OR CONSTANT	QLF FIELD NAME	SIRT NUMB OPR
(1)				(F,ATINID)	(NE)	(C,PF001	)	
(2)		(A)		(T,SELORDR)	(EQ)	(C,X	)	
(3)					(R)	(ORDERNO	)	(T,ORDERSEL)
(4)					(GO)	(SUB ORDRTOTL	)	

Figure 7-13 A Type S Request and Calling Request

This subroutine request is invoked at type O time by request CALLSUB. At this point in the interaction, the CUSTOMER and ORDER segments are in the root of the intermediate structure. The request CALLSUB determines the correct order, replaces that order number into T,ORDERSEL and calls the subroutine ORDRTOTL. This subroutine starts a loop on the ITEMORD segment to compute a total price for the order in temporary field ORDERTOT. This field is then displayed in the root segment of the output screen ORDUPDT4.

Request Back Branching

The back branching capability, available in requests only, allows you to branch back to the beginning of the current request. The Back Branch Limit specification on the ER statement indicates the number of back branches allowed. This specification is required to eliminate the possibility of getting into an endless loop. [Figure 7-14](#) shows how the request is executed when back branching is specified.

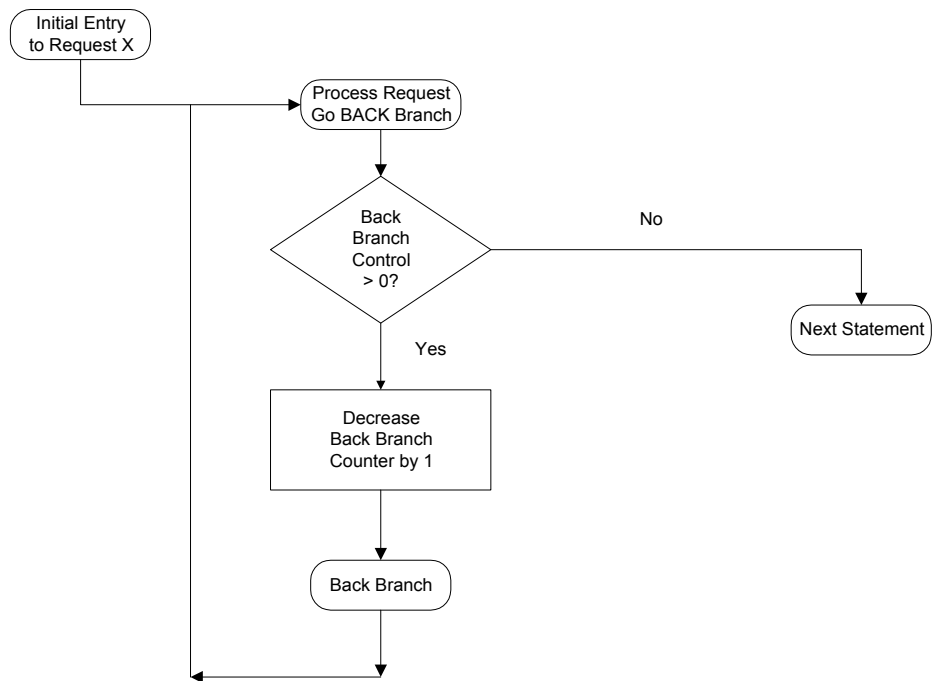


Figure 7-14 Back Branching Flow

The first time the request is executed, the back branch limit counter is set to either zero or the number specified in Back Branch Limit. Each time the request is executed, the counter is reduced by one until it reaches zero, at which time no further back branching is allowed and control passes to the next statement in the request. If the counter was zero to begin with, only one pass through the request is allowed.

To specify back branching, include a GO, GS, or NS branching operator followed by the branch location BACK.

Looping on the intermediate structure is not affected by backward branching. In other words, any loops already started on the intermediate structure are not terminated, and a backward branch will not cause a loop to the next occurrence of any segment. This is in contrast to the GO/GS/NS END branch, that does cause a loop to the next segment occurrence.

A back branch does not cause re-initialization of any temporary fields unless the re-initialization of temporary fields specification is B.

Figure 7-15 shows an example of back branching in a subroutine request.

* REQUEST NAME - DATEAMT *										
* INPUT STREAM REQUEST *										

STMT	CAT						REQ	SCREEN	BACK BR RNT	
TYPE	ACT	REQUEST DESCRIPTION					TYT	NAME	CONTROL TMP	
(ER)										
COMMENT	*****	THIS REQUEST COMPUTES INTEREST OF OVERDUE PAYMENTS.					*****	(2) (SCREENA)		
STMT	FIELD	FIELD	FLD	DEC	OUTPT	EDIT	INITIAL			
TYPE	NAME	LENGTH	TYP	PLC	EDIT	LGTH	VALUE	COMMENTS		
(TF)	(YY	)	(	2)	(C)					
(TF)	(MM	)	(	2)	(C)					
(TF)	(LOOP	)	(	4)	(Z)					
(TF)	(NEWAMT	)	(	8)	(Z)					
(TF)	(INTEREST)	)	(	8)	(Z)					
STMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD		
NUMB	NO.	LEV	CTR	QLF FIELD NAME	ATN	QLF FIELD NAME OR CONSTANT	QLF FIELD NAME	STRT	NUMB OPR	
(	1)	(010)		(F,TODAY	) (R)		(T,YY	) (	5) (2) (A)	
(	2)	(020)		(T,YY	) (-)	(DUEDATE	)	(T,YY	) (	5) (2) (B)
(	3)	(030)		(T,YY	) (LT)	(D,0	)			
(	4)	(040)			(GS)	(NEXT REQUEST	)			
(	5)	(050)		(T,YY	) (*)	(D,12	)	(T,YY	)	
(	6)	(060)		(F,TODAY	) (R)		(T,MM	) (	1) (2) (A)	
(	7)	(070)		(T,MM	) (-)	(DUEDATE	)	(T,MM	) (	1) (2) (B)
(	8)	(080)		(T,MM	) (+)	(T,YY	)	(T,LOOP	)	
(	9)	(090)			(R)	(PRINCIPL	)	(T,NEWAMT	)	
(	10)	(100)			(GO)	(SUB INTEREST	)			
(	11)	(110)		(T,NEWAMT	) (-)	(PRINCIPL	)	(T,INTEREST)		

* REQUEST NAME - INTEREST *										
* INPUT STREAM REQUEST *										

STMT	CAT						REQ	SCREEN	BACK BR RNT	
TYPE	ACT	REQUEST DESCRIPTION					TYT	NAME	CONTROL TMP	
(ER)										
COMMENT	*****	THIS SUBROUTINE COMPOUNDS THE INTEREST OVER THE NUMBER					*****	(S)		
COMMENT	*****	OF MONTHS SPECIFIED IN THE FIELD T,LOOP.					*****	(120)		
STMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD		
NUMB	NO.	LEV	CTR	QLF FIELD NAME	ATN	QLF FIELD NAME OR CONSTANT	QLF FIELD NAME	STRT	NUMB OPR	
(	1)	(010)		(T,LOOP	) (GT)	(D,0	)			
(	2)	(020)		(T,NEWAMT	) (*)	(D,1.015	)	(T,NEWAMT	)	
(	3)	(030)		(T,LOOP	) (-)	(D,1	)	(T,LOOP	)	
(	4)	(040)			(GO)	(BACK	)			

Figure 7-15 Back Branching in a Request

The example computes a compound interest charge for an overdue principle amount at the rate of 1.5% per month. It uses two data base fields, PRINCIPL and DATEDUE (format MMDDYY). The type 2 request DATEAMT computes the number of months over which interest is to be charged. DATEAMT then invokes INTEREST to compute the compound principle-plus-interest amount. INTEREST returns control to DATEAMT at statement 110 and the interest is calculated.

Screen/COMMAREA Field Error Processing in Screen/COMMAREA Requests or Procedures

Set Error and Message Sending

When you code requests or procedures to validate multiple input screen fields, you may want to set the error handling attributes for those screen fields in error (highlighting is the error handling attribute supported by VISION:Transact). In requests, the SE operator sets the error handling attribute for the input screen field named in Operand A of the PR statement. In procedures, the SET command sets the error handling attribute for the named input screen field. The SE operator or SET command does not cause the redisplay of the input screen.

Each time the SE operator or SET command is executed, the value in the flag field F,ERROR is incremented by one (F,ERROR is initialized to zero at the start of each interaction). A non-zero value in F,ERROR is an indication that an SE operator or SET command has been executed. This flag can thus be used in logic to decide whether or not to execute a message sending operator or command to force the redisplay of the input screen.

In requests, the MS operator redisplay the input screen with the message specified by Operand B on the error message line. In procedures, the MSG command redisplay the input screen with the specified message on the error message line. Data base updates are backed out unless NOROLLB is specified with the MS operator or MSG command.

The following example shows how to use the SE and MS operators to validate multiple screen fields in one type 1 request.

```

*****
* REQUEST NAME - TESTFLDS *
* INPUT STREAM REQUEST *
*****

```

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK CONTROL	BR TMP	RNT
(ER)			(1)	(INPUT1)			

STMT NUMB	SEQ NO.	LOG LEV	CON CTR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD NUMB	OPER OPR
(	1			(S, FIELD1)	(EQ)	(P, .INTEGER				)
(	2				(GS)	(200				)
(	3			(S, FIELD1)	(SE)					)
(	4				(GO)	(200				)
(	5			(S, FIELD2)	(EQ)	(P, .TODAY				)
(	6				(GS)	(300				)
(	7			(S, FIELD2)	(SE)					)
(	8				(GO)	(300				)
(	9			(S, FIELD3)	(GT)	(D, 2500.00				)
(	10				(GS)	(400				)
(	11			(S, FIELD3)	(SE)					)
(	12				(GO)	(400				)
(	13			(F, ERROR)	(GT)	(D, 0				)
(	14				(GS)	(500				)
(	15				(GO)	(END				)
(	16				(R)	(C, FIX THE INCORRECT FIELD(S)	(T, MESSAGE)			)
(	17				(R)	(F, ERROR	(T, MESSAGE)	(27)	(02)	(C)
(	18				(MS)	(T, MESSAGE				)

Figure 7-16 Using the SE and MS Request Operators

In the example, three screen fields are being validated. The coding in line 100 checks to see that the data in screen field FIELD1 is an integer. If it is, you branch to the next field check; if not, FIELD1 is set in error using the SE operator (on line 120), and then you branch to the next field check. Similarly, FIELD2 is tested to see if its date format is correct and FIELD3 is tested for a value greater than 2500.00.

In line 400, you check the flag field ERROR for a value greater than 0. If all the fields passed the validation, the value in F,ERROR would be zero, the relational test in line 400 would be false, and statement 420, GO END, would be executed to end this request with no error processing. However, if any of the fields failed validation, then the value in F,ERROR would be greater than zero, the relational test in line 400 would be true, and you would then branch to statement 500 to build an error message and cause the screen to be redisplayed.

Notice that in line 500 a character string is moved into a temporary field named MESSAGE, and in line 510, the value in the flag field ERROR is placed into the field message by partial fielding. Finally, the MS operator is issued to cause the screen to be redisplayed. The character string built in T,MESSAGE will be displayed on the error message line.

Segment Actions and Data Base Availability to Screen/COMMAREA Requests and Procedures

There are two different sets of rules which govern data availability:

1. Timing - a segment may or may not exist in the intermediate structure depending upon when you ask for it.
The validity of a field at any given time is also part of timing. If a dynamically synchronized data base is referenced when the synchronizing field is not available or is invalid, the data base retrieval cannot be made. Should that field be available and valid at a later time in the interaction and the dynamically synchronized data base is again referenced, the retrieval can be made.
2. Screen synchronizing - when you screen synchronize a particular segment occurrence, the other segment occurrences of that type (and all of their dependent segments) become unavailable, if not also referenced on the screen. For example, if you match only on D1, D2 will not be available.

Segment actions (screen synchronization) are performed between type 1 and type 2 times. Segments to be deleted are available at type 2 time and unavailable thereafter. Segments to be inserted are not available until type M time. This is illustrated in [Figure 7-17](#), where D1 is deleted, C1 is inserted, and E1 is updated. The identifier (S) stands for screen segment and (DB) for data base segment. Example 1 shows the effect of deletions, Example 2, the effect of insertions, and Example 3, the effect of updates on individual data base segment occurrences. Unavailable segment occurrences are treated as missing (see [Processing Invalid, Missing, and Overflow Fields on page 9-9](#)) but are still processed as occurrences during a loop. This is reflected in the execution tracing messages output for looping. Execution Tracing is discussed in [Chapter 12, "Execution Tracing"](#).

Note that segment occurrences for which the X segment action is selected are unavailable at type 2 time and thereafter.

Intermediate Structure			
	Example 1	Example 2	Example 3
Time	Deletions	Insertions	Updates
Type-1	(S) D1	(S) C1	(S) E1
Type-2	(S) D1 (DB) D1	(S) C1	(S) E1 (DB) E1
Type-M	(S) D1	(S) C1 (DB) C1	(S) E1 (DB) E1
Type-O		(DB) C1	(DB) E1

Figure 7-17 Deleted/Inserted Segment Availability

Note in [Figure 7-17](#) that at type 2 time a slot is reserved in the intermediate structure for the insertion of data base segment C1, and at type M and type O times a slot remains for the deleted data base segment D1. The data base segment occurrences are considered missing at these points, because there is a slot in the intermediate structure for them, but they are unavailable to process. Your use of fields from these missing occurrences as input to calculations or replace or move operations causes the result fields of the operations to be MARKed invalid.

[Figure 7-19](#) and its accompanying text show how to check for missing or invalid conditions in a screen request or procedure. [Processing Invalid, Missing, and Overflow Fields on page 9-9](#) has more information on the processing of invalid fields.

The dynamic selection of segment actions influences the processing of the intermediate structure, especially during output processing. [Figure 7-18](#) shows an intermediate structure with a match (M) action at the root level and update (U), insert (I), delete (D), and ignore (X) actions performed by different occurrences of level 2 screen segments, acting on different occurrences of a data base segment.

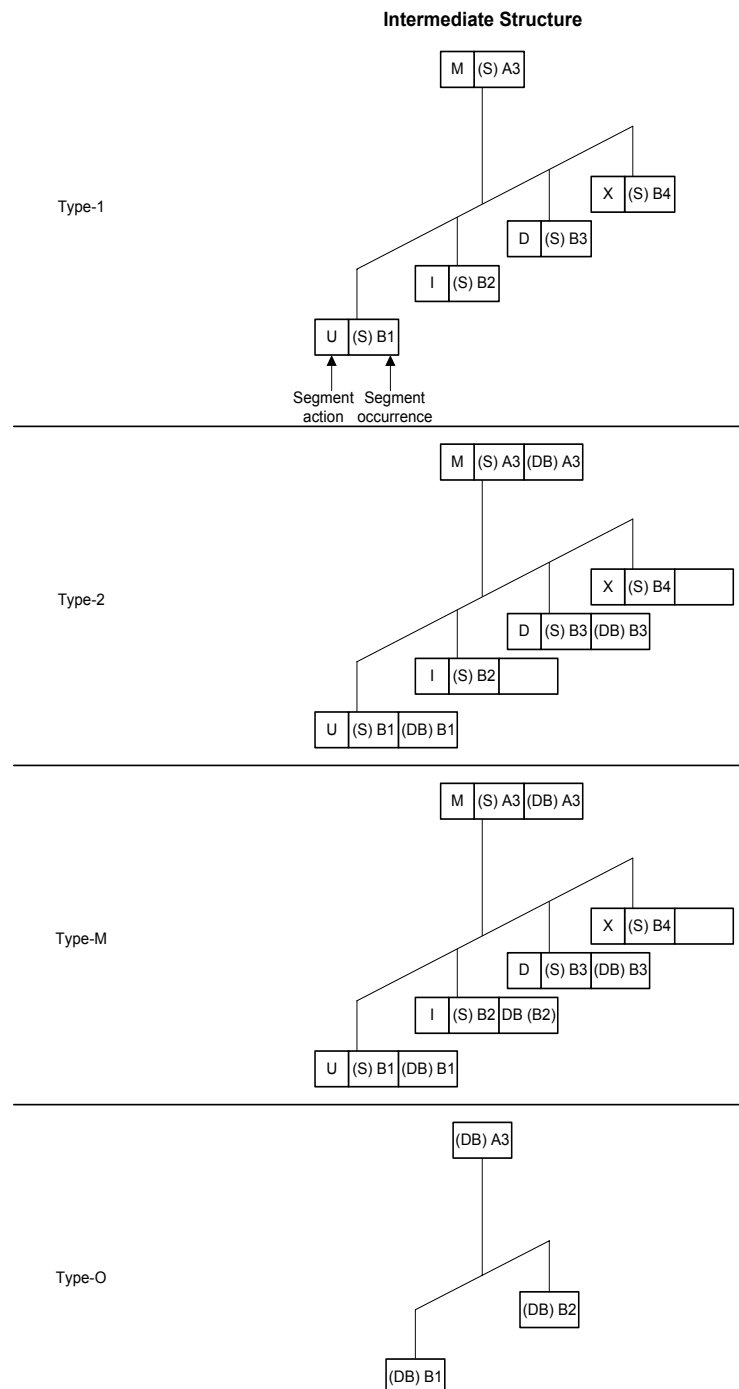


Figure 7-18 Effect of Dynamically Selected Segment Action on the Intermediate Structure

As in [Figure 7-17](#), [Figure 7-18](#) shows a missing data base segment at type 2 time where B2 will be inserted, and a missing data base segment at type M time where B3 has been deleted. Segment B4, which is being ignored, is missing at both type 2 and type M times. The missing segments can affect your request or procedure processing if looping is begun on the intermediate structure by making reference to fields in lower level composite segments. If BFIELD is a field in data base segment B (whose occurrences in [Figure 7-18](#) are B1, B2, B3, and B4), coding such as that in [Figure 7-19](#) will bypass the processing of occurrences of the intermediate structure in which these missing segments reside.

```
REQUEST
-----1-----2-----3-----4-----5-----6-----7--
EXAMPLE PR    BFIELD EQ BFIELD
EXAMPLE PR                NS END
                <PROCESS AVAILABLE OCCURRENCES>
```

```
PROCEDURE

    IF BFIELD EQ BFIELD
        <PROCESSES AVAILABLE OCCURRENCES>
    END

or

DO FORALL BSEGMENT <-- NON ROOTS
<PROCESSES AVAILABLE OCCURRENCES>
END
```

Figure 7-19 Code to Bypass the Processing of Missing Data Base Segments

During type O request or procedure processing, a segment occurrence is removed from the intermediate structure when all of its screen synchronized data base segment occurrences are unavailable or missing. This is shown in [Figure 7-18](#) where segments B3 and B4 have been removed from the intermediate structure.

If a data base segment is completely ignored by the input screen, that is, if it has all X actions dynamically selected for its displayed occurrences, the segment is not synchronized to the screen at all. If the non-synchronized data base segment is the root, then no field from that data base is available to the interaction, because the root of an SSDB must be synchronized to the screen for the SSDB record to be available to the interaction. If the non-synchronized data base segment is a lower level segment, any reference to a field from that segment causes a loop on all occurrences of the segment within its parent(s) which has (have) been synchronized to the input screen. In effect this reference extends the intermediate structure by causing the retrieval of all occurrences of a new segment.

[Figure 7-20](#) shows the result of such a reference in a type 2 or type M request or procedure.

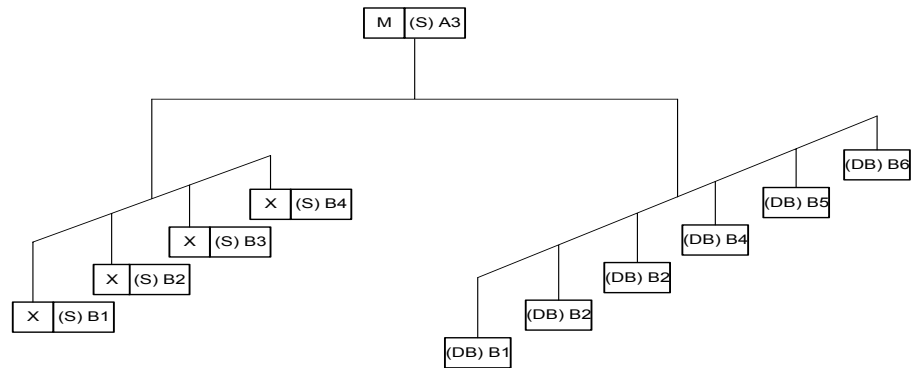


Figure 7-20 Intermediate Structure After Reference to Non-Synchronized Lower Level Segment

[Figure 7-21](#) helps summarize the issue of availability discussed in this section.

Request/ Procedure Type	Screen Fields Available	SSDB Segment Available? Segment Action Codes						
		M	U	I	V U/I	D	X	
Type 1 Time	Yes	No	No	No	No/No	No	No	
Type 2 Time	Yes	Yes	Yes	No	Yes/No	Yes	No ¹	
Type M Time	Yes	Yes	Yes	Yes	Yes/Yes	No	No ¹	
Type O Time	Yes	Yes	Yes	Yes	Yes/Yes	No	No ¹	

¹Segment occurrences with an X action may be available if all of the occurrences for the segment type selected the X action as illustrated in [Figure 7-20](#).

Figure 7-21 Intermediate Structure and Segment Availability

Temporary fields, working storage fields, flag fields, and PCB masks are available to any request or procedure type.

Dynamically synchronized data base fields are available at any time during the interaction cycle, as long as its synchronizing field is valid and available as well.

Extending the Intermediate Structure

Extending the intermediate structure in the manner shown in [Figure 7-20](#) is also possible by your request or procedure referencing SSDB segments which have not been referenced by the input screen.

Imagine a data base record such as that illustrated in [Figure 7-22](#).

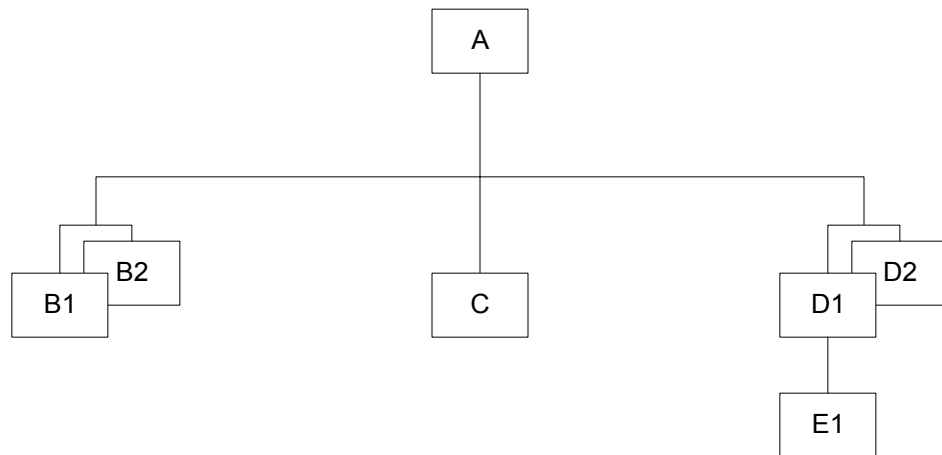


Figure 7-22 Data Base Record

Screen matching on D1 causes the intermediate structure to look like the one in [Figure 7-23](#) (where the symbol(S) stands for screen segment and (DB) stands for data base segment).

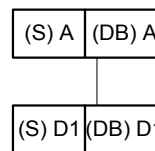


Figure 7-23 Concatenated Segments

When you reference the B and C segments in a request or procedure, the structure looks like [Figure 7-24](#). Notice that all occurrences of data base segments B and C now exist in the intermediate structure.

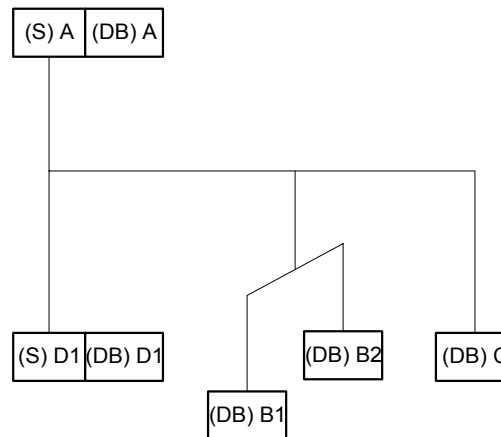


Figure 7-24 Extended Intermediate Structure

Segments thus added to the intermediate structure will be automatically updated if

1. They are referenced in a type 2 or M request or procedure, that is, before VISION:Transact data base update.
2. The screen synchronized parent segment is updatable (i.e., the lowest level that is screen synchronized in the same path has a U or V segment action code).
3. The data fields are updated via request or procedure logic.

In the example, if segment A is updatable, then data base occurrences B1, B2, and C are updatable.

Virtual Screen/COMMAREA Fields

A virtual screen field is a type of screen field which neither displays data on the screen nor receives data directly from the terminal user. A virtual screen field can be designated as Input, Output, or Both (I, O, or B). When a virtual field is an input field, it is a screen field, and has an S qualifier. It is defined within the screen processing definition and used like an input screen field by the application. A virtual screen field can also be used as a multiply occurring temporary field. It is accessible to you, the application developer, but is invisible to the terminal user. Besides the default value operator (DF) in the screen definition, requests and procedures are the only means of entering data into input virtual screen fields. Input virtual screen fields are usually manipulated by type 1 requests or procedures, acting on the input screen content, but they are available throughout input screen processing.

Input virtual screen fields are useful when you need to screen synchronize data that is not input directly from the terminal. For instance, suppose you have a five byte key (of a segment called PARTNO) whose first two bytes are the year and last three are the day of the Julian date. The user can enter two screen fields: one field as the year and one as the Julian day. You can then use request or procedure logic to concatenate the two fields into a virtual screen field, which can have the data base action code associated with it (in which case the two screen fields would have null action codes). In this way you can access the PARTNO segment without the terminal user directly entering the true key. Alternatively, the user could enter two data base keys in one screen field, and request or procedure logic can separate the screen field into multiple virtual screen fields.

An input virtual screen field can also be used to translate input data into codes, for human engineering purposes. Imagine a data base that is organized by city and each city has been assigned a number. Because it is difficult for most terminal users to remember which number represents which city, the application allows him to enter the city name in the screen field, and request or procedure logic uses a virtual screen field to translate the name to a number that can then be used as the key.

The length of a non-virtual screen field is determined by the number of consecutive field designators on the Screen Layout portion of the Screen Processing Definition. A virtual input screen field is usually assigned a length according to its destination field since virtual screen fields do not have field designators on the SL statements. If there is no destination field associated with a virtual screen field, assign its length using the S0 or Sn statement with the LN edit/validation operator. Specify the length of the virtual screen field in the edit/validation field of that statement.

Output virtual screen fields are used primarily to drive output screen building. They are defined by the screen field which does not display on the screen but is used in building the output display. For example, if a structured screen has a lower level screen segment defined consisting of temporary fields only, a virtual output screen field with a multiply occurring source field must be defined in that segment to drive the loop building the lower level output screen segment. Defining a virtual output screen field allows the application to loop through the temporary fields, in the lower level screen segment.

Except for the display on the screen, all characteristics pertaining to displayed screen fields also apply to virtual screen fields.

Note: A virtual field defined as both output and input is only meaningful in recycle interactions (see [Chapter 11, “Advanced Application Flow”](#)).

Looping

Looping is the automatic processing of multiple occurrences of recurring segments (screen or data base) by your VISION:Transact generated application. Within a request or procedure, the first executed reference to any field in a recurring segment initiates a loop. The starting point (or top) of the loop is the first executed field reference. The end (or bottom) of the loop is the END of the same request or procedure. If multiple segments are referenced in the same statement, the leftmost operand starts the first loop.

Within procedures, the DO FORALL segment-name command causes execution of the DO block for all of the named segment occurrences. Each interaction of DO FORALL block processes the next occurrence of the specified lower level segment.

Execution tracing, described in [Chapter 12, “Execution Tracing”](#), can be used to output trace information revealing the status of loops that occur during execution.

It is important to understand that, conceptually, there is VISION:Transact code inserted following each request and procedure. One function of that code is to determine whether:

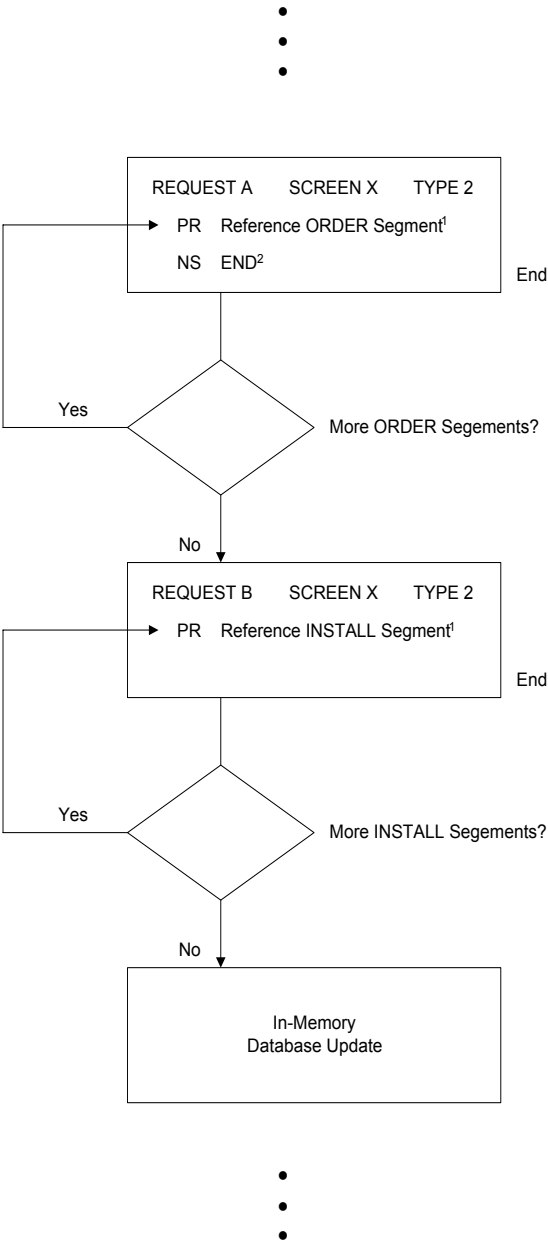
- a loop has been established, and
- if so, whether there are additional loop-driving segment occurrences.

If there are additional loop driving segment occurrences, control will be returned in the request or procedure to the top of the loop. If not, control will be passed to the next request or procedure (if any) or to the next interaction function.

A branch or transfer to the location NEXT REQUEST, NEXT FUNCTION, or SCREEN CONTROL stops all loops that have been started in a request or procedure. This branch or transfer may be used whether or not there is another request or procedure following the current one. The branch means, in effect, that the next interaction function should be invoked, following the immediate execution of the clean-up logic for this request or procedure. The next interaction function may be another request or procedure of the same type or an automatic function generated by VISION:Transact, such as a data base read or an in-memory update.

If a branch or transfer to NEXT REQUEST, NEXT, FUNCTION or SCREEN CONTROL is executed prior to the application reading all of the occurrences of the data base segment involved in the loop, the next reference to that data base segment can cause additional data base reads.

[Figure 7-25](#) illustrates two requests written using fixed form statements for the same input screen. Request A includes a loop through the ORDER segments in the Customer data base. Request B includes a loop through the INSTALL segments.



1 For procedures, the segment reference would be specified on a command (for example, IF ORDERNO EQ '00001').

2 For procedures, the CONTINUE command branches to the end of the procedure.

Figure 7-25 Requests Illustrating Simple Looping

[Figure 7-26](#) presents a situation where the effective bottom of the loop is not the physical end of the request. The first PR statement in the loop is at statement 030; the last one is at statement 120. Statements 130 through 150 are "outside" of the loop and are "reached" by the NS 130 instruction (statement 020). Another loop starts at statement 130 and ends at statement 150. Due to the request logic (statements 010 and 020) only one loop will execute per reference to a lower level segment - never both.

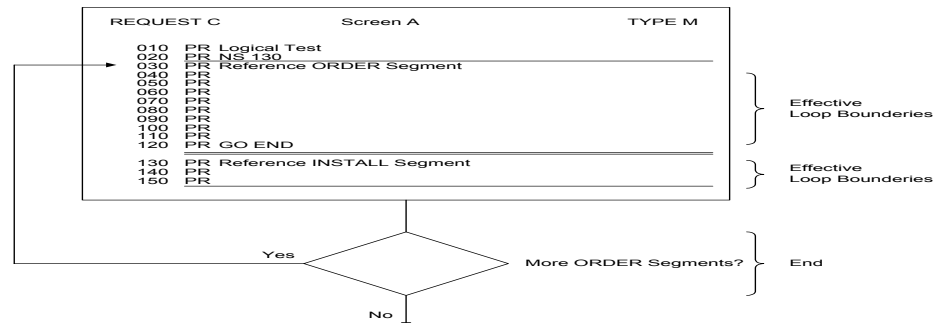


Figure 7-26 Request Illustrating Refinement of Loop Boundaries

This technique is especially useful in the case of type O requests or procedures where you are permitted only one request or procedure per output screen. With careful design of branching and branching conditions, it is possible to expand this technique to include two or more simple loops in one request or procedure, either of which may be executed during a single "execution" of the request or procedure.

Many programmers follow the practice of initiating loops in requests by replacing the appropriate segment key field into itself. The cost of executing such an "extra" instruction is negligible. The practice has the advantage of making the initialization of loops more self-documenting. In procedures, loops can be initiated by replacing the segment key into itself (LET key = key) or by using the DO FORALL segment-name command. Comparison of the key field to itself will also start a loop. Because the loop being started is based on the composite segment, rather than the data base segment itself, it is possible for the data base segment to be missing even though the composite segment does exist. In that case the loop will be active even though the false path is taken, unless the branch or transfer is to NEXT REQUEST or NEXT FUNCTION.

Nested Looping

The concept of the single path nested loop refers simply to the need to process subordinate segments at more than one level along a single path of the data base structure illustrated in [Figure 7-27](#).

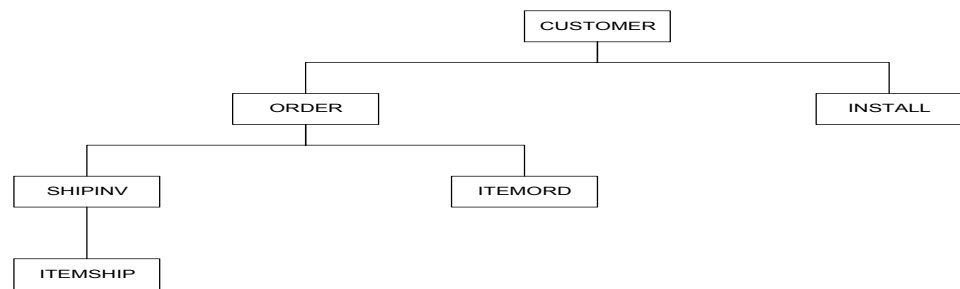


Figure 7-27 Customer Data Base

The VISION:Transact handling of such requirements is straightforward. Consider the request illustrated in [Figure 7-28](#).

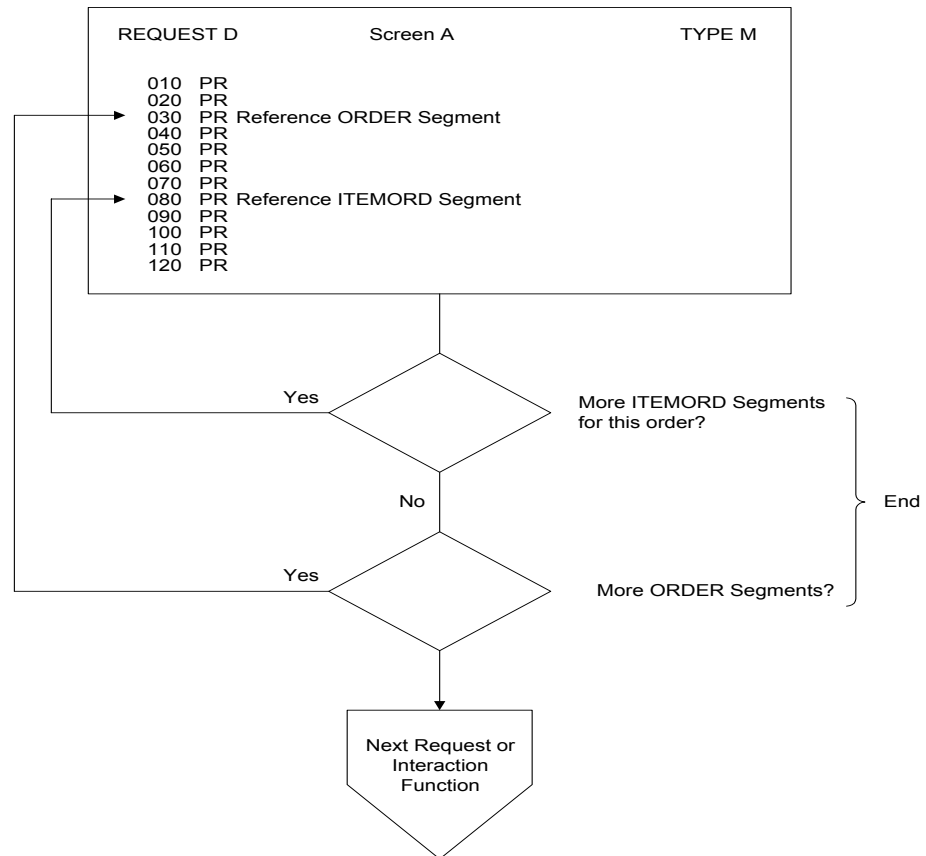


Figure 7-28 Request Illustrating Single Path Nested Looping

Request D initiates a loop through the ORDER segments at statement 030 and then performs processing operations using data in the ORDER segment. At statement 080, this request initiates a loop through the ITEMORD segments for the current order. Statements 080 through 120 have access to the fields in one ITEMORD segment and one ORDER segment.

Initiate nested loops from top to bottom through a path in your data structure. Initiate loops only for those segment types which your request or procedure needs to access. VISION:Transact can then optimize the necessary data base calls across the "missing" segment types. Implicit loops will exist for any unreferenced parent segment types; that is, your request or procedure will "see" all of the occurrences of the segment type you referenced for each occurrence of a parent.

The multi-path nested loop concept refers to the processing of subordinate segments which are along different paths in the intermediate structure (sibling segment processing). VISION:Transact processes multiple path nested loops in the same way as single path nested loops (see [Figure 7-29](#)).

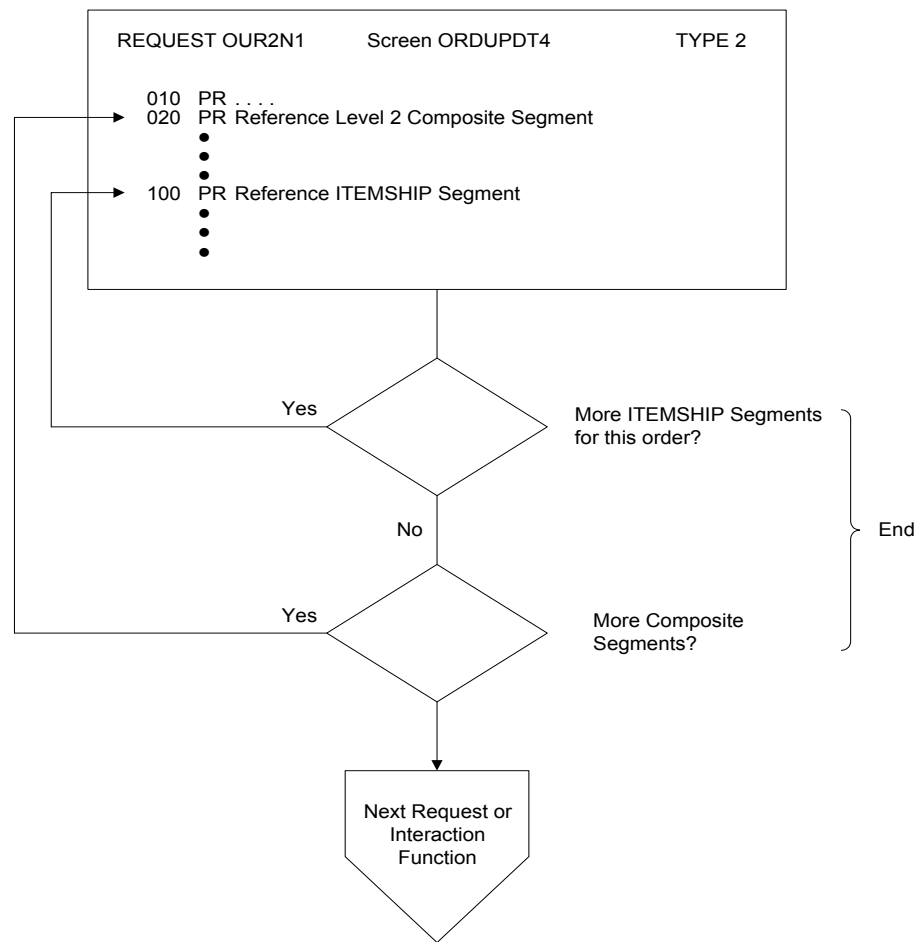


Figure 7-29 Request Diagram Illustrating Multi-Path Nested Looping

[Figure 7-29](#) diagrams the loop processing in request OUR2N1, whose complete source coding is shown in [Figure 7-30](#).

```

*****
* REQUEST NAME - OUR2N1 *
* INPUT STREAM REQUEST *
*****

-----
STMT CAT                                REQ SCREEN BACK BR RNT
TYPE ACT REQUEST DESCRIPTION          TYP NAME CONTROL TMP
-----
(ER)                                     (2) (ORDUPDT4)
COMMENT ***** THIS REQUEST MAKES SURE THAT AN ITEM WHOSE QUANTITY *****
COMMENT ***** ORDERED HAS BEEN CHANGED HASN'T ALREADY BEEN SHIPPED. *****
-----
STMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD
NUMB NO. LEV CTR QLF FIELD NAME ATN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OFR
-----
( 1) (010)          (F,ATINID ) (EQ) (C,ENTER )
( 2) (020)          (A) (S,ACTION ) (EQ) (C,U )
( 3)              (NS) (END )
COMMENT *****
COMMENT ***** DOES THIS ITEM ALREADY EXIST ON THIS ORDER? *****
COMMENT *****
( 4) (040)          (ITEMORD ) (EQ) (ITEMORD )
( 5) (050)          (NS) (160 )
COMMENT *****
COMMENT ***** IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED? *****
COMMENT *****
( 6) (070)          (S,ITMOTYOR ) (EQ) (ITMOTYOR )
( 7) (080)          (GS) (END )
COMMENT *****
COMMENT ***** IF SO, HAS THE ITEM BEEN SHIPPED? TO FIND OUT, *****
COMMENT ***** STATEMENT 100 STARTS A NESTED LOOP ON THE ITEMSHIP *****
COMMENT ***** SEGMENT TO COMPARE THE CURRENT ITEMORD AGAINST ALL *****
COMMENT ***** ITEMSHIP VALUES FOR THIS ORDER. *****
COMMENT *****
( 8) (100)          (ITEMORD ) (EQ) (ITEMSHIP )
( 9) (110)          (NS) (END )
( 10) (120)          (R ) (C,CANNOT CHANGE QUANTITY ORDERED ) (T,MSG )
( 11) (130)          (R ) (C,- ITEM ALREADY SHIPPED ) (T,MSG ) (32) (31) (C)
( 12) (140)          (S,ITEMORD ) (SE) ( )
( 13) (150)          (MS) (T,MSG )
COMMENT *****
COMMENT ***** IF THE PRICE HAS NOT BEEN ENTERED ON THIS NEW ITEM, *****
COMMENT ***** GET THE CURRENT PRICE FROM THE ITEM DATA BASE. *****
COMMENT *****
( 14) (160)          (S,ITMPRICE) (EQ) (C)
( 15) (170)          (NS) (END )
( 16) (180)          (R ) (1,ITMPRICE ) (S,ITMPRICE)

```

Figure 7-30 Request Using a Multi-Path Nested Loop to Perform Validation

Statement 020 of request OUR2N1 initiates a loop on the intermediate structure by referencing the screen field S,ACTION, which is in level 2 of the intermediate structure. Statement 040 references the ITEMORD segment, which has been synchronized to the level 2 composite segment by a V segment action. This reference is contained in the loop begun by statement 020. Statement 050 invokes alternative processing when an ITEMORD segment is being inserted, because the segment to be inserted is missing from the composite segment at type 2 time.

Statement 100 of request OUR2N1 references the ITEMSHIP segment of the same data base. The ITEMSHIP segment, as you can see in [Figure 7-31](#), is in a different path of the same data base than the ITEMORD segment. This reference to the ITEMSHIP segment extends the intermediate structure and begins a multi-path nested loop on the ITEMSHIP segment. The comparison of the ITEMORD field to

the ITEMSHIP field causes the processing of all occurrences of the ITEMSHIP field for each processed occurrence of the ITEMORD field. When an equal condition is found, it is because the terminal user has tried to change the quantity-ordered for an item that has already been shipped. The request invokes error processing to inform the user of this error.

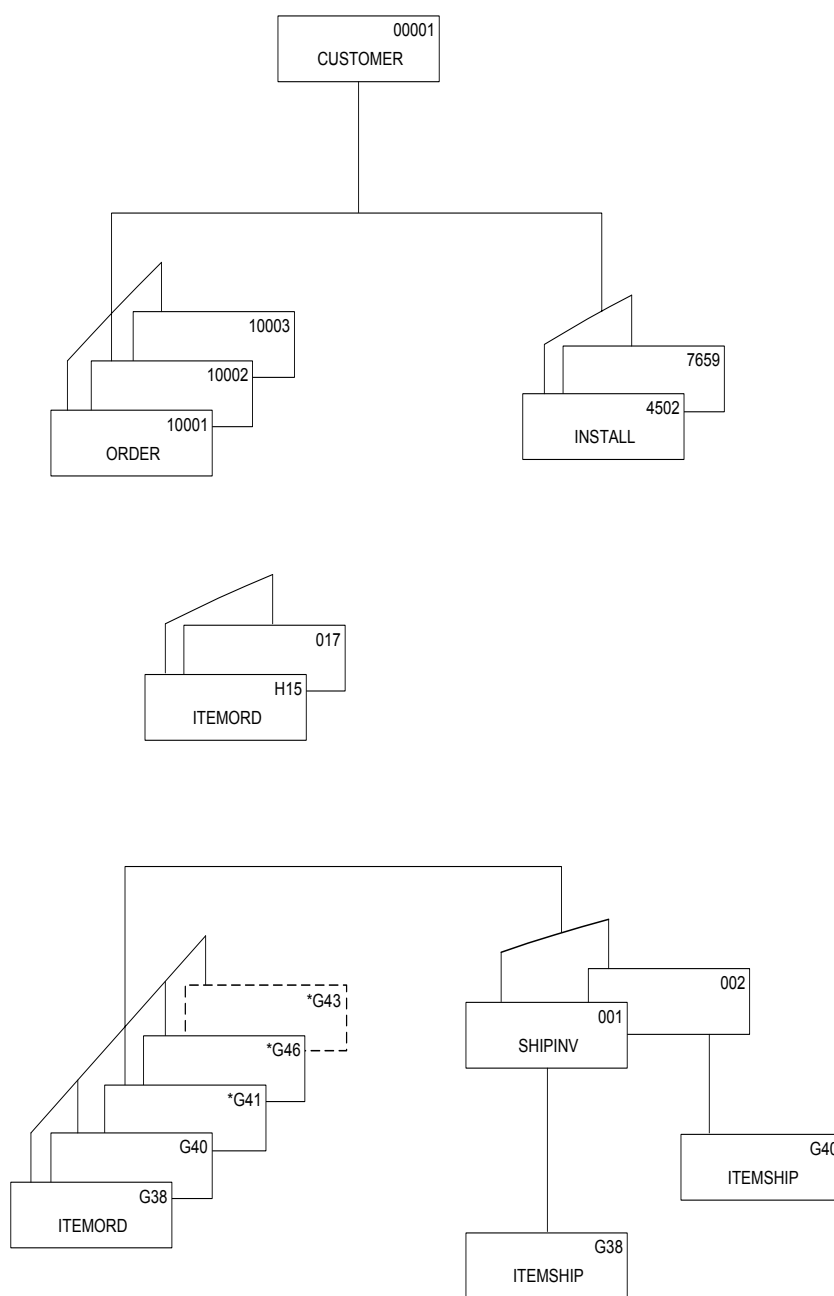


Figure 7-31 Customer Data Base Record

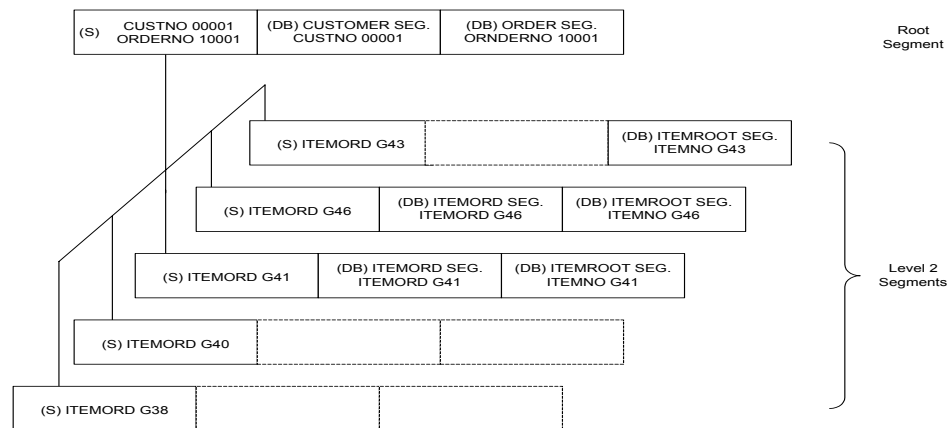


Figure 7-32 Intermediate Structure at Type 2 Time

[Figure 7-33](#) shows the request processing cycles set in motion by these references to the intermediate structure in [Figure 7-32](#) and the data base record in [Figure 7-31](#).

Cycle	ITEMORD	ITEMSHIP
1	G41	G38
2	G41	G40
3	G46	G38
4	G46	G40

Figure 7-33 Segment Processing with Multi-path Nested Looping

By comparing the segment processing in [Figure 7-33](#) to the intermediate structure in [Figure 7-32](#), you can see that some occurrences of the intermediate structure are being bypassed in this processing. At this point (statement 100) in the processing of the request, the first two occurrences of the composite segment, G38 and G40, have already been bypassed because S,ACTION is not a "U" for those occurrences, and statements 010/030 check for this condition. The last occurrence of the composite segment is missing at this point, because it is being added to this order. Statements 040/050 prevent this occurrence from entering the nested loop by branching to statement 160 when the ITEMORD segment is missing.

[Appendix A, "Order/Item Application Screen and Database Definition Code"](#) contains the source code for the ORDUPDT4 screen, input to this interaction. [Figure 7-31](#) shows a Customer data base record which could be processed by this interaction, and [Figure 7-32](#) shows an intermediate structure for this interaction, with some occurrences of ITEMORD segment synchronized to it at type 2 time.

Screen/COMMAREA Request/Procedure Summary

[Figure 7-34](#) highlights the components of a VISION:Transact generated application that extend the interaction.

Screen requests or procedures extend the interaction by allowing you to perform extensive validation of input screen data, perform procedural updates of segments (those synchronized as well as not synchronized to the screen), select segment occurrences for the output screen display, etc. The availability of data to screen requests or procedures depends on the timing of its execution, which is determined by its type. Requests or procedures executed at type 1, 2, M, and O time reference screen data and screen synchronized data bases via the intermediate structure. These same screen requests or procedures reference other data - dynamically synchronized data bases, PCBs, working storage, and temporary fields - which are not part of the intermediate structure. However, access to a dynamically synchronized data base can be based on the contents of a synchronizing field contained in the intermediate structure. The processing of dynamically synchronized data bases is detailed in [Chapter 8, “Dynamic Synchronization”](#). Pseudo files are discussed in [Chapter 10, “Defining Data”](#).

Execution tracing, described in [Chapter 12, “Execution Tracing”](#), can output useful trace information during the execution of requests or procedures such as when a request/procedure is entered and exited, status of loops initiated, I/O calls generated, invocation of the error cycle, and literals or field content output from trace operators or commands.

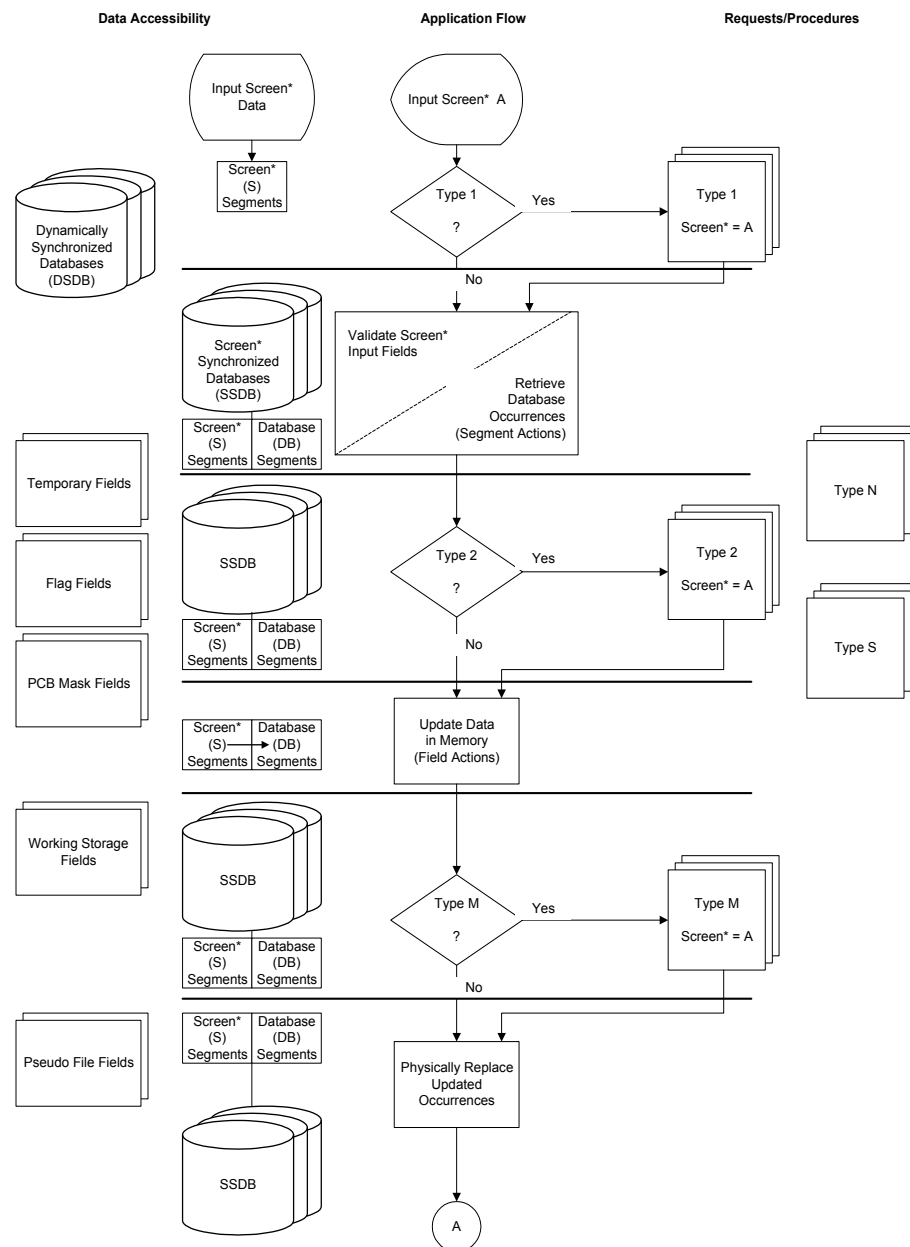


Figure 7-34 Extended Interaction Components (Page 1 of 2)

* "Screen" can also be "COMMAREA section".

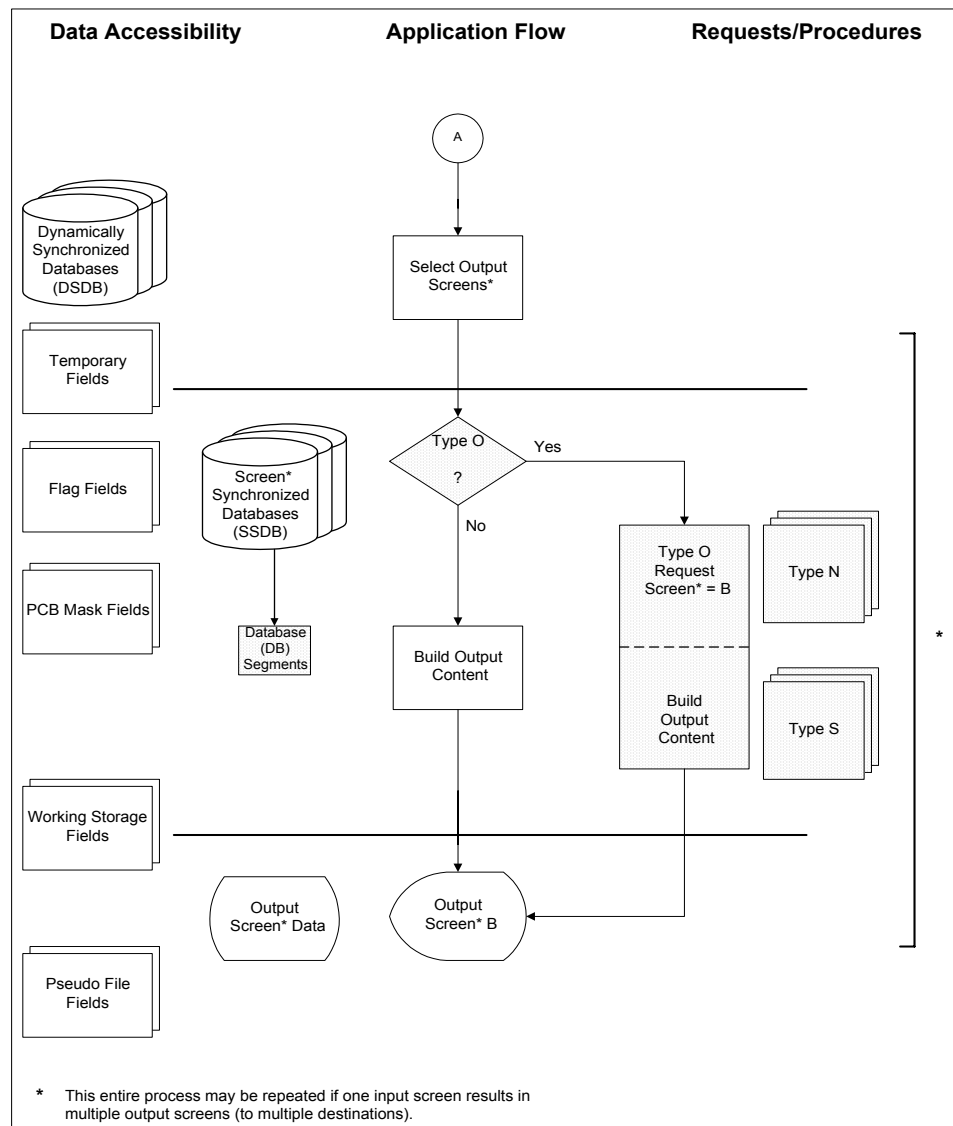


Figure 7-34 Extended Interaction Components (Page 2 of 2)

* "Screen" can also be "COMMAREA section".

Dynamic Synchronization

In [Chapter 6, "Structured Screens/COMMAREA And The Intermediate Structure"](#) we discussed the access of data bases known as screen synchronized data bases (SSDB). Screen synchronization is controlled by screen segment actions specified against a data base. There is another type of data base access, called dynamic synchronization, that allows you to access data bases that are intended for retrieval only.

The access to a dynamically synchronized data base (DSDB) record is not made by your VISION:Transact generated application until a reference is made to a field from that data base. This can occur in a screen request or procedure or on the screen processing definition when a DSDB field is specified as an output field. When reference is made to a field from a DSDB, the appropriate root segment occurrence of the DSDB is determined by the current value in the synchronizing field. The synchronizing field is identified on an RF statement at APPGEN time as containing the root key value of the desired DSDB record to be retrieved. The synchronizing field can be any field that can be referenced at the time DSDB access is desired. For instance, if you wish to access a DSDB record in a type 1 request or procedure a field from an SSDB cannot be used as the synchronizing field because it is not available at that time in an interaction. However, a temporary field initialized to the root key value you desire may be used.

There are two types of dynamic synchronization - direct synchronization and browse synchronization. In direct synchronization a single record is retrieved from a referenced DSDB. The record retrieval is based on the value of a single synchronizing field at the time of the reference.

In addition to direct synchronization, VISION:Transact is capable of retrieving a set of data base records. Browsing allows the application to retrieve a set of DSDB records whose root key value either falls within a range of values or satisfies a "generic" search criterion. In a generic search a leading portion of a key field is used to identify those records that are significant for a certain application. For example, it may be desirable to retrieve all records whose root key begins with the value "M5" regardless of the full key value.

Up to ten data bases may be dynamically synchronized in a VISION:Transact generated application.

Note that only the primary root key field (key field 1 or V) is used for access to dynamically synchronized data bases.

Execution tracing, described in [Chapter 12, “Execution Tracing”](#), can be used to output trace information concerning the looping and I/O calls generated for dynamically synchronized data bases (DSDBs).

Direct Synchronization

Direct synchronization allows read-only access to a single data base record. To specify direct synchronization of data bases, use the RF statement submitted in the APPGEN. The RF statement for the Item data base ([Figure 8-1](#)) illustrates its use.

```
-----
RUN      STMT LIB   SYS  DEBUG SRC LKED  SYS  CICS MSG  POS RUN
NAME     TYPE OPEN DELIM LEVEL LST PGMS ROLLB KEYS PROC PCB TYPE
-----
(RUN      ,RC)                                     (G)
-----
RUN      STMT     FILE      FILE      SYNCHRONIZING      SYNC  SYNC
NAME     TYPE     NAME      TYPE      FIELD 1      FIELD 2      QUAL  TYPE
-----
(RUN      ,RF) (CUSTOMER) (FILE  )                                     (U)
(RUN      ,RF) (ITEM   ) (FILE1 ) (ITEMORD )                             (S)
```

Figure 8-1 Direct Synchronization

In this example the file name entry specifies to use the ITEM file definition in the VISION:Transact library to describe the DSDB. The file type, FILE1, specifies the VISION:Transact name for the DSDB. The 1 implies that this data base is referenced with a 1 qualifier on any request (PR), procedure, and S0/Sn statements. Because an application may synchronize up to ten data bases (FILE-FILE9), the second DSDB may have a 2 qualifier, the third, a 3 qualifier, etc., up to nine. The screen field, ITEMORD, is specified as the synchronizing field (notice the S entry in the Sync Qualifier specification). Based on the RF statement above, one DSDB record will be retrieved whose root key value is equal to the value of the screen field ITEMORD. Direct synchronization uses only one synchronizing field.

Browsing Dynamically Synchronized Data Bases

For an application to browse a set of DSDB records a second synchronizing field is used. This field indicates that browsing is desired for the specified DSDB.

When browse synchronization is the method of processing a DSDB, reference to a field from the DSDB causes your application to retrieve all occurrences of the root segment of the DSDB (subject to the retrieval rules in [Dynamic and Screen/COMMAREA Synchronization on page 8-4](#)) within the range or generic key contained in the synchronizing field(s). Reference to a browse DSDB starts a loop on the root of the DSDB so that each record within the range or generic key is processed automatically.

Range Browsing

When synchronizing field 1 and synchronizing field 2 are different fields, the range of records whose root key is greater than or equal to synchronizing field 1 and less than or equal to synchronizing field 2 will be retrieved. In [Figure 8-2](#), the RF Statement for the Item data base shows an example of its use.

RUN NAME	STMT TYPE	LIB OPEN	SYS DELIM	DEBUG LEVEL	SRC LST	LKED PGMS	SYS ROLLB	CICS KEYS	MSG PROC	POS PCB	RUN TYPE
(ORDENTRY,RC)											(G)
RUN NAME	STMT TYPE	FILE NAME	FILE TYPE	SYNCHRONIZING FIELD 1 FIELD 2		SYNC QUAL TYPE					
(ORDENTRY,RF)		(CUSTOMER)	(FILE)								
(ORDENTRY,RF)		(ITEM)	(FILE1)	(ITEM1)	(ITEM2)	(T)					

Figure 8-2 Range Browse

The run name, FD name, and file type specifications have the same meaning as described previously. Two separate temporary fields, ITEM1 and ITEM2, are specified as the synchronizing range fields for FILE1. Range browsing is initiated when a field from FILE1 is referenced.

Generic Browsing

When synchronizing fields 1 and 2 are the same field, generic browsing is established. The lower and upper limits of the range will be derived as follows:

- The lower limit will be the value in synchronizing field 1 padded in the low order bytes (to the right) with binary zeros (Hex "00"); all trailing blanks in the synchronizing field are replaced with binary zeroes.
- The upper limit will be the value in synchronizing field 2 padded in the low order bytes (to the right) with high values (Hex "FF") all trailing blanks in the synchronizing field are replaced with high values.

[Figure 8-3](#) is an example of specifying generic browse.

RUN	STMT	LIB	SYS	DEBUG	SRC	LKED	SYS	CICS	MSG	POS	RUN	
NAME	TYPE	OPEN	DELIM	LEVEL	LST	PGMS	ROLLB	KEYS	PROC	PCB	TYPE	

(ORDENTRY,RC)										(G)		

RUN	STMT	FILE		FILE		SYNCHRONIZING			SYNC		SYNC	
NAME	TYPE	NAME		TYPE		FIELD 1		FIELD 2		QUAL	TYPE	

(ORDENTRY,RF)		(ITEM	)	(FILE1	)	(ITEM1		(ITEM1		(T)		
(ORDENTRY,RF)		(CUSTOMER)		(FILE	)							(U)

Figure 8-3 Generic Browse

The run name, FD name, and file type specifications have the same meaning as described previously. The temporary field ITEM1 is specified as both synchronizing fields. Generic browsing is initiated when a field from FILE1 is referenced.

Browse Considerations

Generic browsing may only be used for a data base whose root key is a character field.

When synchronizing field 2 is not specified, VISION:Transact will default to direct synchronization.

The synchronizing fields must have the same qualifier. Further, if they are fields in a structured file or screen they must be in the same segment.

At most, one record from a DSDB will be held in memory at any one time. When a new DSDB record is retrieved, the buffer space allocated for the prior record is reused.

When a "not found" condition occurs, VISION:Transact treats the DSDB record as missing data (see [Chapter 10, "Defining Data"](#)).

Dynamic and Screen/COMMAREA Synchronization

A VISION:Transact file (FILE, FILE1, FILE2, etc.) that is being accessed as a DSDB can also be accessed as a SSDB in the same application. Specifying the same data base as two separate VISION:Transact files (FILE1, FILE2) causes the data base to be accessed as a SSDB for FILE1 and as a DSDB for FILE2. In fact, this dual usage is illustrated in the application system shown in [Appendix A, "Order/Item Application Screen and Database Definition Code"](#).

In addition, you can also specify the same data base as an SSDB and DSDB using the same qualifier. [Figure 8-4](#) illustrates this technique.

RUN	STMT	LIB	SYS	DEBUG	SRC	LKED	SYS	CICS	MSG	POS	RUN
NAME	TYPE	OPEN	DELIM	LEVEL	LST	PGMS	ROLLB	KEYS	PROC	PCB	TYPE
(ORDENTRY, RC)										(G)	
RUN	STMT	FILE		FILE		SYNCHRONIZING		SYNC		SYNC	
NAME	TYPE	NAME		TYPE		FIELD 1		FIELD 2		QUAL	
(ORDENTRY, RF)	(CUSTOMER)	(FILE)									
(ORDENTRY, RF)	(ITEM)	(FILE1)		(ITEM1)		(ITEM2)		(T)		(U)	

Figure 8-4 Specifying a Data Base as a DSDB and SSDB Using the Same Qualifier

If used in this way, the data base is processed as a DSDB in type 1 requests or procedures and during output processing. The data base is processed as an SSDB at any other time. In other words, the data base is automatically processed as a DSDB and as an SSDB at specific points in the interaction. Contrast a single qualifier for a DSDB and SSDB with a separate qualifier for each. Using two qualifiers for a data base (processed as a DSDB and SSDB) allows you to dynamically synchronize a data base throughout the interaction; this technique also uses up available qualifiers. Specifying the data base as a DSDB and SSDB with the same qualifier allows greater processing freedom when assigning qualifier to additional files. If a data base is used both as an SSDB and a DSDB, the Sync Qualifier cannot be S (screen field) or a field in an SSDB.

Accessing the Dynamically Synchronized Data Base

A DSDB record is retrieved when the following conditions are met:

1. A reference is made to a field in the DSDB either explicitly or implicitly.
2. The key of the current "direct" DSDB record does not match the current value of the synchronizing field.
3. The synchronizing field is both available and valid.
4. No loop is in progress on the DSDB.

An explicit reference means that the qualifier associated with the synchronized data base must be used in a screen processing definition or a screen request or procedure.

An implicit reference concerns the case where in order to access a segment (in a DSDB) that was explicitly referenced, reference must first be made to another DSDB. [Figure 8-5](#) illustrates this point.

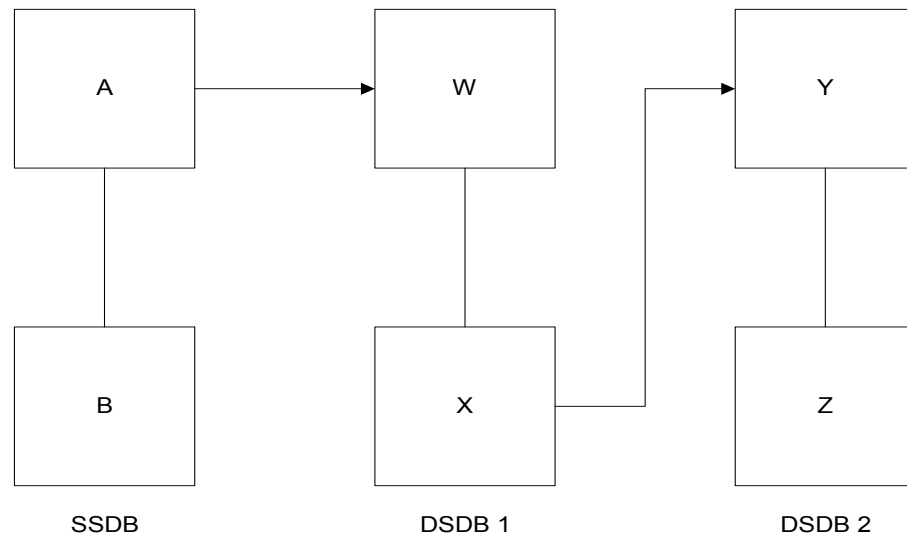


Figure 8-5 Implicit Reference

If an explicit reference is made to Z, implicit references will be made first to segment A, then to segments W, X, and Y before segment Z can be accessed.

The second condition specifies that if the synchronizing field did not change since the last reference to the direct DSDB during the current interaction, it is not necessary to read the synchronized data base since the record is already in memory.

When browsing has been established as the synchronizing method for a DSDB, VISION:Transact establishes an implicit loop when a field from that DSDB has been referenced. The loop will end when one of the following conditions is met:

1. A root segment does not exist on the data base within the range values specified, or
 2. The output screen has been filled and OUTPUT CONTROL "A" has not been specified on the SD statement, or
- Note:** This assumes that a field from the DSDB root segment is displayed on the output screen.
3. A branch or transfer to the location NEXT REQUEST, NEXT FUNCTION, SCREEN CONTROL, or RETURN is executed within the request or procedure logic processed as part of the loop.

As long as the current loop is in process, another read is not issued to the data base.

Browsing is very useful if your data base has non-unique root key values. VISION:Transact will allow you to access all records within the specified range regardless of the individual key value. Recall that browse synchronization holds only one record in storage at a given time. If you reference a browse DSDB in a type M request or procedure and also reference it in a type O request/procedure in the screen as the source of an output screen field, the file will be re-read.

The Use of Dynamically Synchronized Data Bases

The application system described in [Chapter 2, “A VISION:Transact Generated Application System”](#) references the Item data base as a DSDB. In that application system range browsing is used to access the DSDB, even though the application initiates generic browsing for one of its screens (ITMBRWS4). This illustrates one of many possible uses for a dynamically synchronized data base.

Imagine a dynamically synchronized data base with the structure illustrated in [Figure 8-6](#).

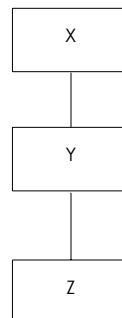


Figure 8-6 Dynamically Synchronized Data Base

If the synchronizing field is D1 and level 3 of the dynamically synchronized data base is referenced, the intermediate structure does not change. The relationship of the intermediate structure and the DSDB is shown in [Figure 8-7](#).

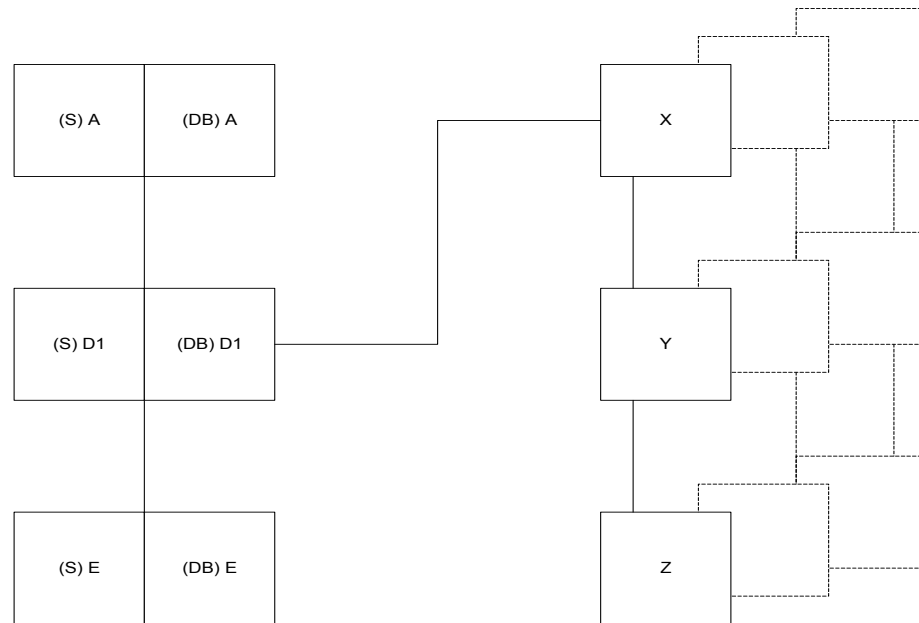


Figure 8-7 The Intermediate Structure and a DSDB

This is a fleeting attachment that depends on the contents of the synchronizing field(s) in (S)D1 or (DB)D1 at the time the DSDB is referenced. If your application is performing browse synchronization, the possibility exists that multiple DSDB records will be retrieved by a reference to a DSDB field (in effect, looping on the root segment of the DSDB); hence, [Figure 8-7](#) shows this possibility with "shadow" records beyond the first one retrieved. Of course, if your application is performing direct synchronization, only one DSDB record is retrieved per reference.

If the contents of the synchronizing field change during a loop on the DSDB, the loop takes precedence. Otherwise, the data base will be resynchronized based on the new value. This means that even if the synchronizing field(s) value(s) changes during the execution of this loop, resynchronization will not occur until the loop has been exhausted and the DSDB is referenced again (see [Figure 8-8](#)). Further, when browsing a DSDB there is an implicit loop in effect on the DSDB root segment. Therefore, changing the synchronizing field(s) value(s) while browsing will have no effect after the DSDB has been referenced. If you need a second range of records, use a second request or procedure, or use a subroutine request or procedure to perform the browse. It is a good practice to set the synchronizing field(s) value(s) prior to referencing a field in the DSDB.

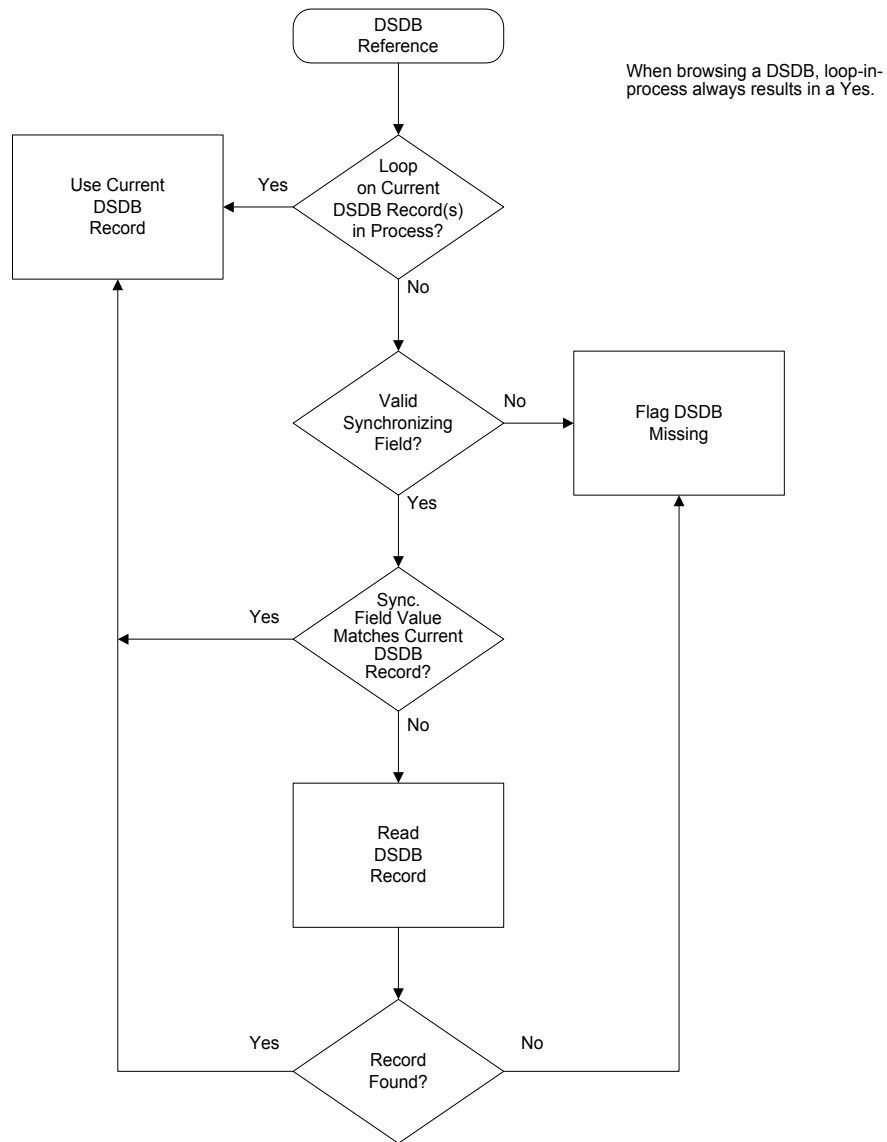


Figure 8-8 The Effect of DSDB Looping on Resynchronization

The application system described in [Chapter 2, “A VISION:Transact Generated Application System”](#) references the item data base as a DSDB. We will describe its use here to illustrate concepts discussed in this section. The source code for the entire system can be found in [Appendix A, “Order/Item Application Screen and Database Definition Code”](#).

The item data base is used in the sample application system as two data bases, FILE1 and FILE2. As FILE1 it is accessed using screen synchronization, while dynamic synchronization is used for FILE2. [Figure 8-9](#) shows the RF statements input to the APPGEN that illustrate this use.

RUN NAME	STMT TYPE	LIB OPEN	SYS DELIM	DEBUG SRC	SRCLIST	SYSLIST	CICS MSG	MSG PROC	POS PCB	RUN TYPE	TRG MON	TRG SYS	PSB NAME	PROGRAM NAME	SEQUENCE 73 TO 80
(M510	,RC)	(#)				(C)			(G)	(C)			(M14CSIT)		
RUN NAME	STMT TYPE	FILE NAME	FILE TYPE	SYNCHRONIZING FIELD 1	SYNCHRONIZING FIELD 2	SYNC QUAL	SYNC TYPE	DL/I PATH	RECORD LENGTH	FLD	STOR TYPE	SQL AUTH-ID			SEQUENCE 73 TO 80
(M510	,RF)	(CUSTOMER)	(FILE)												
(M510	,RF)	(ITEMX)	(FILE1)												
(M510	,RF)	(ITEMX)	(FILE2)	(SYNCFLD1)	(SYNCFLD2)	(T)									

Figure 8-9 Dynamic Synchronization RF Statements

The FD name entry on both RF statements indicates that the same file definition, ITEM, is used for FILE1 and FILE2. The entry of two different names for the synchronizing fields (SYNCFLD1,SYNCFLD2) indicates that a range browse will be used to access FILE2. The T entered for the qualifier indicates that both fields are temporary fields (see) and as such must be initialized in the program to the desired key values prior to referencing a field from FILE2.

DEC 05, 2003 13.18.08														

* REQUEST NAME - TEMPS *														
* INPUT STREAM REQUEST *														

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RVT TMP	LINE NUMBER							
(ER)				(N)										
STMT TYPE	FIELD NAME	FIELD LENGTH	FLD TYP	DEC PLC	OUTPT EDIT LGTH	INITIAL VALUE	COMMENTS							
(TF)	(SYNCFLD1)	(7)	(C)											
(TF)	(SYNCFLD2)	(7)	(C)											
(TF)	(SCREENNO)	(2)	(Z)											
(TF)	(NOSCREEN)	(2)	(Z)											
(TF)	(MSG)	(74)	(C)											
(TF)	(SCRNSLCT)	(1)	(C)											
(TF)	(UPDATE)	(1)	(C)			(U)								
(TF)	(BACKORD)	(1)	(C)			(B)								
(TF)	(MENU)	(1)	(C)			(M)								
(TF)	(ERROR)	(1)	(C)			(E)								
(TF)	(FIRSTOCC)	(2)	(P)											
(TF)	(LASTOCC)	(2)	(P)											
(TF)	(COUNTX)	(2)	(P)											
(TF)	(QANONEND)	(7)	(C)											

Figure 8-10 Defining Synchronizing Fields

The two synchronizing fields are initialized at several points in the system. We will describe one such point associated with the menu screen (ORDMENU4). shows a type 1 screen request (MENUPFCK) for screen ORDMENU4, the first screen displayed in the sample application system.

DEC 05, 2003 13.18.08

* REQUEST NAME - MENUPFCK *
* INPUT STREAM REQUEST *

STMT TYPE	CAT ACT	REQUEST DESCRIPTION				REQ TYP	SCREEN NAME	BACK CONTROL	BR TMP	RVT TMP	LINE NUMBER
(ER)						(1)					
STMT NUMB	SEQ NO.	LOG LEV	CON CTR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	...RESULT... QLF FIELD NAME	PARTIAL FIELD STRT	NUMB	OPR	SEQ NUMBER
(1)	(100)				(R)	(F,ATTNID)	(S,PFKEY)				
(2)	(101)			(F,ATTNID)	(EQ)	(C,PF002)					
(3)	(102)				(NS)	(END)					
(4)	(103)				(R)	(S,ITEMNO)	(T,SYNCFD1)				
(5)	(104)				(R)	(S,ITEMNO)	(T,SYNCFD2)	(1)	(1)	(C)	
(6)	(105)				(R)	(C,999999)	(T,SYNCFD2)	(2)	(6)	(C)	

Figure 8-11 Request Logic to Initialize Synchronizing Range Fields

This logic will be executed whenever screen ORDMENU4 is input to the application. Synchronizing field 1 (T,SYNCFLD1) is initialized to the screen field with the remaining 6 characters set to all 9s. This sets the key ranges for the desired DSDB record and is executed if PF key 2 is pressed from the menu screen. PF key 2 causes the item browse screen (ITMBRWS4) to be displayed next (see).

GLOSSARY FOR SCREEN - ITMBRWS4									
SCREEN FORMAT									
DEC 04, 2003									
PAGE 1									
-----1-----2-----3-----4-----5-----6-----7-----8									
1		ITMBRWS4							1
2									2
3									3
4									4
5		PRODUCT RANGE:		TO				QUANTITY	5
6		ACT ITEM	ITEM	QUANTITY	SHIPPED	ORDERED		BACK	6
7		ION NUMBER	PRICE	ON-HAND	YTD	YTD		ORDERED	7
8									8
9									9
10									10
11									11
12									12
13									13
14									14
15									15
16									16
17									17
18									18
19									19
20									20
21									21
22									22
23									23
24									24
-----1-----2-----3-----4-----5-----6-----7-----8									
SCREEN IDENTIFIER... M510									
TRANSACTION CODE... M502									
FIELD DESIGNATOR.....									
HIGHLIGHT DESIGNATOR... @									
OUTPUT CONTROL..... S									
INITIAL CURSOR POSITION									
LINE: 9 COLUMN: 13									

GLOSSARY FOR SCREEN - ITMBRWS4									
SCREEN CONTENT									
DEC 04, 2003									
PAGE 1									
DATE CATALOGED 12/04/03 20:55:00									
CATALOGED BY									
DATE LAST USED									
EXPIRATION DATE									
DATE LAST UPDATED 12/04/03 20:55:00									
LAST UPDATED BY									

* SEGMENT 1 *									

NUMBER OF SEGMENT OCCURRENCES = 1									
NUMBER OF FIELDS IN SEGMENT = 3									
POSITION OF FIRST FIELD									
LINE: 5 COLUMN: 17									

SCREEN		I	R	E	P	M	N	A	C
FIELD	.SOURCE/DEST.	/	E	H	R	O	U	C	O E/V
NAME	QLF FLD NAME	O	Q	I	R	T	D	M	T N OP

SYNCFLD1	T,SYNCFLD1	O							
	T,SYNCFLD1	I		H	N	N		P	
								NM	SYNCFLD1
	LOC/XATTRS:	LINE	5,	COL	17	=(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)			
SYNCFLD2	T,SYNCFLD2	O							
	T,SYNCFLD2	I		H	N	N		P	
								NM	SYNCFLD2
	LOC/XATTRS:	LINE	5,	COL	28	=(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)			
NONAME	F,MSGLINE	O		H					
	LOC/XATTRS:	LINE	24,	COL	2	=(COLOR=N,HIGHLIGHT= ,PGM SYMBOL=)			

DEC 04, 2003
PAGE 2

NUMBER OF SEGMENT OCCURRENCES = 15
NUMBER OF FIELDS IN SEGMENT = 7

POSITION OF FIRST FIELD
LINE: 9 COLUMN: 13

[illegible]

Dynamic Synchronization Summary

[Figure 8-13](#) highlights the dynamically synchronized data bases accessible by a VISION:Transact generated application.

Access to these retrieval-only data bases is based on the contents of synchronizing fields. Direct synchronization, which retrieves only one record per reference, is performed using the contents of only one field. Browse synchronization, which can retrieve many records per reference, is performed using the contents of either one or two fields. A generic browse uses the contents of one field (which contains a generic key), while a range browse uses the contents of two fields (which contain the low and high ends of the range of records) to accomplish record retrieval.

When a record is not found in a direct DSDB or if a record is not found with a corresponding generic key or range when using browse synchronization, the DSDB is unavailable (missing). See [Processing Invalid, Missing, and Overflow Fields on page 9-9](#).

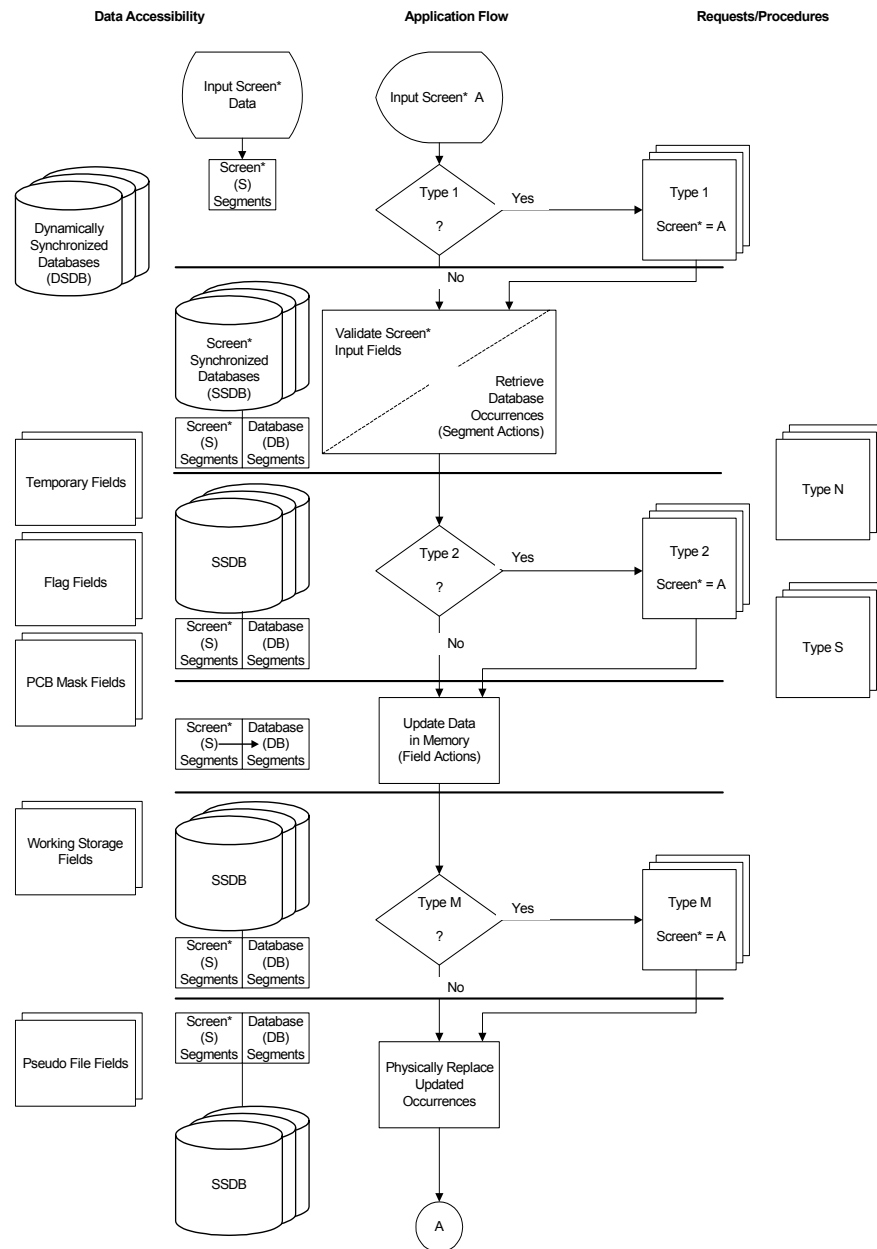


Figure 8-13 Dynamically Synchronized Data Bases in the VISION:Transact Generated Application (Page 1 of 2)

* "Screen" can also be "COMMAREA section".

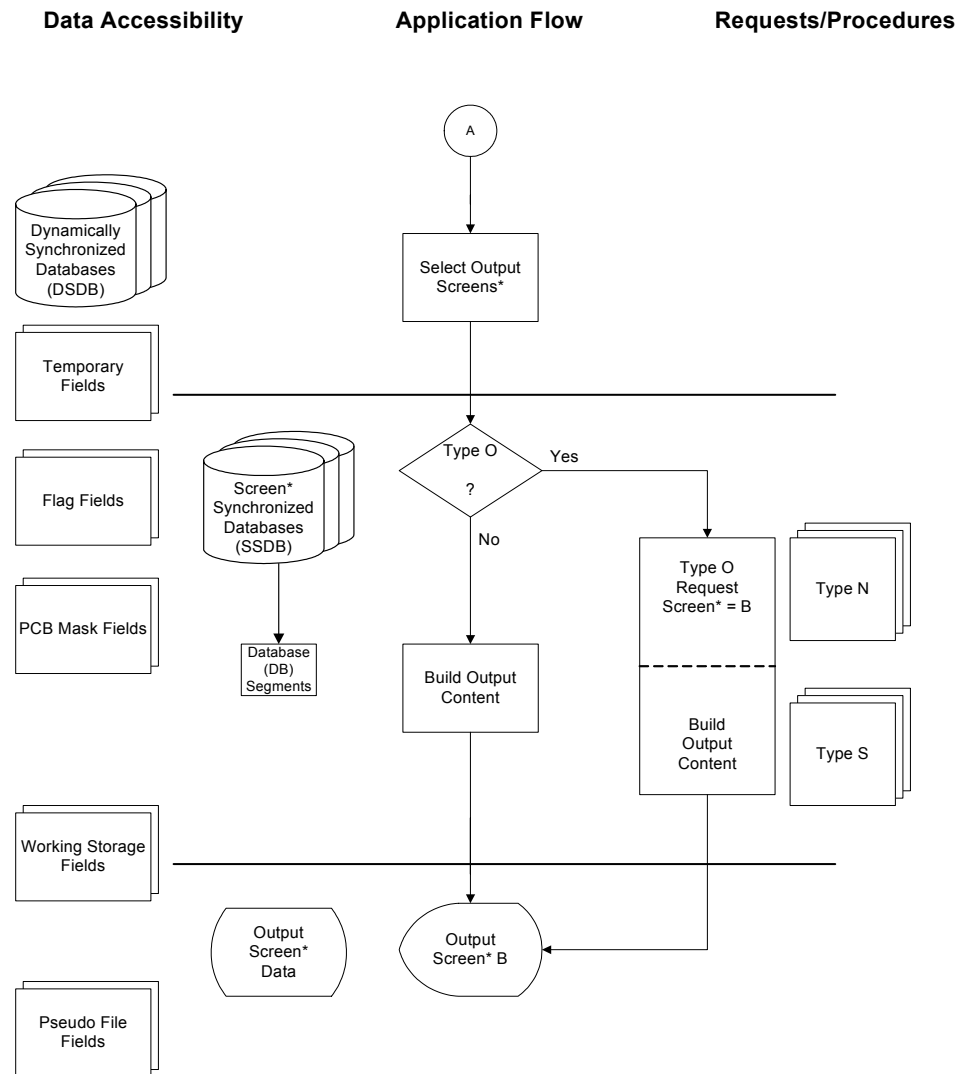


Figure 8-13 Dynamically Synchronized Data Bases in the VISION:Transact Generated Application (Page 2 of 2)

* "Screen" can also be "COMMAREA section".

Data Manipulation/Field Handling

A VISION:Transact generated application performs automatic actions on numeric, alphameric and alphabetic field types.

Using different field types can critically affect application performance. To maximize processing efficiency, logical or arithmetic operations should use fields of the same type. Otherwise, VISION:Transact will generate code to convert the fields according to the rules specified in *Tables 9-1* and *9-2*.

Fields

An alphabetic or alphanumeric field is a character string field - Type C.

Date fields - Type D.

Numeric fields are categorized as follows:

- Zoned decimal fields - type Z.
- Packed decimal fields - type P.
- Fixed-point binary fields - type F.
- Exponential, scientific notation or floating-point fields - type E.

Character String Fields

Character String Fields may consist of any valid EBCDIC character with a length of 1-255 characters. A character field is converted to an appropriate numeric format when it is:

- Used in an arithmetic operation.
- Compared to a numeric field type (Z, P, F, or E).
- Replaced into a numeric field type via field actions on a screen by a request or procedure.

When one of the above operations is performed on a character string field, the VISION:Transact generated application selects the numeric portion of the character string field to use in the operation. The operation is then performed in accordance with the field handling rules in [Field Handling Rules on page 9-3](#). This occurrence should be avoided when possible due to processing inefficiencies and unexpected results.

Date Fields

A field 4 bytes in length representing a Lilian date conforming to IBM Language Environment conventions.

Zoned Decimal Fields

All numeric digits (0-9) are valid in zoned fields. Leading blanks are permitted; embedded blanks (or other non-numeric characters) are invalid. Fields must not exceed 15 digits.

Packed Decimal Fields (Signed)

The maximum size allowed for a packed decimal field is 15 digits (8 bytes of storage).

Fixed Point Fields

Since all bit patterns are valid fixed point numeric, no validation takes place. The maximum fixed point binary field size allowed is 4 bytes; the range of values in 4 bytes is -2,147,483,648 to +2,147,483,647.

Floating Point Fields

A floating point field represents a notation where the decimal number is expressed by an exponent or by a certain power which indicates the actual displacement of the decimal point. For example, a numeric field with a value of 123000000.0 can be expressed in scientific notation as 1.23 X 10⁸ and in exponential notation as 123E6. In VISION:Transact, floating point fields are 4 bytes in length. The approximate range of values allowed in a floating point number (in an IBM System/370 environment) is from 10⁻⁷⁸ to 10⁷⁵, approximately.

Field Handling Rules

The following table summarizes the field conversions for comparisons and arithmetic operations.

Prior to the execution of all logical, arithmetic, and replace/move type operations specified in a request or procedure statements and screen actions specified in S0 or Sn statements, VISION:Transact aligns fields and makes conversions according to *Table 9-1* and the rules which follow it. If an invalid field is detected (see [Processing Invalid, Missing, and Overflow Fields on page 9-9](#)), the intended receiving field remains unchanged for the length of the interaction, and the VISION:Transact application flags the field as invalid.

Field Type 1	Field Type 2	Both Fields are Changed to:	In Accordance with Rule:
C	C	see rule A	A
Z,P,F	C	P	B
Z,P,F	Z	P	C,E
Z,P,F	P	P	C,E
Z,P	F	P	C,E
F	F	F or P	D
C,Z,P,F,E	E	E	F

Rules for comparison, conversion and arithmetic type operations:

1. In comparison or conversion operations involving two character fields of different lengths, the shorter field is padded with trailing blanks before the operation is performed. In arithmetic operations both fields are converted to numeric values in packed decimal format as described in B.
2. In comparison, conversion, or arithmetic operations from a character to a numeric field, the character field is scanned from left to right until the first numeric character or sign (+ or -) is encountered. The scan then continues until the first nonnumeric character other than a properly placed decimal point, comma or exponent sign is encountered or until the limit of 15 digits or the end of the field is reached. Decimal points are aligned, the fields are padded with zeroes, and field type conversions are performed, if necessary, prior to the execution of any operation of these differing types.
3. In comparison, conversion, or arithmetic operations on numeric fields, the fields are aligned on the decimal point. To accomplish this, leading and trailing zeroes are added as necessary; leading and trailing digits may be lost. Field conversions are performed as necessary.
4. Arithmetic operations on two fixed point binary fields are performed in binary format except in the case of a comparison between two fixed point numbers of different scales. In this case, the data type is packed.

5. If numeric fields are a combination of packed decimal, fixed point, and/or zoned decimal formats, or if all fields are zoned decimal, they are converted to packed decimal format before the operation is executed.
6. If one field is not floating point numeric and the other field is floating point, the non-floating point field is changed to floating point format before the operation is executed.

Replace/Move field operations (includes automatic field conversion)

The screen field action (S0 or Sn statement) in VISION:Transact has three data movement operators: R, P, and B. R is used to replace data from a source to a destination field only if the value of the input screen field is non-blank (any screen field in which no data was entered will appear to the application as blank). P is used to replace data from a screen field to a destination field unconditionally. B is used to place the appropriate null value (blanks into character fields, zeroes into numeric fields) into a destination field.

The R operation (requests) or LET command (procedures) indicates unconditional data movement from one operand to another, much like the P action in VISION:Transact Screen Content/ Action Specifications.

In procedures, the sending field can be a field or constant, as in requests, and it may also specify an arithmetic expression. When an arithmetic expression is used as the sending field in a LET command, the field conversion rules for arithmetic expression as shown in *Table 9-1* are first applied. The intermediate result from the arithmetic expression is then moved to the receiving field according to the rules shown in *Table 9-2*. For example, LET CHARFLD=PACKFLD*ZONEFLD+FIXFLD causes the fields in the arithmetic expression to be converted to packed in accordance with Rules C and E, and then the intermediate packed result to be converted to character using Rule G.

Access to IMS data bases, DB2 data bases, and VSAM/ISAM files also involves conversion from the screen field (or synchronizing field for DSDB) to the key/search field of the file.

The VISION:Transact automatic movement and conversion actions are outlined in the following table (remember that screen fields, by definition, are character fields).

Sending Field Type	Receiving Field Type	Movement/ConversionRule:
P,Z,F	C	G
C	C	H
E	C	I
Z	Z	J
C	Z	K
P,F,E	Z	L
E	Z,P,F	M

Rules for automatic movement/conversion

1. When data is replaced or moved from a numeric field (other than floating point) to a character string field, leading zeros are suppressed from the character result field (leading zeros become leading blanks). If there are implicit decimal places in the numeric field, an explicit decimal point appears in the same relative position of the result field. A negative sign, if applicable, appears to the left of the first non-blank character of the result field.
2. When a character string field is filled or replaced from another character field: if the sending field is longer, the right-most characters are truncated from the value stored in the receiving field; if the sending field is shorter, the right-most portion of the receiving field is padded with blanks.
3. Floating-point fields are converted to the exponential notation format when converted to character string fields.
4. When a zoned decimal field is moved into another zoned decimal field, leading blanks are changed to zeroes; e.g., 40F0 is changed to F0C0. Resulting signs in a zoned to zoned move can only be C (positive) or D (negative).
5. When a character field is moved into a zoned decimal field, the result field has a sign of either F (positive) or D (negative).
6. When a packed, fixed, or floating point number is moved into a zoned decimal field, the generated sign of the result field is C or D.
7. A floating-point field, when converted to another numeric field type (zoned decimal, packed decimal, or fixed-point binary), is first converted to fixed point binary format; the intermediate fixed-point field is then converted to the desired field type. Overflow occurs if the converted floating point field exceeds the range of -2,147,483,648 through +2,147,483,647.

General rule: If a packed or zoned decimal field is the result of an arithmetic operation, the generated sign of the result field is always C (positive) or D (negative).

Implicit Definition of Temporary Fields

Temporary fields may be defined explicitly using the temporary field (TF) statement in requests or the FIELD command in ASL procedures. They may be defined implicitly by simply stating the field name with the qualifier, T, on a PR, S0, Sn, or SC statement if not prevented in M5PARM. The TFREQ keyword in M5PARM specifies whether or not TF statements or FIELD commands are required for the definition of temporary fields. (See the *VISION:Transact for CICS Installation Guide* for information on M5PARM.) When a temporary field is defined explicitly (using the TF statement or FIELD command), the programmer assigns attributes to the field as needed. When a temporary field is defined implicitly (by merely referring to it using the T qualifier) VISION:Transact assigns attributes to the field according to the following rules:

- Procedure Statement¹ - implicit temporary field attributes are assigned by the ASL translator according to the order in which expressions are evaluated using the leftmost defined field within the expression. Temporary fields should be explicitly defined with the FIELD command when used in procedures to ensure intended attributes are assigned. If there are no defined fields on a procedure statement, the following default attributes are assigned to the temporary field:
 - field length of eight (8) bytes.
 - field type of P (packed).
 - eight (8) decimal places.
 - output edit length of 19 characters.
- Request PR statement - the temporary field is assigned the same attributes as the leftmost defined field in the same PR statement. A constant in this context is a defined field. If there is no defined field in the PR statement, the following default attributes are assigned to the temporary field:

Note: Temporary fields used with trace operators/commands must be explicitly defined or implicitly defined in a previous non-trace operator/command statement.

 - field length of 8 bytes.
 - field type of P (packed)
 - eight (8) decimal places.
 - output edit length of 19 characters.
- S0 or Sn statement, Source/Destination entry:
 - field length equal to the number of positions reserved for the field on the screen.
 - field type C (character).
 - output edit length equal to field length.
- S0 or Sn statement, Edit/Validation operand:

- field length equal to the number of positions reserved on the screen.
- field type C (character).
- output edit length equal to field length.
- SC statement - Operand A will be assigned the attributes of Operand B field if that is a defined field. Otherwise, default attributes are:
 - field length of 8 bytes.
 - field type P (packed).
 - eight (8) decimal places.
 - output edit length of 19 characters.
- SC statement - Operand B will be assigned the attributes of Operand A if that operand is defined. Otherwise, default attributes are:
 - field length of 8 bytes.
 - field type P (packed).
 - eight (8) decimal places.
 - output edit length of 19 characters.
- SC statement - Operand C:
 - field length of 8 bytes.
 - field type C (character).
 - output edit length of 8 characters.

Definition of Screen\COMMAREA Fields and Virtual Fields

Screen/COMMAREA Fields

Screen fields may be defined and accessed in VISION:Transact to allow the validation and manipulation of screen input data prior to the update of a data base. A screen field is defined in an S0 statement. The first destination field associated with an Sn set for a screen field assigns a name to the screen field. The screen field is referenced in a VISION:Transact request or procedure (type 1, 2, or M only) or in the B operand of subsequent Sn statement by using an S qualifier with the assigned field name. Optionally, screen fields may be assigned a name other than the default name by specifying the NM Edit/Validation operator (illustrated in [Figure 5-5](#)) on the Sn statement for the screen field, followed by the desired name as the B operand. If no NM operator is used and there are multiple screen fields

with the same name (and different qualifiers) associated with a single input screen, any reference to a screen field using that non-unique name will access only the first screen field associated with the name. Further, if the NM operator is used to assign a name to a screen field, the name assigned should be unique within the content/action specifications for that input screen.

Screen fields are always character string fields (type C). All of the field handling rules described in [Field Handling Rules on page 9-3](#) which apply to character fields also pertain to screen fields.

Screen fields may be accessed and modified (using the S qualifier) during input screen processing. Should an error be detected during this input screen processing, the input screen is redisplayed, normally with the input screen fields in their original unmodified form to avoid confusing the terminal user. You may, however, choose to permit a screen field to be redisplayed as modified by the application. For example, this facility can be used for optional display of special diagnostic messages or verification information.

When a screen field is used with a segment action or is a synchronizing field for a DSDB, it is converted to the type, length, and scale of the key/search field. If the key is packed, signed packed is assumed. When the key is zoned, character screen field to zoned conversion is performed resulting in an unsigned zoned field.

Virtual Screen/COMMAREA Fields

A virtual screen field is a screen field whose contents do not appear on the screen. Virtual screen fields are defined on an S0 statement whose V/S/A entry is V. They may be input or output depending on the I/O specification on that S0 statement. They are character string fields, as are all screen fields.

Input virtual screen fields facilitate the manipulation of screen input data by the application. The length of an input virtual screen field is obtained from either the length of its destination field or the length assigned using the LN operator. Likewise, the name of the field is derived from either the name of the destination field or the name assigned using the NM operator.

Output virtual screen fields can be used to drive looping during output processing. During output processing, VISION:Transact applications will loop through a multiply occurring screen segment to load fields only if at least one screen field in that screen segment specifies a field from a multiply occurring data base segment as its source. What this means is that VISION:Transact applications will not automatically loop through a screen segment that contains only temporary fields or working storage fields. In other words, looping is not driven by the structure of the screen (as it is during input processing), but by the data available to display.

One solution to this problem is to include (in the lower level screen segment) any field from a multiply occurring data base segment, and make it a non-display field. However, this takes room on the screen and requires the application to perform additional unnecessary data movement. A better solution is to include an output virtual screen field in the screen segment whose source is a field from a multiply occurring segment. Because output virtual screen fields do not display on the screen, no space is needed for unnecessary data, and data movement is reduced.

Processing Invalid, Missing, and Overflow Fields

When a VISION:Transact generated application encounters an invalid field, it is flagged as invalid but its original contents are not changed. This property is propagated and other result fields dependent upon the invalid field are flagged as invalid. Again, field contents are not destroyed. When there is an attempt to display an invalid field, a special character is displayed.

In the course of processing a field can become invalid. This can occur for a variety of reasons, such as:

- Division by zero.
- Field overflow.
- Illegal field data.
- Replace operator (requests) or LET command (procedures) from a missing or invalid source.

An invalid data base, pseudo file, or screen field remains invalid for the duration of a request or procedure or until the next occurrence is accessed. Thus, a field which is flagged invalid in Request 1 and hence is not available to Request 1, is available to subsequent requests or procedures with no indication that it was ever invalid. The exception to this rule is that of temporary, working storage and flag fields. Since temporary and flag fields are designed to communicate between requests and procedures, they carry their property of invalidity between requests and procedures. All invalid fields may be revalidated by storing valid data into the field. Re-validation does not occur if partial fielding is used in the storing of valid data into the field, even if the partial field specification defines the entire field.

If a destination field is invalid due to a field action specified on the screen definition, the input screen is re-displayed with an error message. If a field is invalid as a result of a request or procedure operation, the field is flagged as invalid.

A replacement, movement, or arithmetic type operation using a missing field will cause a result field to go invalid. A missing field is a field in a data base segment which does not exist or cannot be referenced. For example, a field in a segment occurrence which has been deleted is missing if that field is referenced in a type M request or procedure, because type M executes after the intermediate structure has been updated and the data base segment occurrence deleted from the composite segment. Also, if an attempt at dynamic synchronization fails, fields from the dynamically synchronized data base (DSDB) are missing until the dynamic synchronization succeeds on subsequent attempts.

Testing for invalid or missing fields using EQ, LT, GT, LE, or GE relational operators produces a false condition if the test involves an invalid or missing field. If a field is tested using an NE relational operator, and the field is invalid or missing, the test produces a true condition. During request and procedure processing, fields should be checked for validity and existence. This is done by comparing the field to itself (FLDA EQ FLDA). If the field is valid and exists, the test is true.

VISION:Transact assigns standard default characters to display in place of invalid or missing fields. These characters may be changed as an installation option. The following is a list of the characters supplied as standard:

- * The field is invalid as a result of computation during request or procedure processing. An asterisk will also print if a field is specified as format Z or P on a file definition, but the data in the field does not match the given format.
- This field does not exist in the record (missing).
- + This field is too large to display in the layout provided (overflow).

Invalid, Missing, Overflow, or Error Indicators in COMMAREA Fields

In CICS COMMAREA applications, each field in the COMMAREA is preceded by a 3-byte attribute prefix that is used to communicate whether a field is invalid, missing, or when overflow occurs. The 3-byte attribute prefix field will contain special attribute codes when the application program returns the COMMAREA to the calling program. This attribute prefix will be blank when the field is processed as expected and no messages are issued relating to the field. Otherwise it will contain one of the following codes:

- INV - The field became invalid during internal processing. The field itself will contain an asterisk (*) character.
- MIS - The field is missing in the source database. The field itself will contain the dash (-) character. This can occur when the record (VSAM), segment (IMS), or table row (DB2) for the field does not exist in the source database.

- OVF - An overflow occurred when converting the source field to the COMMAREA field on output. The field itself will contain a plus (+) character. This will never occur if the source field is already a character type field, even if the field must be truncated in order to fit.
- A message id such as Fnn, Snn, or Xnn indicating that the error message in the MSGLINE field applies to this field. The error codes are described in the Online Execution Messages section of the Messages and Codes Manual. Note that the code F00 is used for messages that result from the MS operator on a PR statement. The code F02 is used for messages that result from the MS operator on an Sn statement. Identifying the field with the F00 attribute requires that the SET ERROR command (in ASL) or the SE operation (PR statement) is performed.

A MSGLINE field is added to a COMMAREA section just as it is to a 3270 screen when the field is not explicitly defined for the COMMAREA section. The MSGLINE field will contain the text of the error message that is output, whether from the standard system messages or from user specified coding. This field will contain blanks if no message is output.

The attribute prefix for the MSGLINE field will contain MSG if a message has been generated. If no message has been generated but one of the special attribute codes (INV, MIS, or OVF) has been set for a field, the MSGLINE field attribute prefix will contain the first such code that was applied. Thus, when the COMMAREA is returned to its original source, only the MSGLINE field attribute prefix need be examined to determine if any other fields need be examined for error conditions.

Following is an example of a COMMAREA layout showing the attribute prefix and MSGLINE fields for a simple application.

```

-----1-----2-----3-----4-----5-----6-----7-----
-----0-----+-----0-----+-----0-----+-----0-----+-----0-----+-----0-----
01 DFHCOMMAREA.
*
    05 IVP2SCN1.
      10 DIVIDEND-ATTR                                PIC XXX.
      10 DIVIDEND                                      PIC X(05).
      10 DIVISOR-ATTR                                PIC XXX.
      10 DIVISOR                                      PIC X(05).
      10 QUOTIENT-ATTR                              PIC XXX.
      10 QUOTIENT                                    PIC X(08).
      10 MSGLINE-ATTR                                PIC XXX.
      10 MSGLINE                                      PIC X(79).
      10 TRANSACTION-CONTROL REDEFINES MSGLINE.
          15 ATTNID                                    PIC X(05).
          15 FILLER                                    PIC X.
          15 CONVERSE                                PIC XXX.
          15 FILLER                                    PIC X.
          15 CONVERSATION-ID                          PIC X(12).

```

Figure 9-1 COMMAREA Layout

Defining Data

Before a VISION:Transact generated application can access a data base, that data base must be mapped in a VISION:Transact file definition. Recall that the term *data base*, as used in this guide, refers to any external file requiring I/O operations to be accessed from an application. These data bases may only be IMS data bases (including fast path data bases) and DB2 relational data bases for an IMS TM application, while they can include VSAM, ISAM, IMS (or DL/I under DOS/VSE) data bases, and DB2 (or SQL/DS under DOS/VSE) relational data bases for a CICS application.

Note: ISAM files were desupported in release 1.7 of CICS.

The definitions of each of these data base types have many similarities, as well as differences. We will discuss the similarities first.

The structure commonly mapped in a VISION:Transact file definition is that of hierarchy. A hierarchy depicts a one-to-many relationship between the data at the highest level and the multiply occurring data at the subordinate levels. This relationship is clarified by [Figure 10-1](#), if you keep in mind that many orders could exist for each customer and many items could be on each order.

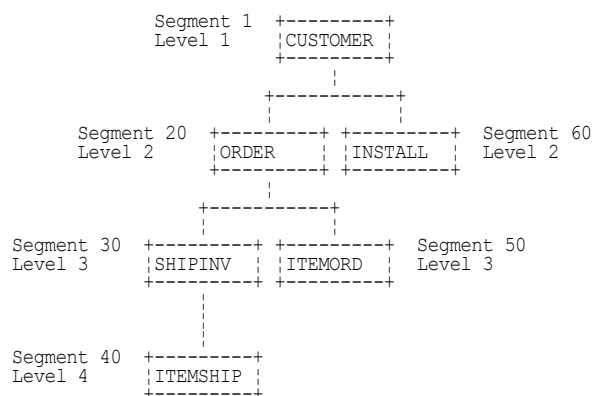


Figure 10-1 A Hierarchical VISION:Transact File Definition

CUSTOMER is at the highest level in the structure illustrated; this highest level is known as the root of the hierarchy. All data below the root is dependent on the root. That is, only when CUSTOMER exists do ORDER and INSTALL exist. This dependency persists all the way down the hierarchy. This dependency between levels can be thought of as a one-to-many relationship between the multiply occurring data at lower levels and the data at the next highest level. Thus, a hierarchy contains a collection of dependent data at different hierarchical levels. Note that an occurrence of data is known as a segment occurrence, while all occurrences of a particular group of information is known as a segment. For example, order number 12340 would be a segment occurrence, while ORDER is a segment. It contains information that solely applies to an order. Each segment is numbered, with respect to the hierarchy, top to bottom left to right. [Figure 10-1](#) also illustrates this numbering of segments and levels. As you can see each layer of the hierarchy receives a level number. The level numbers are assigned consecutively from top to bottom in ascending order.

When mapping the data, whether it has a hierarchical structure or not, each segment must have a key assigned to it. This key allows the VISION:Transact generated application to find and access a particular piece of data requested. This key must be assigned on a VISION:Transact file definition whether the physical file structure requires one or not.

Three statement types are used to define most data structures to VISION:Transact - FD, LS, and L0 (specified using the File Definition Worksheet shown in [Figure 10-3](#)). The FD statement identifies the overall characteristics of the data structure being defined. Here you specify the name under which the definition is cataloged on the VISION:Transact library, the record format of the data base, and either a record size or buffer size. You can also specify the sequencing of the glossary (formatted listing) of this file definition.

With an LS statement you assign a name to a data base segment. This name is a required entry for an IMS (or DL/I) data base and a DB2 (or SQL/DS) relational data base. You also assign a segment number on the LS statement. Segment numbers need not be contiguous as long they are in ascending sequence within the hierarchical structure (top to bottom, left to right). A maximum of 99 segment types may be defined in a VISION:Transact file definition. You must specify segment numbers on the L0 statement as well. These numbers are necessary for VISION:Transact to generate its appropriate data base calls to map the data. A description of these generated calls can be found in the *Advantage VISION:Transact for CICS Environment Guide*. You may also specify the order of segment occurrences within a segment. You may specify either ascending or descending or default to unordered. Note that for IMS data bases, this entry is ignored. For DB2 relational data bases, the LS statement also names the DB2 table, view, or synonym which is to be used to map the VISION:Transact segment.

On an L0 statement you specify the name and attributes of a field. Names must be unique across the file definition. You also assign a level number; this number indicates the level at which the field exists in the hierarchy. For example, the root segment is always at level one, segments directly subordinate to the root are at level two, segments directly subordinate to level two are at level three, etc., to a maximum of nine levels. Level numbers always begin with one and must be contiguous. The attributes you specify include the field's location within the segment, its length and data type, and whether the field is a key field or not. Note that you may have up to eight key fields assigned to a segment. For IMS data bases you may also specify a field as a search field. For DB2 relational data bases, the L0 statement names the table column which is to be used to map data for the associated VISION:Transact field.

An additional statement type (LR) is used in the definition of DB2 relational data bases. The LR statement is used to define the logical relationships between DB2 tables used as segments in a logical record. The LR statement explicitly establishes the relationship between the rows of the DB2 tables within the relational data base hierarchy.

VISION:Transact and IMS

In processing an IMS data base, VISION:Transact uses the file definition to relate the structure it will process to the one known to IMS. Since both VISION:Transact and IMS use similar hierarchical structures, relating the two becomes a matter of mapping the IMS structure in VISION:Transact terms.

It is a good practice to maintain only one VISION:Transact file definition for a data base. Since each application determines its own view of the data base (the PCB), a single file definition for each DBD is usually sufficient. If field sensitivity is used, the segment defined to VISION:Transact should consist of the sensitive fields concatenated rather than the entire segment.

The choice of segment numbers is arbitrary provided the IMS hierarchical structure is maintained. The VISION:Transact segment names must be the same as the IMS names since VISION:Transact uses that information in generated DL/I calls. The VISION:Transact segment keys and search fields must also use the same names as defined in the DBD.

Execution tracing, described in [Chapter 12, "Execution Tracing"](#), can be used to trace DL/I calls issued by the generated applications.

Writing the File Definition

Writing a VISION:Transact file definition from a DBD is a straightforward process. First, specific entries are made on the File Definition Form which result in a skeletal definition. The definition is then completed based on the content of the data base segment. This content must come from something similar to a record layout, since that information is not necessarily included in IMS control blocks. It may be helpful to refer to [Figure 10-2](#) and [Figure 10-3](#) during the following discussion.

Required VISION:Transact entries depend on the actual definition of the data and must be entered by you when writing your own file definition. The first entries are made on the FD statement. The file identification is the DBD name and comes from the NAME parameter on the DBD statement (see [Figure 10-2](#)). Because it is a data base file, the record format entry must be D. The buffer size entry is based on an estimate of the average storage size required to accommodate a working set of segments during an interaction.

Note: VISION:Transact validates this entry but does not use it. This entry is required by other Computer Associates products that may use the file definition.

An LS statement must be entered for each segment used in the data base. The segment name entry in each LS statement is the value from the NAME parameter of the SEGM statement.

Segments are now numbered using the DBD in [Figure 10-2](#), and the values entered in the segment number specification. The starting segment number and the increment between segment numbers are arbitrarily assigned. However, the sequence is determined by the order of the SEGM statements in the DBD (top to bottom, left to right) with the first segment having the lowest number. This completes the LS statement.

Each FIELD statement in the DBD should have a corresponding L0 statement in the VISION:Transact file definition, with a matching field name, location, and data type/length/scale. VISION:Transact uses these field definitions to build SSAs for the screen synchronization process, according to the segment actions provided in the screen processing definition. In addition, the ALIAS entry on the L0 statement allows you to interface with an IMS file. While an IMS file must have unique field names within a segment, VISION:Transact requires unique field names within the entire file definition. The IMS field entry is identified as the ALIAS on the L0 statement. The segment key entry on the L0 statement identifies fields that are defined in the DBD, and thus eligible for use with segment actions on the screen processing definition. Three types of segment key entry are permitted.

- A numeric segment key entry identifies a required key field, with the lowest number being the major key. VISION:Transact requires segment actions on the screen definition for all fields in a segment defined as segment keys.

- An S in segment key identifies a search field, which can be used for segment qualification but is not required for segment access. These fields may be referenced with either segment or field actions, depending on the needs of a particular application.
- A V in segment key identifies a virtual key field, which is a means of defining a field name and data attributes to be used for SSAs but not contained within the physical data base segment. Virtual keys are useful for cases in which the segment key is non-contiguous, or external to the target segment.

```

1.          DBD          NAME=CUSTOMDB, ACCESS= (HISAM, ISAM)
2.          DATASET     DD1=CUSTOMER, DEVICE=3350, OVFLW=CUSTOMOV
3.          *           3350 DISK STORAGE
4.          *
5.          PRINT       NOGEN
6.          SEGM        NAME=CUSTOMER, PARENT=0, BYTES=45, FREQ=50
7.          FIELD       NAME= (CUSTNO, SEQ, U) , BYTES=5, START=1, TYPE=C
8.          FIELD       NAME=CUSTNAME, BYTES=30, START=6, TYPE=C
9.          *
10.         SEGM        NAME=ORDER, PARENT=CUSTOMER, BYTES=39, FREQ=20
11.         FIELD       NAME= (ORDERNO, SEQ, U) , BYTES=5, START=1, TYPE=C
12.         *
13.         SEGM        NAME=SHIPINV, PARENT=ORDER, BYTES=98
14.         FIELD       NAME= (SHIPNO, SEQ, U) , BYTES=4, START=1, TYPE=C
15.         *
16.         SEGM        NAME=ITEMSHIP, PARENT=SHIPINV, BYTES=24
17.         FIELD       NAME= (ITEMSHIP, SEQ, U) , BYTES=7, START=1, TYPE=C
18.         *
19.         SEGM        NAME=ITEMORD, PARENT=ORDER, BYTES=57
20.         FIELD       NAME= (ITEMORD, SEQ, U) , BYTES=7, START=1, TYPE=C
21.         *
22.         SEGM        NAME=INSTALL, PARENT=CUSTOMER, BYTES=81
23.         FIELD       NAME= (INSTNO, SEQ, U) , BYTES=4, START=1, TYPE=C
24.         *
25.         DBDGEN
26.         FINISH
27.         END

```

Figure 10-2 Sample DBD

Any segment that is to be screen synchronized must have at least one FIELD statement in the DBD, since an SSA will be built from the segment action specification in the screen processing definition. VISION:Transact requires at least one key field to be defined in each segment.

Level numbers may now be entered on the L0 statements. The segment (SEGM statement) with the PARENT=0 entry is assigned level 1, and that entry is made in the level number specification of the L0 statement defining its key field. All segments having the level 1 segment specified as their parent on SEGM statements are assigned level 2. This process continues, always assigning a level number one greater than that of the parent.

The length of a segment is computed by VISION:Transact based on the starting location and length of the defined fields for the segment. The length of a segment computed by VISION:Transact must agree with the length defined to IMS in the DBD. Therefore, the last byte in each segment must be included in a field definition. When using a physical DBD the BYTES keyword of the SEGM statement explicitly states the segment length. If a logical DBD is used, segment length is a function of the SOURCE keyword of the SEGM statement.

When the SOURCE for a logical segment is a single segment, the logical segment length is stated by the BYTES keyword in the physical DBD which defines that segment. When the SOURCE keyword names two segments, a logical child and either its physical or logical parent are concatenated to form the defined segment. How much of each segment is included in the concatenation depends on how the SOURCE statement is specified. In general, the segment will contain the key of the logical/physical parent followed by the logical child segment. If the DATA keyword is coded for the SOURCE statement which describes the parent, the entire parent segment follows the logical child.

Field name entries (additional L0 statements) for fields not referenced by the DBD may be supplied and are subject only to VISION:Transact naming conventions. Segment number and level number entries are the same as those for the corresponding key field. With our Customer data base, each segment has ten unused bytes at its end. These last ten bytes for each segment may now be defined as fields, completing the skeletal definition, and insuring that the segment lengths correspond to those specified in the DBD. [Figure 10-3](#) shows the completed skeletal definition. Any additional fields to be addressed by applications may now be added to this definition by assigning them a name and giving them the proper location within their respective segments, length, and data type.

FILE NAME	STMT TYPE	FILE IDENT	DELETE FILE	GLOSSARY LEVEL	RECORD FORMAT	RECORD SIZE	RECORDS PER BLK	BUFFER SIZE			
(CUSTOMER)	(FD)	(CUSTOMDB)	(R)	(B)	(D)			(5300)			
FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMBR	VRT SEG		SUP DPL	SEG KEY	KEY CHAR		
(CUSTOMER)	(LS)	(CUSTOMER)		(01)							
(CUSTOMER)	(LS)	(ORDER)		(20)							
(CUSTOMER)	(LS)	(SHIPINV)		(30)							
(CUSTOMER)	(LS)	(ITEMSHIP)		(40)							
(CUSTOMER)	(LS)	(ITEMORD)		(50)							
(CUSTOMER)	(LS)	(INSTALL)		(60)							
FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMBR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC
(CUSTOMER)	(L0)	(CUSTNO)		(01)	(1)	(0001)	(005)	(C)	(1)		
(CUSTOMER)	(L0)	(CUSTNAME)		(01)	(1)	(0006)	(030)	(C)			
(CUSTOMER)	(L0)	(ORDERNO)		(20)	(2)	(0001)	(005)	(C)	(1)		
(CUSTOMER)	(L0)	(SHIPNO)		(30)	(3)	(0001)	(004)	(C)	(1)		
(CUSTOMER)	(L0)	(ITEMSHIP)		(40)	(4)	(0001)	(007)	(C)	(1)		
(CUSTOMER)	(L0)	(ITEMORD)		(50)	(3)	(0001)	(007)	(C)	(1)		
(CUSTOMER)	(L0)	(INSTNO)		(60)	(2)	(0001)	(004)	(C)	(1)		
(CUSTOMER)	(L0)	(INSTCONT)		(60)	(2)	(0005)	(003)	(C)			
(CUSTOMER)	(L0)	(SEG1FIL)		(1)	(1)	(0036)	(010)	(C)			
(CUSTOMER)	(L0)	(SEG20FIL)		(20)	(2)	(0030)	(010)	(C)			
(CUSTOMER)	(L0)	(SEG30FIL)		(30)	(3)	(0089)	(010)	(C)			
(CUSTOMER)	(L0)	(SEG40FIL)		(40)	(4)	(0015)	(010)	(C)			
(CUSTOMER)	(L0)	(SEG50FIL)		(50)	(3)	(0048)	(010)	(C)			
(CUSTOMER)	(L0)	(SEG60FIL)		(60)	(2)	(0072)	(010)	(C)			

Figure 10-3 Skeletal VISION:Transact File Definition

After the source statements (shown in [Figure 10-3](#)) are completed, this definition is ready to be stored on the VISION:Transact library or included as an instream file definition in an APPGEN. See [Chapter 3, “Essentials of Data Definition”](#) and [Chapter 4, “Application Flow”](#) of the *Advantage VISION:Transact for CICS Environment Guide* for details on storing definitions on the VISION:Transact library or including them in an APPGEN.

VISION:Transact Segment Actions and SSAs

The VISION:Transact generated application builds a segment search argument (SSA) for every segment action specified in a screen processing definition. In unusual cases where you wish to build multiple SSAs for a single destination segment, you can specify multiple key fields for that segment in your VISION:Transact file definition. To facilitate the building of multiple SSAs by your application, you must specify a segment action for each field that is defined as a key in your VISION:Transact file definition.

When you want to build a single SSA from multiple fields, you can define a virtual key field to VISION:Transact. Defining a virtual key to VISION:Transact allows you to access non-uniquely keyed root segments, root segment keys that do not physically reside within the root segment (i.e., with a secondary index) and root keys composed of non-contiguous fields (called scattered keys).

A virtual key is specified by entering a V in segment key entry on the L0 statement for the root key field. The field location is left blank and the length, as specified in the DBD, is entered. The field type must be character. No other key field, except search fields, may be named for this segment. The virtual key field name must correspond to the XDFLD name in the DBD. A virtual key is valid for root segments only.

A segment with a virtual key will be handled by VISION:Transact as follows:

1. During VISION:Transact processing, the virtual key will be available in storage as if it were part of the data base segment.
2. During DL/I calls, the virtual key will not be included in the segment and therefore will not appear in the DL/I file on disk.
3. A virtual key is treated like any other segment key by VISION:Transact. It may appear on a screen processing definition statement (S0/Sn) and in screen requests or procedures (type 2, M, and O).

The use of a virtual key with a secondary index is described later in this section.

IMS Secondary Indexing

In general a different VISION:Transact file definition will be required for each secondary index. The following steps are a guideline for preparing the file definition:

1. Examine a listing of the PCB to be used. If the PCB statement includes a PROCSEQ parameter, secondary indexing is assumed for the data base and an appropriate file definition should be prepared. A listing of the actual PCB to be used should be referenced for this purpose. The sample PCB listing generated by the VISION:Transact APPGEN does not contain secondary index specifications.
2. A PROCSEQ parameter indicates a secondary data structure. The PCB (not the DBD) reflects the hierarchical structure of the data base. The new file definition should map this structure exactly.
3. Segment names for the LS statements of the file definition should be the same as on the PCB SENSEG statements.
4. Non-key fields can be defined in the normal manner, following the file definition specifications described previously.
5. Key and search field names, locations, lengths, and field types can be determined in the normal manner from the DBD for all segment types except the root of the secondary data structure.
6. When PROCSEQ is specified, the root segment of the secondary data structure is always accessed through a secondary index. The record key field described in the VISION:Transact file definition must be the XDFLD of the index. The characteristics of this XDFLD can be determined in the following manner:
 - a. Examine a listing of the DBD named in the DBDNAME or NAME parameter of the PCB statement. Locate the SEGM statement which names the root segment of the secondary data structure. This SEGM statement describes the index target segment type for the structure, and will be followed by at least one LCHILD statement before any other SEGM statements appear in the listing.
 - b. From among these LCHILD statements, locate the one that names the secondary index for the structure. The secondary index name will be the second subparameter of the NAME parameter of the LCHILD statement. For example:

```
LCHILD    NAME= (PARTPTR,PARTINDX) , . . .
```

secondary index name

PARTINDX will be the name specified by the PCB PROCSEQ parameter. Next locate the first XDFLD statement following this LCHILD statement.

- c. The NAME parameter of this XDFLD statement specifies the XDFLD name to use as record key in the VISION:Transact file definition.
- d. The FIELD location in the L0 statement must be left blank for a virtual key.

- e. The length of the virtual key is the combined length of all fields specified by the SRCH parameter of the XDFLD statement. Individual field lengths can be determined by finding each field's BYTES value among the FIELD statements for the index source segment type (which is the segment named by the SEGMENT parameter of the XDFLD statement). For example:

```

SEGMENT  NAME=PARTS
-
LCHILD  NAME=(MANUPTR,MANUINDX), . . .
XDFLD   NAME=(MANUNAME, SEGMENT=MANUFACS,
          SRCH=(MANU,MDESC)
          ↑
          index source
-
-
SEGMENT  NAME=MANUFACS ← segment name
-
-
FIELD    NAME=MANU, START = 5, BYTES=20 ← Search fields
FIELD    NAME=MDESC, START=25, BYTES= 5 ←
          25 (sum of BYTES)
          ↑
          virtual key length

```

- f. The field type for the record key is always character string. This is true regardless of characteristics of the individual fields that make up the XDFLD.

FILE NAME	STMT TYPE	FILE IDENT	DELETE FILE	GLOSSARY LEVEL	RECORD FORMAT	RECORD SIZE	RECORDS PER BLK	BUFFER SIZE			
(PARTMAST)	(FD)	(PARTDBD)	(R)		(D)			(5000)			
FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMBR	VRT SEG		SUP DPL	SEG KEY	KEY CHAR		
(PARTMAST)	(LS)	(PARTS)		(1)							
FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMBR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC
(PARTMAST)	(L0)	(MANUNAME)		(1)	(1)		(25)	(C)	(V)		
FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMBR	VRT SEG		SUP DPL	SEG KEY	KEY CHAR		
(PARTMAST)	(LS)	(MANUFACS)		(2)							
FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMBR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC
(PARTMAST)	(L0)	(MANUNO)		(2)	(2)	(1)	(4)	(C)	(1)		
(PARTMAST)	(L0)	(MANU)		(2)	(2)	(5)	(20)	(C)			
(PARTMAST)	(L0)	(MDESC)		(2)	(2)	(25)	(5)	(C)			

Figure 10-4 Sample of a File Definition Using a Virtual Key

IMS Fast Path Data Bases

VISION:Transact supports both types of Fast Path data bases: Main Storage Data Base (MSDB) and Data Entry Data Base (DEDB). Because Fast Path is strictly a capability of IMS TM, the target environment for a Fast Path application cannot be CICS.

An MSDB consists of fixed length, root-only records and may be terminal related or non-terminal related. The record key of a terminal related MSDB is the LTERM name of the terminal although the record key (LTERM name) is not contained in the record. The LTERM name associated with the terminal determines which terminal related MSDB record may be updated by a transaction. If an MSDB record isn't associated with an LTERM, that record will be "non-owned" and available for retrieval only. Defining the terminal related MSDB to VISION:Transact requires a RECORD FORMAT entry of 1 on the FD statement and a virtual key designation on an L0 statement (Figure 10-5).

FILE NAME	STMT TYPE	FILE IDENT	DELETE FILE	GLOSSARY LEVEL	RECORD FORMAT	RECORD SIZE	RECORDS PER BLK	BUFFER SIZE			
(MSDBA)	(FD)	(MSDBA)	(R)		(1)			(200)			
FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMBR	VRT SEG		SUP DPL	SEG KEY	KEY CHAR		
(MSDBA)	(LS)	(SEG)									
FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMBR	LEV NUM	FIELD LOCATN	FIELD LNNGTH	FLD TYP	SEG KEY	FLD RND	DEC PLC
(MSDBA)	(L0)	(TERM)				(8)	(C)	(V)			
(MSDBA)	(L0)	(TELLER)			(1)	(20)	(C)				
(MSDBA)	(L0)	(COUNT)			(21)	(2)	(F)				
(MSDBA)	(L0)	(AMOUNT)			(23)	(5)	(P)				(2)

Figure 10-5 Sample of a Terminal-Related MSDB File Definition

The name of the key field is assigned on the REL= statement of the DBD. A non-terminal related MSDB record key is located in the record. It must be the first field in the record and cannot exceed 240 bytes in length (Figure 10-6). Although any LTERM may update any record in a non-terminal related MSDB, records may not be inserted or deleted.

FILE NAME	STMT TYPE	FILE IDENT	DELETE FILE	GLOSSARY LEVEL	RECORD FORMAT	RECORD SIZE	RECORDS PER BLK	BUFFER SIZE			
(MSDBB)	(FD)	(MSDBB)	(R)		(1)			(200)			
FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMBR	VRT SEG		SUP DPL	SEG KEY	KEY CHAR		
(MSDBB)	(LS)	(SEG)		(1)							
FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMBR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC
(MSDBB)	(L0)	(SEGKEY)		(1)	(1)	(1)	(16)	(C)	(1)		
(MSDBB)	(L0)	(NAME)		(1)	(1)	(17)	(20)	(C)			
(MSDBB)	(L0)	(BALANCE)		(1)	(1)	(37)	(4)	(F)			(2)
(MSDBB)	(L0)	(DATE)		(1)	(1)	(41)	(6)	(C)			

Figure 10-6 Sample of a Non-Terminal-Related MSDB File Definition

A DEDB provides efficiency when a high volume of segment insertion is required by an application. One of the level two segments may be defined as the sequential dependent. All dependents are maintained in chronological sequence regardless of their root. If a root is deleted, so are all of its sequential dependents. Defining a DEDB to VISION:Transact requires a Record Format entry of 2 on the FD statement and S in Segment Key on the LS statement for the sequential dependent (Figure 10-7).

FILE NAME	STMT TYPE	FILE IDENT	DELETE FILE	GLOSSARY LEVEL	RECORD FORMAT	RECORD SIZE	RECORDS PER BLK	BUFFER SIZE			
(DEDBX)	(FD)	(DEDBX)	(R)		(2)			(200)			
FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMBR	VRT SEG		SUP DPL	SEG KEY	KEY CHAR		
(DEDBX)	(LS)	(ROOT)		(1)							
FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMBR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC
(DEDBX)	(L0)	(ACCNTNO)		(1)	(1)	(1)	(16)	(C)	(1)		
(DEDBX)	(L0)	(NAME)		(1)	(1)	(17)	(30)	(C)			
(DEDBX)	(L0)	(BALANCE)		(1)	(1)	(47)	(5)	(P)			(2)
FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMBR	VRT SEG		SUP DPL	SEG KEY	KEY CHAR		
(DEDBX)	(LS)	(DSEG1)		(2)							
FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMBR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC
(DEDBX)	(L0)	(RELACCT)		(2)	(2)	(1)	(16)	(C)	(1)		
FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMBR	VRT SEG		SUP DPL	SEG KEY	KEY CHAR		

Figure 10-7 Sample of a DEDB File Definition (Page 1 of 2)

NAME	TYPE	NAME	ETE	NMBR	SEG	DPL KEY CHAR					
(DEDBX)	(LS)	(DSEG2)	(	3)						(S)	
FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMBR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC
(DEDBX)	(L0)	(TYPE-DW)	(	3)	(2)	(	1)	(	1)	(C)	
(DEDBX)	(L0)	(AMOUNT)	(	3)	(2)	(	2)	(	5)	(P)	(2)
(DEDBX)	(L0)	(DATE)	(	3)	(2)	(	7)	(	6)	(C)	
(DEDBX)	(L0)	(TIME)	(	3)	(2)	(	13)	(	6)	(C)	
(DEDBX)	(L0)	(IDENT)	(	3)	(2)	(	19)	(	5)	(C)	(1)
FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMBR	VRT SEG	SUP SEG KEY DPL KEY CHAR					
(DEDBX)	(LS)	(DSEG3)	(	4)							
FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMBR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC
(DEDBX)	(L0)	(FLDA)	(	4)	(2)	(	1)	(	5)	(C)	
(DEDBX)	(L0)	(FLDB)	(	4)	(2)	(	6)	(	3)	(C)	(1)

Figure 10-7 Sample of a DEDB File Definition (Page 2 of 2)

A VISION:Transact Fast Path application must satisfy the following IMS restrictions:

1. The application may not be conversational.
2. Each input screen results in only one output screen. (An A in Output Control on the SD statement is not acceptable in Fast Path applications.)
3. Output from the application is sent to the I/O PCB (originating terminal) or an alternate response PCB.
4. Multiple message processing must be specified on the RC statement.

For further details on defining Fast Path data bases and applications, refer to the *Advantage VISION:Transact Reference Summary*.

VISION:Transact and DB2

Information on the definition and processing of DB2 (or SQL/DS for DOS/VSE) relational data bases can be found in the VISION:Transact Relational Support User's Guide. Execution tracing, described in [Chapter 12, "Execution Tracing"](#), can be used to trace SQL calls issued by the generated application.

VISION:Transact and VSAM/ISAM Files (CICS TS Only)

A VISION:Transact generated application executing under CICS can access fixed and variable length ISAM data bases and keyed, entry sequenced and relative record VSAM data bases.

Note: ISAM files are desupported as of release 1.7 of CICS.

The record format entry on the FD statement specifies which type of data base is being defined.

A VISION:Transact application accesses ISAM and VSAM files through the facilities of CICS. A file is identified to CICS through a File Control Table (FCT) entry, while a file is identified to VISION:Transact through a file definition. The CICS FCT name entry for an ISAM or VSAM file must be specified in the file identification entry on the VISION:Transact FD statement. VISION:Transact and CICS must know the file by the same name so that the appropriate file can be accessed.

Execution tracing, described in [Chapter 12, “Execution Tracing”](#), can be used to trace VSAM/ISAM CICS file control I/O commands issued by the generated application.

Considerations for the File Definition

Suppose that your data base had records in it similar to the one illustrated in [Figure 10-8](#).

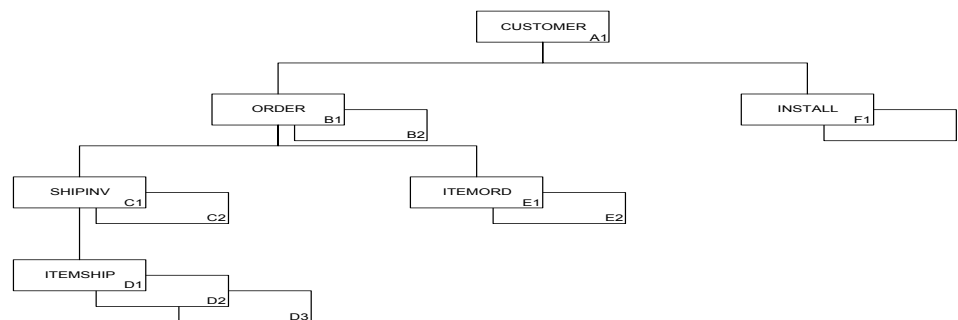


Figure 10-8 A Data Base Record

VISION:Transact would build a structured record in memory like the one depicted in [Figure 10-9](#).

A1	B1	C1	D1	D2	D3	C2	E1	E2	B2	F1

Figure 10-9 The Structure of the Data Base Record in Memory

We can see from the two figures that this record has multiply occurring segments, for example, two orders for customer A1. Non-IMS data bases often consist of segments that occur a specific number of times, referred to as fixed occurring, or that occur a varying number of times, referred to as variably occurring. Both these situations are illustrated in the figures. We can see that the INSTALL segment has an empty occurrence; this is because the INSTALL segment was designated as fixed occurring with the number of occurrences being two. While INSTALL is fixed occurring, we can also see that ITEMSHIP is variably occurring. This is evident by the fact that SHIPINV C1 has three ITEMSHIP occurrences dependent on it, while SHIPINV C2 has no ITEMSHIP occurrences dependent on it.

[Figure 10-9](#) shows the record VISION:Transact builds in memory, which retains an empty segment slot for the missing fixed occurring segment.

A VISION:Transact generated application can access data bases that contain both a fixed and variable number of segment occurrences. In order to access these data bases, VISION:Transact must know how many occurrences exist in the data base. For variably occurring segments, you provide a data field that contains a count of the occurrences in the parent segment. This count field is maintained by VISION:Transact generated application in order to determine the number of segments that are present in any specific record. For fixed occurring segments VISION:Transact uses a specification that denotes how many times that segment occurs. With either of these specifications a VISION:Transact generated application can easily determine how many segments exist in any record.

Certain points must be kept in mind when defining hierarchical structures.

- Every segment must have a key.
- Segments must be numbered top to bottom, left to right, in ascending order.
- In data bases containing fixed occurring segments the file definition must specify the number of times the segment occurs.
- In data bases containing variably occurring segments, a count field must be defined in the parent segment.
- Levels are numbered consecutively from top to bottom in ascending order.
- The last byte in each segment must be defined, whether or not its data is used.
- Fields may be multiply defined or overdefined.

After determining that the data has a particular structure, whether it be hierarchical or not, you can simply map that structure in a VISION:Transact file definition. After this is done and cataloged, you are ready to access them in your VISION:Transact generated application.

VSAM KSDS Files

A VSAM Key Sequenced Data Set (KSDS) is defined to VISION:Transact with a record format of K. The CICS FCT name for the file must be specified in the file identification entry of the FD statement. The definition of the record key to VISION:Transact (level 1, key field 1) must match the VSAM cluster definition and record key content; the starting location, length, and data type of the record key defined to VISION:Transact must match the cluster and data content.

VISION:Transact can process both fixed length and variable length VSAM KSDS files. If the file definition contains no variably occurring lower level segments (no segment count fields), then VISION:Transact assumes the file contains only fixed length records. If variably occurring segments are defined, then the file is assumed to contain variable length records.

Some KSDS files contain variable length records but do not have variably occurring segments. In order to process such files, the largest or maximum size record which may be on the file must be defined in the file definition. In other words, an L0 statement must be provided which defines the last byte of the largest record that may be on the file. In order to provide the record size on insert and update operations, a record length field name must be defined for the file on the RF statement. The record length field is a 4-byte fixed point binary field. After a read operation, the record length field contains the length of the record that was retrieved. Programmers must supply a value for the record length field on insert operations. For update operations, the length need only be modified if the existing record length needs to be changed.

[VSAM Alternate Indexing on page 10-17](#) contains information on processing VSAM files through alternate indexes.

VSAM ESDS Files

A VSAM Entry Sequenced Data Set (ESDS) is defined to VISION:Transact with a record format of E. The CICS FCT name for the file must be specified in the file identification entry of the FD statement. The record key (level 1 key) must be defined as a virtual key field (key field entry of V) in the file definition. The virtual key field must be defined as a 4-byte binary field (field type F) since it will contain the relative byte address (RBA) of the ESDS record.

Records in an ESDS are accessed by their RBA. When an ESDS is processed as a dynamically synchronized file in VISION:Transact, the synchronizing field(s) must contain the RBA(s) of the record or range of records to be retrieved. When an ESDS is processed as a screen synchronized file, segment actions in the screen definition for the root segment must be performed against the file's virtual key field. The screen field whose destination is the virtual key of the ESDS must contain the RBA of the ESDS record to be retrieved (matched), updated, or inserted. If existing records in an ESDS are to be updated, their length cannot be changed. Data records in an ESDS cannot be deleted, but they can be replaced with a record of the same length.

Special considerations exist for inserting records in an ESDS. Records to be inserted in an ESDS are always placed at the end of the file by VSAM. RBAs are not used by VSAM when inserting records in an ESDS. VISION:Transact, however, does use the RBA (virtual key) to uniquely identify a record. Therefore, if multiple records are to be inserted in an ESDS by the VISION:Transact generated application in a single interaction, unique RBA values must be supplied for each record to be inserted. The RBA values supplied for each record to be inserted are arbitrary as VSAM will add each record to the end of the ESDS.

The I segment action should be used for inserting records in an ESDS. If the V segment action (update/insert) is used and record insertion is intended, the RBA value(s) supplied for the record(s) to be inserted must not exist on the ESDS. This is because the VISION:Transact generated application attempts to retrieve records with the V action for update purposes and only when the retrieval attempt is unsuccessful (the RBA value(s) supplied do not exist on the ESDS) will the VISION:Transact generated application insert the record(s) in the ESDS.

VISION:Transact can process both fixed length and variable length VSAM ESDS files. If the file definition contains no variably occurring lower level segments (no segment count fields), then VISION:Transact assumes the file contains only fixed length records. If variably occurring segments are defined, then the file is assumed to contain variable length records.

Some ESDS files contain variable length records but do not have variably occurring segments. In order to process such files, the largest or maximum size record which may be on the file must be defined in the file definition. In other words, an L0 statement must be provided which defines the last byte of the largest record that may be on the file. In order to provide the record size on insert operations, a record

length field name must be defined for the file on the RF statement. The record length field is a 4-byte fixed point binary field. After a read operation, the record length field contains the length of the record that was retrieved. Programmers must supply a value for the record length field on insert operations. For update operations, the length of existing records cannot be changed.

[VSAM Alternate Indexing on page 10-17](#) contains information on processing VSAM files through alternate indexes.

VSAM RRDS Files

A VSAM Relative Record Data Set (RRDS) is defined to VISION:Transact with a record format of R. The CICS FCT name for the file must be specified in the file identification entry of the FD statement. The record key (level 1 key) must be defined as a virtual key field (key field entry of V) in the file definition. The virtual key field must be defined as a 4-byte binary field (field type F) since it will contain the relative record number (RRN) of the RRDS record.

Records in an RRDS are accessed by their relative record number. When an RRDS is processed as a dynamically synchronized file, the synchronizing field(s) must contain the relative record number(s) of the record or range of records to be retrieved. For a screen synchronized file, segment actions for the root segment of an RRDS must be performed against the virtual key field. The screen field whose destination is the virtual key of the RRDS must contain the relative record number of the RRDS record to be retrieved (matched), updated, deleted, or inserted. If existing records in an RRDS are to be updated, their length must not be changed.

Records in an RRDS can only be fixed length.

VSAM Alternate Indexing

VISION:Transact automatically supports the use of VSAM alternate indexing. A VSAM alternate index view must have its own file definition. The alternate index view is defined with a record format of K and the CICS FCT name for the alternate index path (not the base cluster) is specified in the file identification entry of the FD statement. The alternate key is defined as the record key (level 1, key field 1) in the file definition. The attributes of the key (starting location, length, and data type) as defined to VISION:Transact must match the VSAM cluster and data content.

If the alternate key is non-unique, then duplicate keys exist for the alternate view. If duplicate keys exist, the VSAM file must be updated through its primary view (base cluster) instead of through its alternate view (alternate path cluster).

Defining Pseudo Files

While temporary fields and working storage fields provide many different ways to store intermediate results in VISION:Transact generated applications, they may only define single occurring data (root segment). Pseudo files provide a kind of structured working storage you can use to simulate a structured data base record in storage.

To use a pseudo file, you must first catalog a structured file definition using the standard file definition statements (FD, LS, and L0). Segments can be variable or fixed length as with VSAM or ISAM files (see previous section). This definition is used to build the structured working storage record. The record exists only for the duration of one interaction unless you specify that it is saved (Storage Type of S) on the RF statement for the pseudo file. The pseudo file record contains null values at the beginning of an interaction unless your application is using a record saved from a previous interaction. Up to ten pseudo files may be used in an application (FILE - FILE9) and all pseudo file records are available for output processing (type O).

Fixed occurring segments on a pseudo file may be initialized during type 1 request or procedure processing. All fixed occurring segment occurrences are available at this point in the interaction including those without data. After type 1 time, however, those fixed occurring segments with null values (blanks or zeroes) in all key fields are not available. If you initialize fixed occurring segments in a type 1 request or procedure and ascending or descending sequence is specified for a segment in the pseudo file definition (LS statement), then you must ensure that segments are initialized in the correct key sequence. Note that variably occurring segment types in a pseudo file cannot be initialized in a type 1 request or procedure. The insert (I) segment action in the screen definition must be used to create variably occurring pseudo file segment occurrences.

Like SSDBs, pseudo files may be the destination for segment and field actions in the screen definition. Thus, pseudo file segments can be screen synchronized. If no segment actions are specified for a given pseudo file on the input screen, then only one record (including all subordinate segments of the record) will be available from the pseudo file during the interaction. This record will be either the record saved from a previous interaction (if the Storage Type entry for the pseudo file was S) or the record that was initialized during type 1 request/procedure processing. If segment actions are specified for the pseudo file, then only the screen synchronized segment occurrences are available to the application during an interaction. Screen synchronized pseudo file segments are processed identically to screen synchronized data base segments and follow the same rules (see [Chapter 6, "Structured Screens/COMMAREA And The Intermediate Structure"](#)).

Fields in a pseudo file are normally initialized to their appropriate null values (blanks or zeroes) at the beginning of each interaction. This only includes fixed occurring segment types; the count fields for variably occurring segments are set to zero. If you wish to retain values in a pseudo file record between interactions (fixed-length single record only; no variably occurring segments), you can specify

this on an RF statement in your APPGEN (Storage Type entry of S). This specification causes the pseudo file to be initialized to null values only at initial entry to the transaction. In an IMS TM application this specification designates the pseudo file record as SPA- resident. Refer to the *Advantage VISION:Transact for CICS Environment Guide* for a discussion of IMS TM conversational processing. In a CICS application, this specification directs the application to store the pseudo file record in CICS temporary storage between interactions.

Note: For CICS, all working storage areas (regardless of their storage type) are placed in the TWA. Pseudo files are not placed in the TWA but reside in a separate storage area acquired during online execution (DSA of CICS).

Refer to the *Advantage VISION:Transact Reference Summary* for details concerning FD and RF statement requirements.

Defining Working Storage, PCB Masks, and Temporary Fields

Working storage refers to a form of temporary, in-memory storage provided by VISION:Transact. It is defined using a file definition and may be viewed as a single record, single segment file resident in main storage. Including working storage files in an APPGEN is done using an RF statement. Fields in working storage are normally initialized to their appropriate null values at the beginning of each interaction. If you wish to retain values in working storage between interactions, you can specify this on an RF statement in your APPGEN (Storage Type entry of S). This specification causes the working storage area to be initialized to null values only at the initial entry to the transaction. In an IMS TM application this specification designates working storage as SPA-resident. Refer to the *Advantage VISION:Transact for CICS Environment Guide* for discussion of IMS TM conversational processing. In a CICS application this specification directs the application to store the working storage area in CICS temporary storage between interactions. All working storage areas regardless of their Storage Type entry are placed in the TWA. Applications refer to data in working storage using a W qualifier. Any number of working storage areas per application may be designated to be retained between interactions. If more than one working storage definition is used in an application, the field names used must be unique across all definitions. Note that working storage fields, unlike temporary fields, may be defined to overlap each other.

Often an application that processes DL/I files needs access to information in the IMS PCB or requires a PCB mask for a user supplied DL/I call (DLI or CALL statement/command). The PCB mask must be defined if the PCB is to be accessed procedurally in a VISION:Transact request or procedure. The PCB mask in an application is defined using the file definition. The PCB data is maintained only

for the duration of the interaction. Furthermore, PCB data may not be modified by the application program. The qualifier V is used to refer to fields from the PCB file definition. The inclusion of a PCB definition is indicated with an RF statement in the APPGEN. Like working storage, field names must be unique across multiple PCB masks used in an APPGEN.

Temporary fields are working storage areas that are not defined using the file definition. They are explicitly defined by using a temporary field definition (TF) statement in a request or FIELD command in a procedure during an APPGEN run. Temporary fields differ from working storage in that there is no relationship between the location of one temporary field and another. This makes it possible to define temporary fields without specifying location, thus without requiring a file definition.

Advanced Application Flow

The purpose of this chapter is to expand upon the basic concepts covered previously in this guide and to introduce some of the more advanced techniques. Included in this section are topics concerning screen control, extended application generation and the VISION:Transact library.

VISION:Transact Screen/COMMAREA Control

In [Chapter 4, “Application Flow”](#) we covered the basic concepts of screen control and the screen flow of an application. In this section we will cover some of the more advanced concepts including sharing screen requests or procedures between different screens for execution and providing control information for processing and routing the output screen. For more details concerning specifications in screen control, refer to the *Advantage VISION:Transact Reference Summary*.

Output Select Operator

The OS (Output Select) operator may be used in a screen control statement to name the output screen, its destination and the type O request or procedure (if any) associated with the screen. In order to reference a type O request or procedure, the Qualifier for Operand B must be O followed by the name of the type O request or procedure in the Operand B field. Additionally, the SCREEN NAME entry on the ER statement for the type O request or procedure must be blank. A separate OS operator is used for each screen to be output to a destination and follows the logical expression.

If you use the OS operator and a type O request or procedure is to be executed, it must be specified on the same screen control statement. Even if a type O request or procedure is included in stream, it will be ignored if the type O request or procedure name is not included on the same screen control statement. In addition, if a type O request is referenced on a screen control statement and is not found in the input stream, VISION:Transact will attempt to retrieve it from the library.

Note: If the output request or procedure has a screen name in the ER statement, an OS operator on a screen control statement cannot reference that request or procedure.

In [Figure 11-1](#), the OS operator on statement 110, specifies output procedure PROCORDR for SCREENB if F,ATTNID is equal to PF008. If F,ATTNID is not equal to PF008, statement 120 specifies output procedure PROCFMT for SCREENB. When SCREENB is input to the interaction and F,ATTNID is equal to ENTER, output procedure (PROCTIME) is selected for output SCREENC. Otherwise, SCREENC is output to the terminal without executing a type O request or procedure and also output to another terminal executing an output procedure PROCTIME.

INPUT SCREEN	SIMT TYPE	SEQ NO.	LOG LVL	CON CTR	OPERAND A.. QLF FIELD	OPER ATN	 QLF FIELD	OPERAND B..... NAME/CONSTANT	OUTPUT SCREEN	..OPERAND C.. QLF FLD/CNST
(SCREENA)	(SC)	(100)			(F,ATTNID)	(EQ)	(C,PF008	)		
(SCREENA)	(SC)	(110)				(OS)	(O,PROCORDR	)	(SCREENB)	
(SCREENA)	(SC)	(120)		(E)		(OS)	(O,PROCFMT	)	(SCREENB)	
(SCREENB)	(SC)	(140)			(F,ATTNID)	(EQ)	(C,ENTER	)		
(SCREENB)	(SC)					(OS)	(O,PROCTIME	)	(SCREENC)	
(SCREENB)	(SC)			(E)		(OS)			(SCREENC)	
(SCREENB)	(SC)					(OS)	(O,PROCTIME	)	(SCREENC)	(C,TERM)

Figure 11-1 Using the OS Operator in Screen Control

Output Control Operator

The OC (Output Control) operator is used to supply control information for processing and routing the output screen. In order to specify control information, the Operand B Qualifier must be blank followed by the control information in the Operand B field. One or more OC operators apply to the nearest preceding OS operator or last output screen named.

Valid OPERAND B entries include:

XIND	Suppresses the indicators for missing, invalid, numeric overflow, and null fields.
CONV	The IMS TM conversation is continued by the next transaction named in the preceding OS operator as Operand C or the previous SC statement as Operand C. For IMS TM, CONV indicates that the SPA is to be inserted before the message. For CICS, it means the application should converse with the primary terminal whose output screen was previously named on the OS operator. (The output screen is written to the primary terminal and the application awaits a response. All resources are held until the response is processed.)

START	In CICS, flag TRANCODE names a transaction that is to be initiated via the CICS START command. Only one page of the previously named output screen and any saved working storage and pseudo files are passed to the started transaction. (Not applicable to IMS TM.)
RECYCLE	One page of output is produced and is immediately processed as input to this application.
OUTLIM nnn	Specifies the maximum number of pages to be output where nnn=1-999. If Output Control of A is specified on the SD statement for a screen, then the OUTLIM parameter can limit the number of pages to be output. (The flag field PAGENO can be used to display page numbers on output screens.) Note that OUTLIM overrides an Output Control of blank or S on the output screen's SD statement.

[Figure 11-2](#) demonstrates the use of the OC operator in screen control. If statement 100 or 110 are true, the output screen selected will be ITEMSEL. In addition, statement 130 limits the number of pages to be output to the terminal to five. If, on the other hand, statements 100 or 110 have a false condition, the output screen BRWSITEM will be sent to an alternate destination, ALTTRAN. For IMS TM statement 160 specifies that any indicators for null, missing, invalid or numeric overflow conditions will be suppressed. This prevents ALTTRAN from having to include program specifications to ignore these conditions. Statements 170 through 200 apply to a CICS application. When Operand C of the OS operator is left blank, the transaction will be started for the originating terminal. Operand C of the OS operator may specify the terminal ID the transaction is to be started for. The transaction code of the transaction to be started has to be moved into the TRANCODE flag prior to screen control. The additional OC operator (statement 190) is required to start the CICS transaction. VISION:Transact passes the output screen and any saved working storage and/or saved pseudo files (Storage Type S on the RF statement) to the started transaction.

INPUT SCREEN	SIMT TYPE	SEQ NO.	LOG LVL	CON CTR	OPERAND A.. QLF FIELD	OPER ATN	 QLF	OPERAND B..... FIELD NAME/CONSTANT	OUTPUT SCREEN	..OPERAND C.. QLF FLD/CNST	PARTIL FLD STR NM	FLD OP
(INPTWSE)	(SC)	(100)			(T,ITEMND)	(EQ)	(C,G	)		(1, 1,A)		
(INPTWSE)	(SC)	(110)		(A)	(F,ATTNID)	(EQ)	(C,ENTER	)				
(INPTWSE)	(SC)	(120)				(OS)			(ITEMSEL)			
(INPTWSE)	(SC)	(130)				(OC)	(OUTLIM	5	)			
(INPTWSE)	(SC)	(140)		(E)								
(INPTWSE)	(SC)	(150)				(OS)			(BRWSITEM)	(C,ALTTRAN)		IMS TM
(INPTWSE)	(SC)	(160)				(OC)	(XIND	)				
(INPTWSE)	(SC)	(170)		(E)								
(INPTWSE)	(SC)	(180)				(OS)			(BRWSITEM)			CICS TS
(INPTWSE)	(SC)	(190)				(OC)	(START	)				
(INPTWSE)	(SC)	(200)				(OC)	(XIND	)				

Figure 11-2 Use of the OC Operator in Screen Control

Execute Request or Procedure Operator

The XR (Execute Request or Procedure) operator gives you the capability of identifying the type 1, 2, and M requests or procedures to be executed for the input screen through screen control statements. The Operand B Qualifier must contain the type as specified on the ER statement followed by the name of the request or procedure to be executed in the Operand B field. In addition, requests/procedures referenced by an XR operator must have blanks in the Screen Name entry on the ER statement. All SC statements with XR operators must precede all SC statements for the same input screen. In addition, all XR operators for type 1 requests/procedures must precede those for type 2 requests/procedures and the same is true for type M requests/procedures.

VISION:Transact first verifies that requests and procedures named on XR operators are included in the APPGEN input stream. If the requests are not included instream, VISION:Transact attempts to retrieve them from the library.

Including XR statements as part of the input stream also gives you the capability of specifying the order in which the requests or procedures will be executed within a given type. This is necessary if you have, for example, three type 2 requests or procedures for the same input screen. If you reference a request in the input stream and it is not found, VISION:Transact will attempt to retrieve it from the library.

Note: A request or procedure specified by an XR operator must not have a screen name on its ER statement for that request. Screen requests or procedures with a screen name specified on the ER statement cannot be referenced by an XR operator. An APPGEN that includes screen requests or procedures with and without screen names must have those requests or procedures without screen names specified by XR operators. When executing, those requests or procedures referenced by an XR operator are executed first followed by APPGEN requests or procedures naming a specific screen on the ER statement.

The XR statements in [Figure 11-3](#) specify that type 1 procedure ORDERVAL will be executed first followed by the type 1 procedure DATERTN. The type 2 procedure ALIGNDB for input screen ORDRUPDT is then executed. The XR statements precede the rest of the statements for ORDRUPDT. If F, ATTNID is equal to PF005, output screen ORDRDISP will be selected. Otherwise, output screen ORDRDUE is selected along with the output procedure FRMTOUT. Remember that all requests/procedures named in XR and OS screen control statements must have a blank entry for screen name on the ER statement.

Note: If, in [Figure 11-3](#), the APPGEN also includes a type 1 procedure specifying the ORDRUPDT screen, it is executed after type 1 procedures ORDERVAL and DATERTN.

INPUT SCREEN	STMT TYPE	SEQ NO.	LOG LVL	CON CTR	OPERAND QLF	A.. FIELD	OPER ATN	 QLF	OPERAND B..... FIELD NAME/CONSTANT	OUTPUT SCREEN
(ORDRUPDT)	(SC)	(100)					(XR)	(1, ORDERVAL	)	
(ORDRUPDT)	(SC)	(110)					(XR)	(1, DATERTN	)	
(ORDRUPDT)	(SC)	(120)					(XR)	(2, ALIGNDB	)	
(ORDRUPDT)	(SC)	(130)			(F, ATTNID	)	(EQ)	(C, PF005	)	
(ORDRUPDT)	(SC)	(140)					(OS)			(ORDRDISP)
(ORDRUPDT)	(SC)	(150)		(E)						
(ORDRUPDT)	(SC)	(160)					(OS)	(O, FRMTOUT	)	(ORDRDUE)

Figure 11-3 Use of the XR Operator in Screen Control

Screen Request/Procedure Sharing

Screen requests or procedures can specify a particular screen by including the screen name on the ER statement of the request or procedure. However, your application may share a single request or procedure among several screens. In order to do this, the screen name entry on the ER statement of the request or procedure must be blank. Screen control operators are used to specify which requests or procedures to execute for an input screen (XR) or output screen (OS).

Extended Application Generation

An extended application generation run may include all statement types. Below is a brief description of the input sequence and how the statements are processed when included in an application generation run. Refer to the *Advantage VISION:Transact Reference Summary* for an in depth discussion of statement types and runs.

Statement Types and Input Sequence

If additional run control group statements are to be included in an application generation run, the recommended sequence is as follows: (RC, IT, RF, RP, and ?T), file definition statements (FD, LS, LO, and Ln) followed by screen control statements, screen processing definition statements, and request/procedure statements in any sequence. While changes may be made to existing file definitions by adding L0 statements, etc., these updates are "temporary" (for the duration of the run). In addition, new or replacement file definitions are not

cataloged on the library. Updating, replacing or creating file definitions on the VISION:Transact library requires a file definition run. Screen definitions included in an application generation run are cataloged on the VISION:Transact library. In an APPGEN you also have the option of adding, replacing, or deleting requests on the library.

Note: If LIBOPNA=R is specified in M5PARM, screen definitions cannot be input and individual requests cannot be added, replaced, or deleted on the library in an APPGEN run. This specification can be overridden by the Library Open entry [28] on the RC statement.

VISION:Transact Library Facilities

The VISION:Transact library contains data definitions, screen definitions, requests and request groups stored during a definition run and may also contain screen definitions and requests cataloged during an application generation run. Using VISION:Transact library retrieval and maintenance operations (CT statements) you can perform functions such as glossary listing, save or delete request groups, retrieve source statements, etc. Details regarding specifications for library maintenance (CT statements) may be found in the *Advantage VISION:Transact Reference Summary*. Information regarding initializing, condensing, dumping, restoring, merging and copying items from one library to another is discussed in the *Advantage VISION:Transact for CICS Environment Guide*.

Execution Tracing

Execution tracing is a debugging tool for VISION:Transact application programmers. During application generation (APPGEN), the Trace Control entry on the ?T statement can be used to include execution tracing in an application for debugging purposes or exclude it when placing an application into production.

When execution tracing is included in an application (Trace Control entry of 0 or 1 on the ?T statement), the following tracing options are available:

- **Application Flow**
Displays trace messages (prefixed by AF) at application execution milestones and strategic points in the interaction cycle and trace messages (prefixed by TP) for teleprocessing interface I/O calls issued for screens and saved working storage/pseudo files. The I/O areas for the TP calls may optionally be output.
- **Procedure**
Displays trace messages (prefixed by PT) to indicate key events that occur during the execution of requests or procedures, and displays output from trace operators/commands coded in requests/procedures.
- **Segment Action**
During field validation and data base retrieval which occurs in the interaction after type 1 processing, outputs a trace message (prefixed by SA) to show the segment action selected for a segment occurrence just prior to the segment being read from the data base.
- **Intermediate Structure Looping**
Displays trace messages (LP01 through LP10) that reveal the status of loops on the intermediate structure and the content of the intermediate structure. The intermediate structure consists of input screen segments, data base segments that are synchronized to the input screen segment (segment actions specified), and the child segments of synchronized data base segments. These child segments are referred to as unsynchronized segments since they do not have a segment action specified or the X segment action was selected for all occurrences of the segment.
- **Dynamically Synchronized Data Base (DSDB) Looping**
Displays trace messages (LP31 through LP40) that reveal the status of retrieval and looping on DSDBs.

-
- **Screen Synchronized Data Base (SSDB) I/O Calls**
Displays trace messages (prefixed by DB) that provide information about the data base calls (DL/I, SQL, or VSAM) performed by the application for those data bases defined as SSDBs. I/O areas for each call may optionally be included in the trace output.
 - **Dynamically Synchronized Data Base (DSDB) I/O Calls**
Displays trace messages (prefixed by DB) that provide information about the data base calls (DL/I, SQL, or VSAM) performed by the application for those data bases defined as DSDBs. I/O areas for each call may optionally be included in the trace output.

Trace messages are listed along with their explanations in the *Advantage VISION:Transact Messages Guide*.

The Trace Control entry on the ?T statement and its corresponding flag field TCONTROL, can be used to enable or disable all tracing. The individual tracing options can be enabled or disabled during APPGEN by an appropriate entry on the ?T statement or during execution through the use of a corresponding trace flag. The ?T statement and trace flags are fully documented in the *Advantage VISION:Transact Reference Summary*.

Trace flags can be modified by requests or procedures as well as screen field actions in order to enable or disable all or individual tracing options. Unlike other flags, trace flags retain their value between interactions; they are not reinitialized until application termination which can span multiple interactions. For IMS TM, application termination occurs after all input messages have been processed. For CICS TS, application termination occurs when the pseudo conversation has terminated. This can span multiple transactions.

Note that if a ?T statement is not included during APPGEN or the ?T statement contains an entry of P in Trace Control, then tracing is not included in an application and any modifications to the trace flags are ignored. The trace flags cannot be used to enable or disable tracing unless a ?T statement has been included during APPGEN with a Trace Control entry of 0 or 1.

Modifications to the trace flags do not affect the tracing options being used until the request or procedure terminates. The new tracing options will become effective upon exit from the request or procedure and remain effective until they are changed or the application terminates. In the case of field actions modifying the flags, the trace options do not become effective until in-storage update processing has been completed, thus, becoming effective for the first type M request or procedure. The reason the new trace options do not become effective until execution of a request or procedure completes is to avoid incomplete trace messages being output. For example, if the trace flags were modified after a data base loop had been established within the request or procedure, incomplete and misleading looping and I/O call trace messages could be output.

If you want to change the tracing options in effect during the execution of a given request or procedure, then a subroutine request or procedure can be used to modify the appropriate flags. The new tracing options would become effective upon exit from the subroutine and return to the calling request or procedure.

When application execution begins, the trace options in effect are those specified on the ?T statement. Type 1 processing is the earliest point within the interaction cycle when the trace options in effect can be modified. Therefore, the only way to modify the trace options in effect prior to type 1 processing (e.g., trace messages AF01-AF14), is to modify the trace flags in a previous interaction.

When procedure tracing is enabled, literals or the contents of fields can be displayed in the trace output during the execution of requests or procedures. Displaying literals during the execution of requests or procedures is useful for identifying lines or blocks of code. Displaying field contents is extremely useful for debugging application logic.

In requests, two trace operators are provided: ?C to display a literal/field in character format, and ?X to display a literal/field in both character and hexadecimal format. In procedures, the TRACE command is used to display literals/fields in character or hexadecimal. Note that the trace operators/commands are only executed when procedure tracing is enabled; literals/fields are not output in the trace data when procedure tracing is disabled.

Since trace operators and commands must not have any substantial effect upon the operations performed during application execution, looping is not initiated for fields referenced by trace operators or commands. Thus, a loop must be started on a segment before the contents of any field within the segment can be displayed by a trace operator or command. If no loop has been started for the segment when the trace operator/command is executed, then the literal "NO LOOP STARTED FOR SEGMENT" is displayed instead of the field contents in the generated trace output.

If the field named in a trace operator or command is flagged as invalid or the segment containing the field is missing, then the literal "INVALID FIELD" or "SEGMENT IS MISSING" is displayed instead of the field contents in the generated trace output.

For IMS TM applications, execution tracing is only supported for applications executing under BTS. Under BTS, trace data is output to the BTSOUT data set and also to the terminal if BTS TSO is used. For more information on execution tracing of IMS TM applications, see Chapter 6 of the *Advantage VISION:Transact for CICS Environment Guide*.

For CICS TS applications, trace output can be written to the terminal, temporary storage, and/or transient data. For more information on execution tracing of CICS TS application, see the Chapter 6 of the *Advantage VISION:Transact for CICS Environment Guide*.

Introduction and Overview (DB2 UDB)

This document explains the concepts and facilities provided by VISION:Transact in support of relational data bases.

The document is written for people who have been trained in VISION:Transact and understand basic VISION:Transact concepts and facilities. If you do not know VISION:Transact, stop here.

In this document, we show how the VISION:Transact concepts and specifications are extended to support relational tables. You will be surprised to learn how simple relational support is and will welcome the fact that, for a VISION:Transact applications designer there is virtually nothing new to learn. There are no new specifications for Screen Control or Screen Definition. There are no new specifications for request or procedure processing. The only new specifications are in file definition (and one on the RF statement). Even these specifications reduce to:

- 1 on the FD statement
- 3 on the LS statement
- 1 on the L0 statement
- and a new statement, the LR statement.

The VISION:Transact relational data base users will be happy to learn that these specifications are identical to the specifications used to access relational data bases with VISION:Transact. Indeed, the file definitions that are used with VISION:Transact can be used with VISION:Transact.

There are only a few new specifications to learn.

What is new, is the concept of the logical record. This provides you with the ability to solve problems by defining the data to be the way that you wish it were stored, totally independent from its actual physical organization.

[Chapter 14, “A Conceptual Overview \(DB2 UDB\)”](#) introduces the concept of the logical record. We show how physical tables are mapped into logical records. Once you have a logical record, everything acts the same as if the record had been obtained from IMS or VSAM. In short, once you have defined your logical record, problem solving with screen synchronization, dynamic synchronization, and procedures and requests are identical regardless of the source of the data.

[Chapter 15, “Specifications \(DB2 UDB\)”](#) details specifications associated with relational data base support. Chapter 15 is not a complete specifications manual. We assume you know and understand the VISION:Transact specifications. Chapter 15 highlights only the new specifications.

[Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#) and [Chapter 17, “Updating and Designing for Referential Integrity \(DB2 UDB\)”](#) include a considerable number of examples to aid in understanding the logical record and to demonstrate the remarkable capabilities that it provides you. Unfortunately, the syntax of VISION:Transact does not lend itself well to hard copy. To demonstrate our points, we have used a sort of “pidgin” VISION:Transact syntax. Thus, once again we emphasize that you must know VISION:Transact in order to be able to read it.

[Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#) illustrates the logical record. It becomes evident in this chapter that there is virtually no requirement to understand relational data base technology. The mere understanding that tables exist is all that is required. In [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#), we focus upon retrieval and display from the relational data base. Since this document is about relational support, the focus is exclusively upon specifications and techniques that support relational data bases. However, VISION:Transact supports any desired combination of relational data bases, IMS, or VSAM at the same time. Each of the examples in [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#) illustrates some facet of the logical record. Some of the examples demonstrate the same capabilities as other examples, but in a different context. This is to assist you should the context be in the way of understanding the capabilities.

[Chapter 17, “Updating and Designing for Referential Integrity \(DB2 UDB\)”](#) introduces the various aspects of updating the data base. Updating is, in general, exactly what you would expect it to be - putting new data back in place of the old data. However, relational operations are set operations and that means, in a carelessly designed application, an update can be made to affect more than one row of the table. In [Chapter 17, “Updating and Designing for Referential Integrity \(DB2 UDB\)”](#), we discuss this aspect of relational data bases and show how easy it is to protect against it using VISION:Transact. Updating relational data bases introduces the well known problem of referential integrity. We define and explain referential integrity and show how the logical record capability can provide automatic referential integrity. In addition, we show how an application can deal with sophisticated referential integrity considerations very easily using VISION:Transact.

[Chapter 18, “Helpful Hints \(DB2 UDB\)”](#) is a potpourri of information. A VISION:Transact application is no different from any other application that uses embedded SQL statements. Compiles, Binds, Authorizations, etc. are exactly the same with VISION:Transact as with PL/I or COBOL. We generally expect that an installation will understand the mechanics of running online jobs against DB2 (or SQL/DS) data bases. In this guide, we do not provide a comprehensive set of information on running a VISION:Transact application; that information is in our *Advantage VISION:Transact for CICS Environment Guide* and in the IBM manuals. Rather, [Chapter 18, “Helpful Hints \(DB2 UDB\)”](#) includes a set of miscellaneous notes and hints to aid with your understanding of the VISION:Transact processing.

A Conceptual Overview (DB2 UDB)

Introduction

In this chapter, we introduce the basic concepts of the VISION:Transact support of relational data bases.

A whole host of questions come to mind:

Do we have to learn SQL? Are there new verbs and/or codes? What are the differences from VSAM and IMS? Can we use relational data bases and VSAM and IMS? How much training is involved?

Let us deal with these issues now.

Basically, everything works just the way it does with VSAM and IMS (DL/I). If you know how to write an application using VSAM or IMS, you are fully capable of writing applications using relational data bases. Screen synchronization, dynamic synchronization, and all of the other powerful features of VISION:Transact are unchanged. There is no need to learn SQL and there are no new codes for application generation. All of this is achieved by the powerful new concept of the logical record, which is explained below. There are, of course, some new specifications but they all occur as part of the file definition or run control statements. Make your file definitions and from that point on nothing has changed.

Training? You are doing it. By the time that you have read this document and tried your first application, you will be completely trained.

The power of the logical record is at the heart of the simplicity of the relational data base support.

In this chapter, we are going to deal with concepts. The details of the specifications are in [Chapter 15, "Specifications \(DB2 UDB\)"](#) and examples of usage will be found in [Chapter 16, "Details of Operation and Examples \(DB2 UDB\)"](#) and [Chapter 17, "Updating and Designing for Referential Integrity \(DB2 UDB\)"](#).

We start this chapter by observing that if you prefer to treat each of the relational tables separately, then you can do so with no more difficulty than processing a series of flat VSAM files.

We then go on to talk about the Logical Record. We talk about defining it and illustrate how it is used. Master the concept of the logical record, and you know all that you need to know to create applications involving an unlimited number of relational tables. The chapter ends with a discussion of this powerful feature. We find it wise to step back, ask “Why?”, and look at the advantages that might be overlooked with the simplicity of the specifications.

Relational Data Bases as Access Methods

The simplest way to look at relational data bases with VISION:Transact is as a set of flat files. With this view of data, each table is regarded as a file and the rows in the table as records of the file. This approach simply uses the relational data base manager as another access method in place of VSAM or IMS/DB (or DL/I).

The processing of the records with this approach is identical to the processing that would occur with single level VSAM or IMS data bases. In constructing your VISION:Transact application, there are no new specifications and nothing new to learn. During execution, a record is obtained from a relational table instead of a single level VSAM (IMS/DB or DL/I) file. But this is immaterial to you as an application developer. There are some new specifications, but they all occur as part of the file definition.

Logical Records Versus Physical Records

A powerful feature of VISION:Transact is the ability to define logical records composed of data from several tables. This, in effect, joins data from tables in such a way that you can view your data within the structure of the problem that you are trying to solve.

The logical record is both a new and an old concept for the VISION:Transact user. It is new in that data from several tables is being merged together to construct a record that does not, in fact, physically exist anywhere in a data base. It is an old concept in the fact that VISION:Transact applications have always used logical records. Every VISION:Transact application ever written has used logical records.

What is really new is the terminology. The advent of relational data bases leads us to take a more disciplined approach to our terminology. VISION:Transact operates by copying physical data from disk into memory, making all changes and extractions of data, and copying the changed data back to the disk. The record in memory, against which all the activity takes place, is really a logical record. In the

past there was little point in making this distinction since there was little or no difference between the physical and logical records. In fact, for ease of explanation, much of our documentation is described in terms of operations against the physical data base. However, with relational data bases, there is a considerable difference between the physical data as it resides on disk and the logical, in-memory, data. For this reason, we must emphasize the distinction between the physical (what exists on disk) and the logical (what is seen by the application).

Since VISION:Transact operates against logical data, the VISION:Transact specifications and processing flow are independent of the particular data base used to construct the logical record. Therefore, VISION:Transact is used to develop applications using relational data bases in exactly the same way that it is used with VSAM or IMS (DL/I). Hierarchical records are materialized to become the objects of screen synchronization or dynamic synchronization.

Support of a data base access method (relational, in particular) revolves around mapping the physical data from disk into the required logical representation to be used by the application. Once you have the logical record, then all the rules that you have learned about looping, extending a screen synchronized record by dynamic synchronization, and request and procedure processing, all apply.

Terminology

We digress a little to clarify and emphasize the terminology. This chapter is included for those who like to have a complete understanding of the full framework.

VISION:Transact follows this sequence of events:

1. Logical data is assembled from a series of physical files.
2. The application acts upon the logical data.
3. Logical data is returned to physical files.

Logical Data	Logical data within a VISION:Transact application is composed of data from a series of logical files. One <i>logical record</i> is obtained from each logical file, according to some sequence, to form a <i>logical data view</i> .

Logical File	A logical file is a file in a VISION:Transact application. It has a separate file definition and is identified on a RF statement. In VISION:Transact, the logical files are the set of screen and dynamically synchronized data bases identified as FILE and FILE1 through FILE9.

Logical Record	A logical record is a hierarchical record that has been defined in a VISION:Transact file definition. Historically, there has been a one-to-one correspondence between a logical record and a physical record, so there was little need to make a distinction. But with the advent of relational data bases, the logical record takes on a significant meaning as it may be composed of data from many physical sources (tables).

The VISION:Transact file definition has been extended so that it not only describes the logical record structure but provides the information that enables VISION:Transact to initialize the logical data from the fields and tables that comprise the physical data.

Logical Data View	A logical data view is a set of logical records combined together in an application. In VISION:Transact, the logical data view is achieved by the process of screen synchronization extended by dynamic synchronization. A logical data view is the combination of the records selected. It is an assemblage of all of the data that you can see at any point in time and is defined by the set of RF statements.

Each logical record is composed of data from a *specific data base* or access method. The logical data view is the mechanism for combining data from *different* data bases or access methods.

Defining the Logical Record

From the discussion above on the difference between logical and physical records, it is evident that the key to the support of relational data bases is that of defining logical records and mapping the data from the tables into the records. All of this is done through the file definition.

The basic concepts are explained here. The detailed specifications are described in [Chapter 15, "Specifications \(DB2 UDB\)"](#) and examples are illustrated in [Chapter 16, "Details of Operation and Examples \(DB2 UDB\)"](#).

Logical records are defined to VISION:Transact in exactly the same way as for VSAM and IMS (DL/I). There are some new extensions to the file definition to establish the physical mapping, i.e., the physical data that is to be moved into the logical record structure.

Within a *relational* VISION:Transact file definition, every segment represents a table. This is illustrated in [Figure 14-1](#).

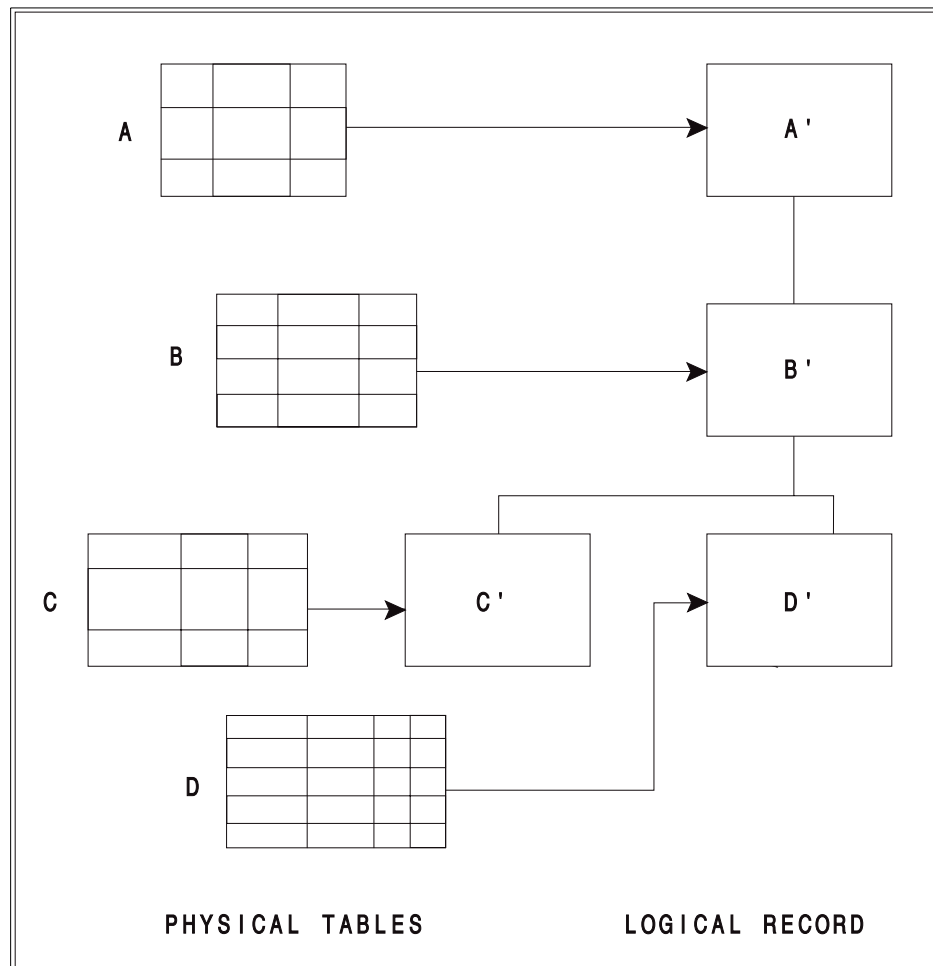


Figure 14-1 Mapping Tables to Segments of a Logical Record

[Figure 14-1](#) shows the logical segment A representing the table A, B representing B, and so on. The only specification required to complete this mapping is to identify to VISION:Transact which table the segment is to represent. This is done by a single entry on the LS statement to provide the name of the relational table.

A segment is a logical entity so it does not have to contain all of the fields (columns) of the relational table that it represents. This differs from VSAM and IMS where the logical and the physical must be the same. This is illustrated in [Figure 14-2](#).

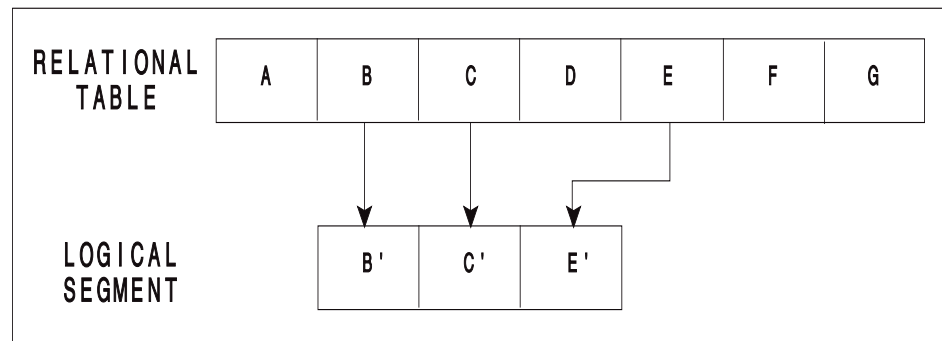


Figure 14-2 Subsetting the Table Columns

[Figure 14-2](#) shows the mapping (the rules for movement) of physical fields B, C, and E to logical fields B, C, and E. This mapping is achieved by a single additional entry on the L0 statement which identifies the name of the field (column) that is the source of the data. When defining the logical field you merely identify which physical field is to supply the data to fill it.

With relational data bases, we can completely separate the physical from the logical, we can introduce a powerful new capability, that of unmapped fields in the logical segment. These are simply fields in the segment that are not filled with data from a relational table. They can be used as temporary working fields associated with the segment.

Unmapped fields are illustrated in [Figure 14-3](#).

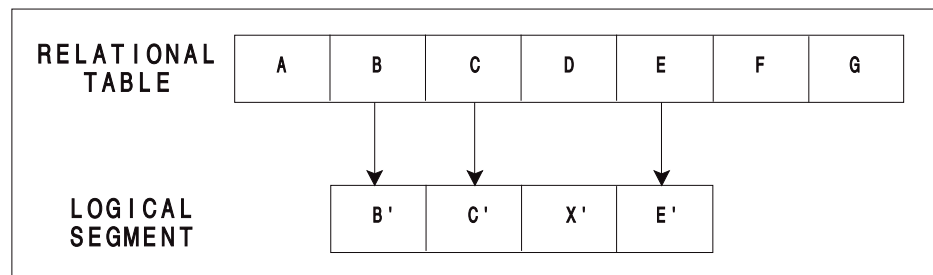


Figure 14-3 Defining Unmapped Fields

In [Figure 14-3](#), the logical field X exists for the use of the application but does not exist in the data base.

A logical segment, therefore, consists of a set of mapped fields, which are filled with data from columns of a single relational table and a set of unmapped fields which are initialized to blank or zero as appropriate. Logical fields may overdefine other logical fields. This generally only makes sense when an unmapped field overdefines some mapped fields. Thus we might have “date” (unmapped, 6 characters) overdefining the mapped fields of “month,” “day,” and “year” (each mapped as 2 characters).

There remains one more thing to do and that is to establish the correct parentage between the segments. In IMS and VSAM, this is established by the position of the data in the record. With relational tables, the separate tables have no positional relationship. So we must provide an explicit rule that establishes which child segment belongs to which parent segment. This is done by means of a new statement (or series of statements), the Logical Relationship (LR) statement. This looks very much like the conditional logic that is used in a request, only it is included in the file definition.

For example, we might have two tables, the DEPARTMENT_TABLE and the EMPLOYEE_TABLE shown below.

Note: We assume that you are familiar with the VISION:Transact syntax. Here and in other examples, we depict physical data in upper-case and logical data (VISION:Transact data) in lower-case.

DEPARTMENT_TABLE

DIVISION	DEPTNO	NAME	MANAGER_NO
WIDGETS	A01	FINANCE	123
WIDGETS	B01	SALES	456
WIDGETS	C02	ENGINEERING	789

EMPLOYEE_TABLE

EMPNO	NAME	AGE	SALARY	DEPT	LOC
231	JONES	23	25000	B01	L65
234	SMITH	34	45000	B01	L80
123	WILSON	45	75000	A01	L65
477	BROWN	29	35000	B01	L80
456	WHITE	34	45000	B01	L75
333	GREEN	44	25000	C02	L80
789	BLACK	44	67000	C02	L80
567	BLUE	23	25000	A01	L65
444	ADAMS	23	25000	B01	L65

Viewed as two separate tables, we have a collection of departments and a collection of employees. To view the employees within their respective departments, define a logical record with the root segment (parent segment) as the department and the subordinate segment (child segment) as the employees within the department. We do this with the following logical structure:

File name: dept

Segments	Fields	TABLES
deptmnt	dptno, deptname	DEPARTMENT_TABLE
emps	empnum, name, age, sal, locn	EMPLOYEE_TABLE

The definition is:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>	<u>EXTERNAL</u>	<u>NAME</u>
dept	FD		S				
	LS	deptmnt	10			DEPARTMENT	TABLE
	L0	dptno	10	1	1	DEPTNO	
	L0	deptname	10	1		NAME	
	LS	emps		20			EMPLOYEE
	L0	empnum	20	2	1	EMPNO	TABLE
	L0	name		20	2	NAME	
	L0	age		20	2	AGE	
	L0	sal		20	2	SALARY	
	L0	locn		20	2	LOC	
	LR	emps				DEPT EQ	dptno

The process of constructing a logical record is done a segment at a time. First, a root segment is obtained. Subordinate segments are added if they satisfy the rule(s) or condition(s) provided by the LR statements. In the case that we have here, we demand that rows of a table be transformed into segments only if the field (column) DEPT of the EMPLOYEE_TABLE matches the field “dptno” of the root segment.

With that definition, the VISION:Transact application is written, not against separate relational tables, but against the logical record. With the sample data of the illustration, the VISION:Transact application will process the following records:

A01	FINANCE	123	WILSON	45	75000	L65
		567	BLUE	23	25000	L65
B01	SALES	231	JONES	23	25000	L65
		234	SMITH	34	45000	L80
		477	BROWN	29	35000	L80
		456	WHITE	34	45000	L75
		444	ADAMS	23	25000	L65
C02	ENGINEERING	333	GREEN	44	25000	L80
		789	BLACK	44	67000	L80

While these records do not physically exist, VISION:Transact will make them exist for the purposes of the application.

Screen Synchronization and Dynamic Synchronization

Up to now, the discussion has assumed that the entire logical record is actually constructed. This is really a conceptualization. In a VISION:Transact application, the data that is available to the application is dictated by the screen synchronization and the dynamic synchronization. A VISION:Transact application sees only the data that it needs. The application is written as if the logical records all existed in a logical file. VISION:Transact does not, in fact, build all of those records. It is optimized to retrieve only the data required which, by virtue of screen synchronization, is often a small subset of the total data.

Screen synchronization and dynamic synchronization are performed on the logical records in exactly the same way that they would be performed if the records actually existed in a VSAM or IMS (DL/I) data base.

Updating and Output

Updating and output is very simple. There is really only one rule and no new specifications. *The rule is that rows are replaced from whence they came.* VISION:Transact puts the data back into the row of the relational table from which it originally retrieved the data. This rule is obeyed regardless of the changes to the data made by the application. You cannot make VISION:Transact get data from one row and, by changing VISION:Transact's key fields, put it back into another row.

If logical segments are created, new rows will be added to the tables represented by the appropriate segments. When we discuss update in the context of relational tables, we must be concerned with the problems posed by referential integrity. Referential integrity deals with the issues that arise when data is changed in one table but other tables exist that are dependent on that data. In our Department-Employee example, the integrity of the data base would be seriously compromised if the Sales department code were changed to E01 in the Department table. If this were to happen, it would not be possible to determine the employees of the Sales department since there are no employees in the Employee table with DEPT equal to E01. Briefly, *referential integrity demands that when data is changed in one table, it must be changed in all tables that reference it.* [Chapter 17, "Updating and Designing for Referential Integrity \(DB2 UDB\)"](#) illustrates how to design VISION:Transact logical records to preserve the integrity of the data base.

The Advantages of Logical Records

The logical record facility is an extremely powerful capability unique to VISION:Transact (and VISION:Builder). This capability enables you to develop sophisticated online transaction processing applications rapidly and accurately.

The logical record capability provides you with the following capabilities and advantages:

1. Improved Problem Solving

In using relational data bases, you are not required to think of your data as a set of disjointed tables, but may regard the data as a set of data integrated and structured in a manner that is suitable for solving the problem at hand. In short, you need not worry about how the data is stored. You are at liberty to define the data to be the way that you wish it were stored. The same set of relational tables can be viewed as different logical structures. Problem solving is greatly facilitated if the data is organized in a convenient manner. The logical record capability of VISION:Transact enables you always to have the data organized in the most convenient way. An illustration of this is given in [Chapter 16, "Details of Operation and Examples \(DB2 UDB\)"](#) by the Organization Chart of Example 9.

2. A Conceptual View

The logical record enables you to think of the problem within an appropriate framework which makes it understandable. Relational tables are a nice way to organize sets of related data, but they do not lend themselves to increasing the understanding of what is going on. This is illustrated by the Department-Employee example. We had two tables, the Department table and the Employee table. We were interested in the employees organized by department. We chose to think of them in just that way; a department root with all of the employees subordinate to it. This is far more understandable than trying to wrestle with separate tables joined on matching values. The Organization Chart in Example 9, [Chapter 16, "Details of Operation and Examples \(DB2 UDB\)"](#), provides another interesting example of the ability to understand the problem by formulating it appropriately.

3. Focus on the Business Problem

In solving a problem and developing an online transaction processing application, you (as the application developer) are faced with the dual problems of implementing the business requirements with all of the sequences and algorithms involved as well as wrestling with the data organizations to ensure that the correct data is available when required. The logical record capability enables you to define your data requirements completely independent of the business application proper. You can now forget about the data and get on with solving the real problem. VISION:Transact will guarantee that the right data will be available when wanted.

4. Automatic Navigation

In defining a logical record, you are defining all of the relationships between the data. Thus, in developing the application proper, it is only necessary to reference fields by name and VISION:Transact will automatically navigate through the tables to ensure that all of the correct combinations of data are presented to your application for processing.

5. Views and Security

The Logical Relationship (LR) statement, that we spoke of while describing the building of the logical record, actually has two functions. The first is, that of

relating the data. The LR statement can also be used to filter the data so that the application sees only a relevant subset. In the Department-Employee example, had we included a LR statement on the root segment of,

```
LR          DEPTNO EQ C,B01
```

then the application would have seen only the logical record for the Sales department. It would be as if the Finance and Engineering departments did not exist. This capability is valuable in limiting the scope of the data to that which is relevant to the situation. Since the LR statements that control this filtering exist in the file definition, they provide an effective way to impose security upon the data, i.e., limiting application users only to that data for which they have a “need to know.” Other examples of views are included in [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#).

6. Referential Integrity

The logical record greatly facilitates the maintenance of referential integrity. This subject is discussed in greater detail in [Chapter 17, “Updating and Designing for Referential Integrity \(DB2 UDB\)”](#). However, a brief example will help here. Suppose that we wish to close down the Sales department in our Department-Employee example. The action of deleting the Sales root segment will automatically delete all subordinate segments, i.e., rows from the Employee table. The integrity or consistency of the data base will be maintained. “What,” you may ask, “if I wanted to delete the Sales department but did not want to delete the employees?” All of that is covered in [Chapter 17, “Updating and Designing for Referential Integrity \(DB2 UDB\)”](#).

7. Processing Efficiency

The logical record capability, with the ability to subset the fields retrieved from the tables, with the ability to filter the data by means of the LR statement, and with the screen synchronization capability, enables VISION:Transact to perform the minimum number of data base accesses to retrieve the data for the application. Expensive DB2 or SQL/DS joins are avoided. VISION:Transact only retrieves those fields from the relational table that are referenced in the application and updates/inserts only those fields used as result fields in the application.

8. Data Accuracy

The logical record facility is actually the equivalent of an outer join in relational terms. An outer join occurs when data exists in one table but there is no corresponding data in the related table. A normal join will simply ignore this situation and the application will never see it. This could provide a very misleading set of results and lead to incorrect corporate decisions being made. An outer join is normally quite difficult for a COBOL, PL/I, or other program to achieve. It is not native to SQL. The outer join is the normal mode of operation for VISION:Transact thus ensuring that all relevant data is made available to the application. We illustrate from our Department-Employee example. Suppose that we have a department (Legal) which has no staff (either just formed or the result of attrition). The department table is:

DEPARTMENT_TABLE

DIVISION	DEPTNO	NAME	MANAGERNO
WIDGETS	A01	FINANCE	123
WIDGETS	B01	SALES	456
WIDGETS	C02	ENGINEERING	789
WIDGETS	D05	LEGAL	

The VISION:Transact application will see a record:

D05 LEGAL

There would be no subordinate segments since there are no employees and this would be reflected on the screen in the usual way. Obviously, a VISION:Transact application is able to update this logical record (and subsequently the physical tables) as employees are hired for this department. No extra work is required in developing your VISION:Transact application. It is all taken care of automatically. Any application developed in a conventional 3rd generation language utilizing a join (the normal way of solving this kind of problem) would be unable to handle this boundary situation.

9. Preservation of Concepts and Applications

The logical record concept preserves all of your VISION:Transact training and expertise. Looping, screen synchronization, extending the interaction, request and procedure processing, etc., all work in exactly the same manner as they do for IMS and VSAM. This means that applications developed for IMS will work with relational tables without any source changes to the screen definitions or application logic. File definitions must be prepared to map the tables into logical records, some minor request or procedure statements added to preserve the referential integrity of the relational tables, and a new APPGEN is required, but that is all.

Summary

The logical record is both a simple and powerful capability. It provides you with complete independence from the physical structure of the data base. At the same time, it preserves all of your VISION:Transact training and expertise. It is compatible with and works in conjunction with VSAM and IMS (DL/I).

Most importantly, this capability introduces a new era of problem solving. You no longer solve a problem by trying to make do with the data that you have. You solve problems by dictating that the data shall be the way that you would like to have it.

VISION:Transact is unique in its ability to give you this capability.

Specifications (DB2 UDB)

Introduction

This chapter assumes that you are familiar with VISION:Transact and are conversant with the current specifications. In this chapter, only the specifications that are unique or germane to the use of relational data bases in VISION:Transact applications are described. Restrictions and additions apply only to relational based files and do not apply to other file types unless specifically identified.

Data Base Definition

The specifications for data base definition have been chosen carefully to assist migrating and converting existing applications to support relational tables. Every effort has been taken to ensure that applications which executed successfully with non-relational data bases will work with relational data bases with minimal changes to existing definitions.

In support of conversion and migration facilities, all relational specifications may be included in definitions of any record format. They must be specified correctly. Relational specifications included in file definitions will have no impact on VISION:Transact applications if the data base is not relational. They are permitted purely as a convenience to assist you in converting to relational data bases or in prototyping relational applications with non-relational data bases.

The required entries are listed in this chapter. Examples and illustrations of usage are given in [Chapter 16, "Details of Operation and Examples \(DB2 UDB\)"](#). If there is no comment provided with the entry it can be assumed that it has its usual meaning within VISION:Transact and the usual entries are valid.

The specifications that are documented in this chapter are relevant to relational based definitions.

File Definition (FD) Statement

The following table lists which entries should be supplied for new definitions; all other entries are ignored.

Specification	Columns	Notes
File Name	1-8	Any suitable name for VISION:Transact. (This is not used for relational processing.)
Statement Type	9-10	FD
Record Format	21	Enter S to denote a relational based file.
Buffer Size	29-33	Number of bytes of buffer space that VISION:Transact should reserve for the logical record or table.
Delete	19	This entry has its usual meaning.
Glossary	20	This entry has its usual meaning.

Segment Definition (LS) Statement

For relational tables, the LS statement has the function of identifying the relational table or view that the segment is to represent. When a logical hierarchical record is defined to VISION:Transact, each segment represents a relational table. The hierarchical record provides a method of relating tables to each other, in effect pre-defining joins.

For relational definitions, it is required that every segment have a name provided by means of an LS statement. The name of the relational table or view that the segment represents is obtained from the LS statement for that segment. The fully qualified name of Authorization-ID.tablename may be provided on the LS statement.

The following table lists general specifications which should be provided for the LS statement.

Specification	Columns	Notes
File Name	1-8	
Statement Type	9-10	LS
Segment Name	11-18	Enter the name as known to VISION:Transact. An entry is required for relational tables.

Specification	Columns	Notes
Delete	19	This entry has its usual meaning.
Segment Number	20-21	Enter the segment number.

The following table lists the logical record specifications which should be provided on the LS statement.

Specification	Columns	Notes
Suppress Duplication?	30	Leave blank or enter N if all qualifying rows are to be included as repeated occurrences of the segment. Enter Y to eliminate duplicate rows as segment occurrences automatically. Note: The entry of Y may cause DB2 or SQL/DS to consume significant additional CPU cycles. This specification applies to Record Format S only.
Segment Order	31	Leave blank if there is no order required for the segments. Enter A to retrieve the segments in ascending order. Enter D to retrieve the segments in descending order. Note: The entries of A and D may cause DB2 or SQL/DS to consume significant additional CPU cycles. Descending root segments are permitted for relational data bases.

The following table lists the physical data specifications which should be provided on the LS statement.

Specification	Columns	Notes
Creation or Authorization ID	36-43	Leave blank or enter the authorization ID of the creator of the table or view. If an unqualified table or view name or a synonym name is to be used, then leave this entry blank. If an entry is provided, this name will be concatenated with the external table name entry in the form of : Authorization-ID.tablename for the purpose of accessing tables or views and building the logical record. If all tables or views in the logical record have the same creator, then this entry may be left blank and the authorization ID can be provided on the RF statement.
External Table Name	44-61	Enter the relational object name. This may be the name of a relational table, view, or synonym.

For relational data bases, all LS statement entries not shown in the three previous tables are ignored.

Field Definition (L0) Statement

The process of field definition for relational data bases is almost identical to that of definition for conventional data bases. There is only one new specification required to define the mapping of data from a physical table to a logical record.

The following table lists the entries that are applicable to relational data bases.

Specification	Columns	Notes
Segment Key	31	Leave blank or enter 1 to 9. Any field that is a key field must also be a mapped field (i.e., an External Name must be provided in positions [44-61]).

Specification	Columns	Notes
This Segment Occurs N Times	36-38	This entry is not allowed for relational definitions.
External Field Name	44-61 or 44-63	Enter the name in 44-61 or delimited name in 44-63 of the field as it has been defined to DB2 or SQL/DS. (A delimited name is a name enclosed in double quotation marks.) This field supplies the data to initialize a logical record and is the recipient of any updates to the logical record. If left blank, no mapping or logical to physical (or physical to logical) data transfers will take place. The field is an unmapped field (a logical field defined to VISION:Transact) and can be used as a transient field in a VISION:Transact application. In addition, the use of unmapped fields on LR statements allows programmers to relate tables based on calculated or derived argument values effectively creating a dynamic rule.

All other entries on the field definition statement have their usual meaning.

It should be noted that VISION:Transact allows a different scope of field definition from DB2 or SQL/DS. The user need not worry about this since, in the transference of data from the table to the segment (and vice versa), conversions for compatibility take place automatically. The difference can be relevant to the user to the extent that it affects performance. In short, where possible, it is best to have the VISION:Transact field specification be the same as the relational field specification. The *Advantage VISION:Transact Reference Summary* shows the relational field types and their equivalent VISION:Transact field types.

Logical Relationship (LR) Statement

In conventional data base records, the relationship between a parent and a child segment is established implicitly by the physical position of the data in the record. With relational data bases, the segments (rows of a table) are physically separated and therefore the relationship must be established explicitly. The relationship is defined to VISION:Transact by the Logical Relationship definition statement, the LR statement.

LR statements perform two functions. They perform the function of relating or joining tables and the function of qualifying the rows of the tables that are to be the result of the join.

The following table lists the entries required for the LR statement.

Specification	Columns	Notes
File Name	1-8	
Statement Type	9-10	LR
Segment Name	11-18	Enter the name of the segment that is being qualified or filtered by the relational conditions. This is the same name as on the corresponding LS statement.
Delete?	19	Enter Y to delete a statement from an existing file definition.
Sequence Number	20-22	Enter a three character alphanumeric sequence number used to sequence the LR statements within the segment definition. This entry is required if a segment has multiple LR statements. This field is also used for file definition maintenance to insert LR statements and to delete them from the definition. To delete an LR statement the user must provide File Name, LR, Segment Name, Delete=Y, and Sequence Number. Duplicate sequence numbers are not permitted. An LR statement may only be changed by first deleting the old statement and then replacing it.
Logic Level	23	Leave blank or enter 0 to 9. This functions exactly as for PR statements.
Connector	24	Leave blank or enter O for OR or A for AND. This functions exactly as for PR statements.
Relational Field Name (Foreign Key)	27-44	Enter the name of a column of the table that is to be used as a foreign key to establish matching relationships. This name is the name that has been defined to the DB2 or SQL/DS catalog.
Operator	45-46	Enter one of the following: EQ, NE, GT, GE, LT, or LE as defined for the PR statements. Field B is required. NL=NULL. Tests the field for existence. No field B is required. NN=NOT NULL. Ensures that the field always exists. No field B is required.

Specification	Columns	Notes
B Qualifier	48	Leave blank if field B is a field name. Enter one of the following: C = field B is a character string constant. D = field B is a decimal constant.
Field B Constant or Primary Key	49-66	Enter the value that corresponds to the Qualifier used. Permissible qualifiers are C, D, or blank. Note that the definition of these qualifiers differs from that permitted on the PR statements. C = Character string data. D = Decimal data. This may have a leading sign, digits 0-9, decimal point, followed by digits 0-9. No other characters, including grouping characters, are permitted. blank = A VISION:Transact field name that has been defined in some ancestor segment (i.e., some higher level segment that is on the same branch of the tree as the current segment). The field named here may be a mapped field or unmapped field. A mapped field contains external name entry on the L0 statement whereas unmapped field contains no external name entry. If mapped fields are named here, then VISION:Transact uses the original retrieved contents of the fields when constructing the WHERE clause of all SQL statements. This is done to ensure that the same row retrieved is the same row updated. If unmapped fields are named here, then VISION:Transact uses the current contents of the fields when constructing the WHERE clause of all SQL statements. This allows the programmer to procedurally set the contents of unmapped fields prior to the retrieval operation for the segment (table) which uses the unmapped fields on the LR statement. The use of unmapped fields on LR statements allows programmers to relate tables based on calculated or derived argument values effectively creating a dynamic rule.

In the LR statements, mixed mode arithmetic is not permitted. Thus, LR statements must compare character string to character string or numeric to numeric.

LR statements may be provided for the root segment, but in this case, field B may not be a field name (qualifiers may only be C or D).

LR statements MUST be provided for all non-root segments.

Logical File Assignment

The files that are used in a VISION:Transact application are logical files, not physical files. That is to say, a VISION:Transact file does not always map one-to-one to a physical record or data base. This is particularly true in the case of relational data bases where a VISION:Transact record may be composed of data from several tables. It is the purpose of the RF and LR statements to provide the rules for mapping the physical data into the logical records as defined by the file definition and to establish the combination of logical records available for processing. The set of records available for processing is the logical data view. In VISION:Transact, the logical data view is both a function of the application logic (dynamically synchronized data bases) and the operator selected actions (screen synchronization). The logical data view at any instant is the set of logical records available for processing. The logical data view dynamically changes with time. The rules that establish the changes are contained in the set of RF and LR statements. For brevity, these statements are referred to collectively as the logical data view.

Basically, the RF statement identifies how the file is to be used in a VISION:Transact application; the LR statement defines the logical record.

File Selection (RF) Statement

The file selection (RF) statement identifies the logical file and the data that is to be assigned to it. Each logical file participates in a VISION:Transact application in a well-defined manner and it is the RF statement that provides the specifics for the processing of the file.

There should be an RF statement for each logical file in the application.

The RF statement assigns a qualifier to the data base in question so that fields from that data base may be referenced in the VISION:Transact request or procedure statements. For relational tables, it can also be used to provide the creator's identification (authorization ID). When used with relational data bases, the RF statement functions identically to the way that it works with DL/I data bases. In particular, screen synchronization and dynamic synchronization work the same way.

For non-relational files, the RF statements have their usual meanings.

For VISION:Transact, the following files may use relational tables or views:

FILE
FILE1-9

The following table lists the RF statement specifications for VISION:Transact and how they apply to relational files.

Specification	Columns	Notes
Statement Type	9-10	RF
File Name	11-18	Enter the name of the definition of the file as it exists in the VISION:Transact library (M5LIB) or as supplied in an APPGEN definition.
File Type	22-27	Enter FILE or FILEn (n=1-9) for a relational file.
Synchronizing Fields 1 and 2	28-43	The usual specifications apply consistent with the other specifications provided on the RF statement.
Sync Qualifier	52	Enter the usual qualifier.
Screen Sync Type	53	Has its usual meaning.
DL/I Path Call Usage	54	Leave blank for relational files.
Record Length Field Name	55-62	Leave blank for relational files.
Storgae Type	64	Not applicable for file type FILE or FILE1-9.
Authorization ID	65-72	Leave blank or enter the authorization ID of the creator of the tables or views. This entry must be left blank if an unqualified table, view, or synonym name is used on the LS statement for any of the segments within the relational data base hierarchy. If an entry is made, then the ID provided will be applied to any tables or views that are used to construct the record that do not have authorization ID specified on the associated LS statement. DB2 or SQL/DS accesses all tables (or views) with a fully qualified name: Authorization-ID.tablename.

Details of Operation and Examples (DB2 UDB)

Introduction

This chapter contains examples of using VISION:Transact with relational data bases.

[Chapter 14, “A Conceptual Overview \(DB2 UDB\)”](#) explained the concept of the logical record. In [Chapter 14, “A Conceptual Overview \(DB2 UDB\)”](#), we emphasized that once the logical record was constructed, the VISION:Transact processing was identical to that of VSAM and IMS (DL/I). Thus, screen control, screen synchronization, dynamic synchronization, interactions, request or procedure processing, etc., are all as described in this guide. For this reason, we do not focus upon building interactive applications, but rather, upon defining logical records. Once these definitions are created, it is business as usual.

We have included several examples. In some cases, we illustrate the same concept in a different context to give a different perspective on the functional capability.

The applications illustrated in this chapter demonstrate data retrieval from a data base. In this chapter, we do not consider updating tables. The subject of updating tables and considerations for preserving referential integrity is discussed in detail in [Chapter 17, “Updating and Designing for Referential Integrity \(DB2 UDB\)”](#).

In all of the examples in this chapter (and in this document), we use the convention of upper-case for physical data and lower-case for logical data. Thus, upper-case denotes a reference to the relational tables (including DB2 or SQL/DS views), whereas lower-case denotes a reference to the in-memory logical record.

We will also take liberty with the syntax of the VISION:Transact statements and show only enough to illustrate the points that are being demonstrated. If you want to try these examples, remember to fill in the missing specifications.

Finally, we will only show parts of the application. Specifically, we will omit the opening menu screen and associated screen control. We will generally begin at the construction of the output screen and the subsequent entries that activate the next interaction.

Note that the data bases used for examples and applications have been chosen to illustrate the capabilities of the relational support provided by VISION:Transact. For illustrative purposes, the examples have been kept simple. The data bases used are not always of the best design. However, this illustrates a powerful capability of VISION:Transact. In the real world, you are not always working with well designed data bases.

In many cases, your application development concentrates on trying to make sense of poorly structured data. The logical record capability of VISION:Transact provides you with the ability to avoid this problem by specifying that the data be presented in a logically ordered, well designed fashion. This is at the heart of VISION:Transact design; ask for the data in the way that you wish it were, and forget the way that it actually is.

As an aid to reading this chapter, the following is a brief review of the examples and the key points that are illustrated.

Example 1 Relational Tables Viewed as Separate Files.

This example illustrates an application which treats each table as a separate VISION:Transact file. The example shows that the application designer is obliged to program the synchronization of the tables (joins) and must use virtual fields and requests or procedures to establish the correct alignment of the rows of the table. This apparently straightforward approach is the hardest to program.

Example 2 Use of the Logical Record.

This example repeats the application of Example 1. It illustrates the use of the logical record. Application development is simplified as the screen and action definitions do not have to incorporate data synchronization considerations.

Example 3 Suppression of Duplicate Segments and Ordering Segments.

This example illustrates further extensions in the definition of logical records. Expressions consisting of multiple LR statements are shown, as well as demonstrating that relational table fields can participate in the LR selection rules without being mapped into the logical record. The example also shows the capability of automatically suppressing duplicate segments and the ability to order the segments in the record.

Example 4 Tables with Hierarchically Organized Keys.

This example differs only slightly different from the previous examples. It shows the creation of a logical record using a different context. The example shows that LR statements may refer to any previously generated segments and not just the immediate parent.

Example 5 Relational Tables Where the Keys are Not in Hierarchical Order.

This example contrasts with Example 4 in that it shows the keys and foreign keys between the relational tables do not have to exist in a strictly hierarchical order. In addition, the example illustrates the use of the LR statements on the root segment.

Example 6 Logical Record from Normalized Tables.

This example shows the construction of a logical record where the relational tables are normalized and the data base is organized with tables that act as connecting links between two other sets of tables. The example shows that the construction of the logical record is straightforward.

Example 7 A Different View of the Customer Order Data Base.

This example illustrates that different logical records can be created from the same physical data bases.

Example 8 A Variant of a Different View.

This example is an advanced version of Example 7. Example 7 generated several records which had to be gathered together by the browse capability. This example shows how to gather them all together within the same logical record.

Example 9 The Organization Chart.

This powerful example shows how a single table can be viewed as a four level logical record. Data is mapped from the rows of the relational table into different logical segments depending upon the LR rules provided. This example shows the solution to a problem which would take weeks to achieve in a 3rd generation language but is achieved in a few minutes using the logical record approach.

Example 10 The Suppliers of Red Parts.

Although no new capabilities are illustrated by this example, it is of interest since it is often quoted in texts on relational data bases. In this example, we have included the equivalent SQL statements that would be used to achieve the same results. It provides an interesting contrast of the VISION:Transact solution and the SQL solution.

Example 11 Another SQL Comparison.

This example also shows equivalent SQL code to contrast the VISION:Transact approach with native SQL.

Example 12 Multiple Column Browsing.

This example illustrates an efficient method that can be used to browse a table based on a range of values for multiple columns.

Sample Data

In the next few examples, we will use the sample data from the following three tables.

REGION_TABLE

REGION	DIV
EAST	WIDGETS
EAST	SPURS
WEST	LOCKS

DEPARTMENT_TABLE

DIVISION	DEPTNO	NAME	MANAGERNO
WIDGETS	A01	FINANCE	123
WIDGETS	B01	SALES	456
WIDGETS	C02	ENGINEERING	789

EMPLOYEE_TABLE

EMPNO	NAME	AGE	SALARY	DEPT	LOC	SEX
231	JONES	23	25000	B01	L65	M
234	SMITH	34	45000	B01	L80	M
123	WILSON	45	75000	A01	L65	M
477	BROWN	29	35000	B01	L80	F
456	WHITE	34	45000	B01	L75	F
333	GREEN	44	25000	C02	L80	M
789	BLACK	44	67000	C02	L80	M
567	BLUE	23	25000	A01	L65	F
444	ADAMS	23	25000	B01	L65	M

Column names must be unique within a table, but the same name may be used in different tables. For the sake of clarity, we have used different names for all of the columns with the one exception of the column name NAME. This name has been used in different tables simply to show how it is defined in the applications.

We assume that we have a personnel application which calls for the display of the data base through a screen of the form:

```

DIVISION: -----
DEPARTMENT NO: ---

DEPARTMENT NAME: -----
REGION: ----

  ACTN      EMPNO      NAME      AGE      SALARY      LOC
  -         -         -         -         -         -
  -         -         -         -         -         -
  -         -         -         -         -         -
  -         -         -         -         -         -
  -         -         -         -         -         -
  -         -         -         -         -         -

```

A typical screen would be:

```

DIVISION: WIDGETS
DEPARTMENT NO: B01

DEPARTMENT NAME: SALES
REGION: EAST

  ACTN      EMPNO      NAME      AGE      SALARY      LOC
  -         -         -         -         -         -
  -         231      JONES      23      25000      L65
  -         234      SMITH      34      45000      L80
  -         477      BROWN      29      35000      L80
  -         456      WHITE      34      45000      L75
  -         444      ADAMS      23      25000      L65

```

This screen enables the application user to display information about specific employees. The user specifies the division and department number. By entering an action of S and a specific employee number, the data concerning those employees will be displayed. The division, department number, action (S), and employee number may be entered on a blank screen or typed over previously displayed data.

Example 1 - Relational Tables Viewed as Separate Files

In this example, we will not use logical records. We will simply treat the application as one involving three flat files. If we want to be strictly accurate, we will use logical records, but each logical record will be a copy or a subset of the physical records. In Example 2, we will contrast this implementation with that of using the structured logical record capability.

We should state at the outset that this example is the most complex of all the examples. Intuitively, you might think that this is the most obvious way to use relational tables within VISION:Transact. Each VISION:Transact file represents a table and it only becomes necessary to manipulate (join) the tables. And, therein, is the fallacy. When you view the application as a set of tables, you absorb the extra responsibility of programming the usage of the tables. You have to do what a COBOL programmer would do and you lose many of the benefits of the automatic capabilities of VISION:Transact - particularly with respect to referential integrity, which is discussed in [Chapter 17, "Updating and Designing for Referential Integrity \(DB2 UDB\)"](#).

We start with this example because it does appear to be the “obvious” way to handle relational tables. Example 2 and the remaining examples in this chapter illustrate the far simpler use of the logical record.

First, we need three file definitions.

For the region table, the definition is:

			<u>KEY</u>	<u>EXTERNAL NAME</u>
div	FD		S	
	LS	divisn		REGION_TABLE
	L0	divname	1	DIV
	L0	region		REGION

For the department table, the definition is:

dept	FD		S	
	LS	deptmnt		DEPARTMENT_TABLE
	L0	divn	1	DIVISION
	L0	dptno	2	DEPTNO
	L0	deptname		NAME

For the employee table the definition is:

emp	FD		S	
	LS	emps		EMPLOYEE_TABLE
	L0	empnum	2	EMPNO
	L0	name		NAME
	L0	age		AGE
	L0	sal		SALARY
	L0	dept	1	DEPT
	L0	locn		LOC

In each of the above definitions, the files are defined in the usual way with additional specifications on the LS statement. The LS statement not only identifies the segment but also identifies the table that it represents. In like manner, the L0 statements define the logical fields and the fields (columns) of data that are to be retrieved from the data base to initialize the logical fields.

Only the fields that are required need be defined. For example, MANAGER_NO in the DEPARTMENT_TABLE and SEX in the EMPLOYEE_TABLE have not been included since they are not required. You may, however, define all fields in the table since VISION:Transact only retrieves those fields from a table that are referenced in the application.

We have used one capability that is unique to relational tables in these definitions. This is in the definition of the key fields. Since relational tables have no key fields intrinsic to the tables, we are at liberty to select any fields as the key fields of the logical record for the VISION:Transact processing. In our application, we wish to use the EMPLOYEE_TABLE in a browse mode to set up our output screen. For this reason, we select “dept” as the primary key to facilitate the browsing.

This apparently small point is very important. We would not be able to complete this application using flat VSAM or IMS files unless “dept” was defined to IMS (or VSAM) as a key. Given the nature of this flat file, that would be unlikely. The ability to select any field as a key of a logical record adds significant dimensions to developing applications.

Our screen is:

```

DIVISION: -----
DEPARTMENT NO: ---

DEPARTMENT NAME: -----
REGION: ----

ACTN      EMPNO      NAME      AGE      SALARY      LOC
-         -         -         -         -         -
-         -         -         -         -         -
-         -         -         -         -         -
-         -         -         -         -         -
-         -         -         -         -         -
-         -         -         -         -         -

```

Our screen definition looks something like:

```

      SEG   VSA      I/O  ACT
employee SD      A    M5XX

      SS 1 1

      S0 1      divname  B  M
      S0 1      S 1,divn  I  M
      S0 1      1,dptno  B  M
      S0 1      1,deptname 0
      S0 1      region   0

      SS 2 6

      S0 2      I  N      NM action
      S0 2      V 2,dept  I  X
      S1      M      IF S,action EQ C,S
      S0 2      2,empnum  B  X
      S1      U      IF S,action EQ C,S
      S0 2      2,name    B  P
      S0 2      2,age     B  P
      S0 2      2,sal     B  P
      S0 2      2,locn    B  P

```

In order to set the value of the department number in the virtual field S,dept (needed so that the employees from the correct department will be selected), we need to have a type 1 request or procedure for the screen, as follows:

Type 1 Request

```

setdept  ER      1
PR      S,dptno  R      S,dept

```

Type 1 Procedure

```

setdept  ER
##PROC
Let S.dept = S.dptno
##PEND

```

The virtual field and the request or procedure are necessary because each line of the repeated screen segment comes from a different root segment of the EMPLOYEE_TABLE. It is necessary, therefore, to provide the department number on each line. We could, of course, have generated it on the screen, but this would have made a cluttered screen as the value of the department number is the same for all the employees. Also, it occurs once in the top right hand corner of the screen. For these reasons, we keep our screen clean and internally, by means of the virtual field, generate the values that we need.

Our RF statements are:

```
M5XX          RC
              RF,div  FILE                M  DVOWNR
              RF,dept FILE1              M  DPOWNR
              RF,emp  FILE2      dptno dptno 1 M  EMPOWNR
```

As these RF statements show, we are assuming different creators (authorization IDs) for each of the tables.

This example is fairly straightforward and the construction of the application has little or no constraints imposed by relational technology. A little care has to be taken with the matching since we demand that the DEPARTMENT_TABLE be screen synchronized by matching on both "divn" and "dptno." Dynamic (browse) synchronization is required on the logical file "emp" to create the appropriate output screen. As we explained above, this requires setting up a virtual field to access the correct department from the EMPLOYEE_TABLE.

In this example, we direct VISION:Transact to retrieve all relevant data (Output Control = A on the SD statement) relying on the paging facility of the terminal monitor to view the data (because our screen definition is only set up to view six employees at a time).

Example 2 - Use of the Logical Record

In this example, we will repeat Example 1 using the logical record capability to implement the application.

Example 1 used the conventional approach of viewing the relational tables as separate logical files. The example, while simple, still required careful study. This is because not only did we have to concentrate on specifying the screen content, but we had also to worry about getting the correct data from the three files properly synchronized and this had to be done in different ways. For the "div" and "dept" files, we used screen synchronization. Specifically, we included fourth statement in the screen definition in order to synchronize the "dept" file. For the "emp" file, we used dynamic synchronization, included a virtual field (tenth and eleventh statements in the screen definition), a type 1 request or procedure to initialize it, and a browse capability on the RF statement in the logical data view.

The logical record concept enables us to define the relationships between the tables and get all the synchronization problems out of the way prior to screen definition. The construction of the screen definition and logical data view (RF statements) can then take place without concern for the data.

We start by defining the logical record.

To implement the application, we conceive (imagine) that we had a file in exactly the structure that we needed. We assume a file of the following structure:

File: product

Segments	Fields	TABLES
divisn	divname, region	REGION_TABLE
deprtmnt	dptno, deptname	DEPARTMENT_TABLE
emps	empnum, name, age, sal, locn	EMPLOYEE_TABLE

The definition is:

			SEG	LEV	KEY	AUTH-ID	EXTERNAL NAME
product	FD	S					
	LS	divisn	10			DVOWNR	REGION_TABLE
	L0	divname	10	1	1		DIV
	L0	region	10	1			REGION
	LS	deprtmnt	20			DPOWNR	DEPARTMENT_TABLE
	L0	dptno	20	2	1		DEPTNO
	L0	deptname	20	2			NAME
	LR	deprtmnt					DIVISION EQ divname
	LS	emps	30			EMPOWNR	EMPLOYEE_TABLE
	L0	empnum	30	3	1		EMPNO
	L0	name	30	3			NAME
	L0	age	30	3			AGE
	L0	sal	30	3			SALARY
	L0	locn	30	3			LOC
	LR	emps					DEPT EQ dptno

Each segment represents one of the tables. Since each table has a different creator (authorization ID), the fully qualified name of Authorization-ID.tablename is provided on the LS statement.

As with Example 1, we defined only the fields that are of interest to the sample application. In this definition, we do not include DIVISION from DEPARTMENT_TABLE, nor DEPT from EMPLOYEE_TABLE as these fields are not actually displayed by the application (we will, however, have a little to say about this on the subject of referential integrity in the next chapter). The fields “divn” and “dept” (the logical representations of DIVISION and DEPT) were used in Example 1 to synchronize the data retrieval from the different tables. In this example, we achieve this synchronization by using the logical relationship (LR) statement.

The LR statement provides the rule by which a row is retrieved from a table to become a subordinate segment of a previously retrieved parent segment. You may think of the process as if VISION:Transact first retrieves the root segment (based on screen input) and then assembles the logical record level by level. Decisions concerning which rows of a table are to be added to the logical record are made based upon the segments already constructed (i.e., rows already retrieved).

You may think of the LR statements as rules. The first LR statement provides the rule which says: "Include as a subordinate 'deptmnt' segment any row from the DEPARTMENT_TABLE where the field (column) DIVISION has the same value as the logical field 'divname' in the parent segment."

And the second LR statement is the rule: "Include as a subordinate 'emps' segment any row from the EMPLOYEE_TABLE where the field (column) DEPT has the same value as the logical field 'dptno' in the parent segment."

Using the sample data, a typical record would be:

```
WIDGETS  EAST
  A01  FINANCE
        123    WILSON    45    75000  L65
        567    BLUE     23    25000  L65
  B01  SALES
        231    JONES    23    25000  L65
        234    SMITH    34    45000  L80
        477    BROWN    29    35000  L80
        456    WHITE    34    45000  L75
        444    ADAMS    23    25000  L65
  C02  ENGINEERING
        333    GREEN    44    25000  L80
        789    BLACK    44    67000  L80
```

In other words, as far as the VISION:Transact application is concerned, it is as if this record actually exists. For reasons of efficiency, the complete record is not actually materialized; only the data that is seen on the screen is retrieved. But for the purposes of design, the record can be thought of as if it actually exists.

Our screen is the same as for Example 1:

```

DIVISION: -----
DEPARTMENT NO: ---
DEPARTMENT NAME: -----
REGION: ----

ACTN      EMPNO      NAME      AGE      SALARY      LOC
-         -         -         -         -         -
-         -         -         -         -         -
-         -         -         -         -         -
-         -         -         -         -         -
-         -         -         -         -         -
-         -         -         -         -         -
```


Our screen definition now looks something like:

	<u>SEG</u>		<u>I/O</u>	<u>ACT</u>	
empee	SD	A M5XX			
	SS 1 1				
	S0 1	divname	B	M	
	S0 1	dptno	B	M	
	S0 1	deptname	O		
	S0 1	region	O		
	SS 2 6				
	S0 2		I	N	NM action
	S0 2	empnum	B	X	
	S1			U	IF S,action EQ C,S
	S0 2	name	B	P	
	S0 2	age	B	P	
	S0 2	sal	B	P	
	S0 2	locn	B	P	

Only one RF statement is required:

M5XX	RC		
	RF,product	FILE	M

As you can see, the application becomes clear and is simpler to code. Virtual fields and associated requests or procedures are not required.

In Example 1, we accepted the data as it is and developed the application to cope with the situation. The penalty we paid was a degree of complexity.

In this example, we did not accept the data as it is but defined it to be the way that we wanted it to be. This led to a simpler application development.

It is this latter capability that provides the uniqueness of VISION:Transact. The ability to define the data the way that you want it is an immense improvement in the application development process.

It is, therefore, the logical record that is the most important feature of VISION:Transact's support of relational data bases. The examples in the remainder of this chapter focus upon the logical record capability.

Since this is the first logical record example, we will address a few features about logical records:

- The logical record approach allows for the integration of up to 99 relational tables or relational views into a single logical record. This makes for a maximum of 990 tables or views in a VISION:Transact application which is far more than the 10 available if each VISION:Transact file is treated as a single relational table or view.
- Up to nine key fields can be specified in each logical segment.

- In a logical record, each segment must represent a table (although the same table can be mapped to more than one segment as is illustrated in Example 9). All VISION:Transact segment key fields must be mapped from the table and all logical segments must be connected to their parent segments by a rule expressed as a (series of) LR statement(s).

Relational tables do not demand uniqueness of rows. In particular, it is possible to have multiple rows of the table with the same values for fields defined as VISION:Transact keys. This means that more than one row of the table could satisfy the selection criteria established by the combination of LR statements and screen supplied (synchronized) values. In this case, it is indeterminate as to which row will appear on the screen. For this reason, it is important that the fields selected as VISION:Transact keys uniquely identify a row of the table.

Example 3 - Suppression of Duplicate Segments

The logical record concept involves constructing records that contain only the data of interest and not all of the data available. Even if the file definition reflects all of the data available in the table, VISION:Transact subsets the definitions to include only the fields that have been referenced or are needed for the application. This frequently generates segments where the data is identical. For some applications it is useful to eliminate the duplicate data. An entry of Y in the Suppress Duplication? specification (column 30) on the LS statement will achieve this.

Example:

Assume that we wish to display the age and sex of all people who earn more than \$20,000 per year. The definition to achieve this is:

File: product

Segments	Fields	TABLES
divisn	divname	REGION_TABLE
deptmnt	dptno,deptname	DEPARTMENT_TABLE
emps	age,sex	EMPLOYEE_TABLE

The definition is:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>		<u>AUTH-ID</u>	<u>EXTERNAL NAME</u>
product	FD					S		
	LS	divisn	10				DVOWNR	REGION_TABLE
	L0	divname	10	1	1			DIV
	LS	deptmnt	20				DPOWNR	DEPARTMENT_TABLE
	L0	dptno	20	2	1			DEPTNO
	L0	deptname	20	2				NAME
	LR	deptmnt					DIVISION EQ	divname
	LS	emps	30				EMPOWNR	EMPLOYEE_TABLE
	L0	age	30	3	1			AGE
	L0	sex	30	3				SEX
	LR	emps		001	DEPT		EQ	dptno
	LR	emps		002 A	SALARY		GT	D,20000

Once again, our picture (definition) of the data is only that which we wish to see.

This definition also serves to show that logical relationships can consist of multiple LR statements connected with AND or OR. The rules may involve fields (columns) from the tables that are not in the logical records. In this example, DIVISION, DEPT, and SALARY are all fields that exist in the relational tables but are not included in the logical record. The example also illustrates that the rules may involve constants. The LR statements are the equivalent of the WHERE clause of an SQL SELECT or, in the IMS world, they serve a function similar to SSA's.

Using the sample data, a typical record would be:

```

WIDGETS
A01  FINANCE
    45  M
    23  F
B01  SALES
    23  M
    34  M
    29  F
    34  F
    23  M
C02  ENGINEERING
    44  M
    44  M
  
```

When displayed on a screen, the application user would see duplicate line entries, just as in the record representation. The screen might look like:

```

DIVISION: WIDGETSDEPARTMENT NO: B01

DEPARTMENT NAME: SALES

ACTN      AGE      SEX
          23      M
          34      M
          29      F
          34      F
          23      M
  
```

Had we modified the definition to request the suppression of duplicate data, the example would become:

File: product

Segments	Fields	TABLES
divisn	divname	REGION_TABLE
deprtmnt	dptno,deptname	DEPARTMENT_TABLE
emps	age,sex	EMPLOYEE_TABLE

The structure is unchanged; the definition merely controls the data that is included.

The definition is:

		SUPPRESS					
		SEG	LEV	DUPLICATION	KEY	AUTH-ID	EXTERNAL NAME
product	FD	S					
	LS divisn	10		Y		DVOWNR	REGION_TABLE
	L0 divname	10	1		1		DIV
	LS deprtmnt	20		Y		DPOWNR	DEPARTMENT_TABLE
	L0 dptno	20	2		1		DEPTNO
	L0 deptname	20	2				NAME
	LR deprtmnt					DIVISION EQ	divname
	LS emps	30		Y		EMPOWNR	EMPLOYEE_TABLE
	L0 age	30	3		1		AGE
	L0 sex	30	3				SEX
	LR emps				001	DEPT	EQ dptno
	LR emps				002 A	SALARY	GT D,20000

Using the sample data, a typical record now becomes:

```
WIDGETS
A01 FINANCE
  45 M
  23 F
B01 SALES
  23 M
  29 F
  34 F
  34 M
C02 ENGINEERING
  44 M
```

and the corresponding screen is:

```
DIVISION: WIDGETS      DEPARTMENT NO: B01

DEPARTMENT NAME: SALES

ACTN      AGE      SEX
          23      M
          29      F
          34      F
          34      M
```

Duplicate segments are automatically removed both from the record and from screen displays created from the record. Notice that a segment is removed from the logical record only if the entire segment (i.e., all fields) is duplicated. Duplicate keys alone do not cause the suppression of duplicate segments. Also, in this illustration, the order of the rows in the logical record is different. This is because the Suppress Duplication? action causes a sort (by DB2) which results in the elimination of duplicates and presentation of the rows in a different order.

While we illustrate the sorted order, we caution that it is unwise to rely upon the implicit sort by DB2. DB2 does not guarantee that the suppression of duplicates will also provide a sorted result. There may be circumstances where a sorted order is not returned. If the segments are desired in a sorted order, this should be explicitly identified on the LS statement as illustrated below.

Finally, in this example, we illustrate the capability of ordering the segments in the record and consequently on the screen. Segments may be defined to be unordered (as we have so far used them), in ascending order, or in descending order (including root segments). These specifications can be combined with the other specifications that we have used. The ordering specification is made on the LS statement.

The definition becomes:

		<u>SEG</u>	<u>LEV</u>	<u>SUPPRESS</u> <u>DUPLICATION</u>	<u>ORDR/</u> <u>KEY</u>	<u>AUTH-ID</u>	<u>EXTERNAL</u> <u>NAME</u>
product	FD						
							S
		LS	divisn	10			
		L0	divname	10	1	DVOWNR	REGION_TABLE DIV
		LS	deprtmnt	20			
		L0	dptno	20	2	DPOWNR	DEPARTMENT_TABLE DEPTNO
		L0	deptname	20	2		NAME
		LR	deprtmnt				DIVISION EQ divname
		LS	emps	30			
		L0	age	30	3	EMPOWNR	EMPLOYEE_TABLE AGE
		L0	sex	30	3		SEX
		LR	emps				
		LR	emps				
					001	DEPT	EQ dptno
					002 A	SALARY	GT D,20000

The record now can be visualized as:

```

WIDGETS
C02  ENGINEERING
    44  M
B01  SALES
    23  M
    29  F
    34  M
    34  F
A01  FINANCE
    23  F
    45  M
  
```

And the screen becomes:

```
DIVISION: WIDGETS      DEPARTMENT NO: B01
DEPARTMENT NAME: SALES

  ACTN      AGE  SEX
          23   M
          29   F
          34   M
          34   F
```

The capability of ordering segments can be very useful in designing well organized screens for production applications. Data presented in alphabetical or numerical order can greatly facilitate use by the application user.

There is one caveat here. The ordering of segments and the suppression of duplicate segments are actually performed by DB2 (or SQL/DS) per VISION:Transact's instructions. Achieving these results will require the data base manager to consume additional CPU cycles. This could be costly if there are no DB2 (SQL/DS) indices on the fields concerned. Creating indices on the tables is a data base manager function external to VISION:Transact.

Example 4 - Tables With Hierarchically Organized Keys

This example illustrates defining a logical record where the keys are in a strictly hierarchical organization.

Assume we have the following relational tables:

```
CUSTOMER TABLE
CUSTOMER_NO, CUSTOMER_NAME, CUSTOMER_PHONE

ORDER TABLE
CUSTOMER_NUM, ORDER_NUMBER, ORDER_DATE, PO_NUMBER, AGENT

ORDERED ITEMS Table
CUSTOMER_NUMBER, ORDER_NO, ITEM_NAME, ITEM_PRICE, QUANTITY
```

The relational names have been chosen to be different for clarity. In practice, since these are separate tables, it would be logical to use the same names for like fields, e.g., the customer number would probably be CUSTOMER_NUMBER in all three tables.

We want to create the following logical record:

Segments	Fields	TABLES
customer	custno, custname, cusphone	CUSTOMER TABLE
order	ordernum, orddate, ponum, agnt	ORDER TABLE
item	itemname, price, itemqty	ORDERED_ITEMS

The VISION:Transact file definition would resemble:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>	<u>EXTERNAL NAME</u>
cstmer	FD	S				
	LS	customer	10			CUSTOMER_TABLE
	L0	custno	10	1	1	CUSTOMER_NO
	L0	custname	10	1		CUSTOMER_NAME
	L0	cusphone	10	1		CUSTOMER_PHONE
	LS	order	20			ORDER_TABLE
	L0	ordernum	20	2	1	ORDER_NUMBER
	L0	orddate	20	2		ORDER_DATE
	L0	ponum	20	2		PO_NUMBER
	L0	agnt	20	2		AGENT
	LR	order			CUSTOMER_NUM	EQ custno
	LS	item	30			ORDERED_ITEMS
	L0	itemname	30	3	1	ITEM_NAME
	L0	price	30	3		ITEM_PRICE
	L0	itemqty	30	3		QUANTITY
	LR	item		001	CUSTOMER_NUMBER	EQ custno
	LR	item		002	A ORDER_NO	EQ ordernum

Summarizing briefly, the LS and L0 statements are defined in the usual way but augmented to add the 18-character relational names. Here, we do not show the creator's ID implying that all tables have the same authorization ID and this can be provided on the RF statement during APPGEN. The LR statements are included to make sure that the right subordinate segments are attached to the corresponding parents. Notice in the LR statements for segment "item" that the segments are attached to the correct "order" *within* "customer" and that the VISION:Transact field names apply to higher level segments but not necessarily the immediate parent ("custno" is in the root segment).

Example 5 - Relational Tables Where the Keys are Not in Hierarchical Order

This example is similar to Example 4, but here we assume that order numbers are unique across all customers. In this case, the table of ordered items would not contain the customer number (knowing the order number is sufficient to establish where the ordered items go). We are also only interested in customers with numbers in the range of 100 to 200 and the range 500 to 600.

The tables are:

```
CUSTOMER_TABLE
CUSTOMER_NO, CUSTOMER_NAME, CUSTOMER_PHONE

ORDER_TABLE
CUSTOMER_NUM, ORDER_NUMBER, ORDER_DATE, PO_NUMBER, AGENT

ORDERED_ITEMS_2
ORDER_NO, ITEM_NAME, ITEM_PRICE, QUANTITY
```

We want to create the following logical record:

Segments	Fields	TABLES
customer	custno, custname, cusphone	CUSTOMER_TABLE
order	ordernum, orddate, ponum, agnt	ORDER_TABLE
item	itemname, price, itemqty	ORDERED_ITEMS_2

Notice that this logical record has the identical structure to the logical record of Example 4 even though the physical tables are different (ORDERED_ITEMS_2 replaces ORDERED_ITEMS).

The VISION:Transact file definition would resemble the following:

		SEG	LEV	KEY	EXTERNAL NAME
cstmer	FD	S			
	LS	customer	10		CUSTOMER_TABLE
	L0	custno	10	1	CUSTOMER_NO
	L0	custname	10	1	CUSTOMER_NAME
	L0	cusphone	10	1	CUSTOMER_PHONE
	LR	customer	001	CUSTOMER_NO	GE D,100
	LR	customer	002	A CUSTOMER_NO	LE D,200
	LR	customer	003	CUSTOMER_NO	GE D,500
	LR	customer	004	A CUSTOMER_NO	LE D,600
	LS	order	20		ORDER_TABLE
	L0	ordernum	20	2	ORDER_NUMBER
	L0	orddate	20	2	ORDER_DATE
	L0	ponum	20	2	PO_NUMBER
	L0	agnt	20	2	AGENT
	LR	order		CUSTOMER_NUM EQ	custno
	LS	item	30		ORDERED_ITEMS_2
	L0	itemname	30	3	ITEM_NAME
	L0	price	30	3	ITEM_PRICE
	L0	itemqty	30	3	QUANTITY
	LR	item		ORDER_NO EQ	ordernum

Here, we show that LR rules can also be applied to the root segment thereby limiting the application's view to only those logical records of interest. Screen synchronization and dynamic synchronization subset the logical records that satisfy the rules on the LR statements for the root segment. Establishing the parentage of the "item" segment requires only reference to its immediate parent. This example shows that the relational tables do not have to have a series of fields on a strictly hierarchical order in order to generate a logical record.

The LR statements on the root segment are not used to establish a relationship but are used to filter the available records. This same filter could have been achieved by a request in the application. However, this would have cluttered up the understanding of the application. It is also more efficient in terms of CPU resources to include the filtering or selection statements as part of the logical record definition.

Example 6 - Logical Records From Normalized Tables

Examples 4 and 5 illustrated a customer order example with relational tables of slightly different structures. The tables used were not “normalized.” Normalized tables call for the tables to contain only data strictly relevant to that table. Normalized tables are focused and easy to maintain, however they tend to breed tables. You effectively end up with tables pointing to other tables. As relational data bases gain in popularity, you will encounter many examples of normalized data bases. Now, we illustrate our customer order example with normalized data bases.

The data is the same as in the previous examples but physically it is organized differently. The tables are:

```
CUSTOMER TABLE
CUSTOMER_NO, CUSTOMER_NAME, CUSTOMER_PHONE

CUST ORDER TABLE
CUSTOMER_NUMBER, ORDER_NUMBER, PO_NUMBER

ORDER TABLE 2
ORDER_NUM, ORDER_DATE, AGENT

ORDER ITEM TABLE
ORDER_NO, ITEM_NUMBER, QUANTITY

ITEM Table
ITEM_NO, ITEM_NAME, ITEM_PRICE
```

The logical record is:

Segments	Fields	TABLES
customer	custno, custname, cusphone	CUSTOMER TABLE
custord	ordernum, ponum	CUST_ORDER TABLE
order	orddate, agnt	ORDER TABLE 2
orditem	itemnum, itemqty	ORDER_ITEM TABLE
item	itemname, price	ITEM TABLE

At first glance, this structure may look rather strange or artificial. The reason is that conventional data base systems have habitualized us into designing segments and records that are optimal for processing by the data base manager in question. Thus, we have designed data bases to fit the constraints of the data base manager and which make the most effective use of the resources under these constraints. Short records “connecting” different pieces of data do not normally provide for efficient operation under conventional data base managers.

Normalized relational data bases force us to think in an orderly manner. Thus, our logical records (where a segment represents a table) become better organized. It is also evident in this example that a single occurrence of the level 2 segment has a single child level 3 segment (true also of levels 4 and 5). An occurrence of a segment with a single child is equivalent to the concatenation of the two segments. Viewed in this light, Example 6 is basically identical to Examples 4 and 5. This is not surprising since we have the same data only it is organized in a more segmented manner.

More tables lead to a greater depth of logical record, but that in itself does not lead to greater inefficiency.

The VISION:Transact file definition resembles the following:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>	<u>EXTERNAL NAME</u>
cstmer	FD	S				
	LS	customer	10			CUSTOMER TABLE
	L0	custno	10	1	1	CUSTOMER_NO
	L0	custname	10	1		CUSTOMER_NAME
	L0	cusphone	10	1		CUSTOMER_PHONE
	LS	custord	20			CUST ORDER TABLE
	L0	ordernum	20	2	1	ORDER_NUMBER
	L0	ponum	20	2		PO_NUMBER
	LR	custord			CUSTOMER_NUMBER EQ custno	
	LS	order	30			ORDER TABLE 2
	L0	orddate	30	3	1	ORDER_DATE
	L0	agnt	30	3		AGENT
	LR	order			ORDER_NUM EQ ordernum	
	LS	orditem	40			ORDER ITEM TABLE
	L0	itemnum	40	4	1	ITEM_NUMBER
	L0	itemqty	40	4		QUANTITY
	LR	orditem			ORDER_NO EQ ordernum	
	LS	item	50			ITEM TABLE
	L0	itemname	50	5	1	ITEM_NAME
	L0	price	50	5		ITEM_PRICE
	LR	item			ITEM_NO EQ itemnum	

This example illustrates how the normalized tables may be joined together to form a logical record. The LR statement on the “orditem” segment (segment 40) shows that a rule does not have to depend on its immediate parent (“ordernum” is in segment 20). Rules are generally formulated in terms of the immediate parent, but any ancestor segment will do. Actually, rules do not even have to be formulated using variable data. The “orditem” rule could have been:

```
LR  orditem  ORDER_NO      EQ C,134567
```

if this rule made sense within the context of the application.

Examples 4, 5, and 6 - A Screen Display

Examples 4, 5, and 6 have discussed the definition of logical records. But how does this affect the development of the applications?

We show a typical screen:

```

CUSTOMER NO.  ----- NAME -----
                                PHONE NO. -----

ORDER NO.  ----- DATE -----

ITEM      PRICE      QUANTITY
-----
-----
-----
-----
-----

ORDER NO.  ----- DATE -----

ITEM      PRICE      QUANTITY
-----
-----
-----
-----
-----

```

This is a three level screen. The screen definition resembles:

```

          SEG          I/O  ACT
scrndef  SD           A  M5XX
          SS 1  1

          S0 1        custno  B    M
          S0 1        custname 0
          S0 1        cusphone 0

          SS 2  2

          S0 2        ordernum 0
          S0 2        orddate  0

          SS 3  5

          S0 3        itemname 0
          S0 3        price    0
          S0 3        itemqty  0

```

Example 7 - A Different View of the Customer-Order Data Base

The previous examples have all viewed the Customer-Order data base from the same conceptual viewpoint - with customer as the root segment. This time we focus more on items than customers. Suppose that we would like to keep track of which customers had purchased a specific item.

We will assume the following tables:

```
CUSTOMER_TABLE
CUSTOMER_NO, CUSTOMER_NAME, CUSTOMER_PHONE

ORDER_TABLE
CUSTOMER_NUM, ORDER_NUMBER, ORDER_DATE, PO_NUMBER, AGENT

ORDERED_ITEMS_2
ORDER_NO, ITEM_NAME, ITEM_PRICE, QUANTITY
```

As we have said before, the logical record capability of VISION:Transact gives us the ability to define the record in the way that we would like it to be.

For this application, we would like to have a record of the following structure:

Segments	Fields	TABLES
itemseg	itemname, orderno	ORDERED_ITEMS_2
orderseg	custnum	ORDER_TABLE
custseg	custname	CUSTOMER_TABLE

This will give us a series of records with item as the root segment. This approach will generate a series of records with duplicate keys, one for each separate order. The browse feature of VISION:Transact can be used to gather customers for the same item onto the same screen.

The definition is:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>	<u>EXTERNAL NAME</u>
item	FD	S				
	LS	itemseg	10			ORDERED_ITEMS_2
	L0	itemname	10	1	1	ITEM_NAME
	L0	orderno	10	1		ORDER_NO
	LS	orderseg	20			ORDER_TABLE
	L0	custnum	20	2	1	CUSTOMER_NUM
	LR	orderseg				ORDER_NUMBER EQ orderno
	LS	custseg	30			CUSTOMER_TABLE
	L0	custname	30	3	1	CUSTOMER_NAME
	LR	custseg				CUSTOMER_NO EQ custnum

It might be argued that this is not new - a similar approach could be used with secondary indexing under IMS. This might be true for this example, but Example 9 is one that cannot be achieved by secondary indexing.

Example 8 - A Variant of a Different View

This example is for the experts.

Example 7 showed that by taking a different view, a series of item records could be constructed, each containing a customer. The browse capability is then used to gather the records (customers) for the same item together. As an illustration of the power of the logical record capability, it is possible to construct a logical record for each item that contains all of the customers. In this case, the browse capability does not have to be employed.

This is done by constructing an item root segment that is independent of any specific order and then establishing all of the orders as lower level segments. It effectively gathers all of the records created by Example 7 under the umbrella of a single root segment.

```
CUSTOMER TABLE
CUSTOMER_NO, CUSTOMER_NAME, CUSTOMER_PHONE

ORDER TABLE
CUSTOMER_NUM, ORDER_NUMBER, ORDER_DATE, PO_NUMBER, AGENT

ORDERED ITEMS 2
ORDER_NO, ITEM_NAME, ITEM_PRICE, QUANTITY
```

The structure is:

Segments	Fields	TABLES
itemroot	item	ORDERED_ITEMS_2
itemseg	orderno	ORDERED_ITEMS_2
orderseg	custnum	ORDER TABLE
custseg	custname	CUSTOMER TABLE

The definition is:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>	<u>EXTERNAL NAME</u>
items	FD	S				
	LS	itemroot	10			ORDERED_ITEMS_2
	L0	item	10	1	1	ITEM_NAME
	LS	itemseg	20			ORDERED_ITEMS_2
	L0	orderno	20	2	1	ORDER_NO
	LR	itemseg			ITEM_NAME	EQ item
	LS	orderseg	30			ORDER TABLE
	L0	custnum	30	3	1	CUSTOMER_NUM
	LR	orderseg			ORDER_NUMBER	EQ orderno
	LS	custseg	40			CUSTOMER TABLE
	L0	custname	40	4	1	CUSTOMER_NAME
	LR	custseg			CUSTOMER_NO	EQ custnum

This example shows that while each segment represents a table, there is no constraint that each segment represent a different table. Segments “itemroot” (segment 10) and “itemseg” (segment 20) both represent the same table (ORDERED_ITEMS_2). As a root segment, “itemroot” simply obtains the first qualifying record as dictated, perhaps, by screen synchronization. “itemseg” is governed by the LR rule which enforces that all rows with the selected item name will be brought into the record.

The value of doing it this way is:

1. This method performs all data synchronization independent of any screen facilities.
2. There is no need to become involved with the browsing capabilities.
3. The suppress duplicate feature can be used to remove duplicates from appearing on the screen.

The reason that records with duplicate keys are generated in Example 7 is that the tables in Example 7 are not normalized. The technique that we show in this example basically amounts to normalizing the tables on the fly.

Example 9 - The Organization Chart

In this example, we assume only one table, a personnel table containing a row for each employee. The table also contains each employee's manager's name. We assume that the president has no manager and consequently, in his case, the MANAGER_NAME field (column) is NULL.

The table is:

```
PERSONNEL TABLE
EMPLOYEE_NAME, MANAGER_NAME, EMPLOYEE_SALARY
```

The company hierarchy is:

```
President
Vice-President
Manager
Technical Staff
```

We want to develop an application that uses an organization chart (perhaps to display all managers). This is best done by creating a logical record of the form:

Segments	Fields	TABLES
pres	presname, pressal	PERSONNEL_TABLE
vp	vpname, vpsal	PERSONNEL_TABLE
mgr	mgrname, mgrsal	PERSONNEL_TABLE
tech	techname, techsal	PERSONNEL_TABLE

The VISION:Transact file definition looks like:

```

org      FD
        LS  pres      10
        L0  presname  10  1  1
        L0  pressal   10  1
                                PERSONNEL_TABLE
                                EMPLOYEE_NAME
                                EMPLOYEE_SALARY

        LR  pres      MANAGER_NAME NL

        LS  vp        20
        L0  vpname    20  2  1
        L0  vpsal     20  2
                                PERSONNEL_TABLE
                                EMPLOYEE_NAME
                                EMPLOYEE_SALARY

        LR  vp        MANAGER_NAME EQ presname

        LS  mgr       30
        L0  mgrname   30  3  1
        L0  mgrsal    30  3
                                PERSONNEL_TABLE
                                EMPLOYEE_NAME
                                EMPLOYEE_SALARY

        LR  mgr       MANAGER_NAME EQ vpname

        LS  tech      40
        L0  techname  40  4  1
        L0  techsal   40  4
                                PERSONNEL_TABLE
                                EMPLOYEE_NAME
                                EMPLOYEE_SALARY

        LR  tech      MANAGER_NAME EQ mgrname

```

Here we see the same table being used with the same fields to form a four-level hierarchy. It is the LR rules that make sure that the right rows are inserted into the right levels. The LR statement for the root segment establishes that the president is the individual with no manager (the test is that the MANAGER_NAME field is NULL).

To illustrate, we have taken a table such as the following:

EMPLOYEE_NAME	MANAGER_NAME	SALARY
WATERS	SIMPSON	65000
HILL	WATERS	50000
ANDREWS	SIMPSON	60000
WISE	THOMAS	35000
MILLS	THOMAS	26000
MARTIN	WATERS	45000
SIMPSON	(NULL)	85000
SMITH	MARTIN	33000
JACKSON	MARTIN	25000
KRULL	HILL	30000
THOMAS	ANDREWS	35000
ARNOLD	MARTIN	31000

And made it look like the following logical record:

```

SIMPSON      85000
  WATERS      65000
    MARTIN    45000
      SMITH    33000
      ARNOLD   31000
      JACKSON  25000
    HILL       50000
      KRULL    30000
  ANDREWS     60000
    THOMAS     35000
      MILLS    26000
      WISE     17000

```

The four LR statements shown are all that is required to construct the complete organization hierarchy. Again, we should emphasize that VISION:Transact does not, in fact, build this entire hierarchy; it builds only the pieces required.

Displaying the organization chart is now trivial.

It is important to understand this example since it is a powerful illustration of the logical record capability. How would you have solved this problem without the logical record capability? Secondary indexing can not solve this one.

You would have used a procedural approach.

- Pick up the first record.
- Is the manager field NULL?
- If not, go get the manager.
- Repeat all this until you find the president.
- Now somehow flag the record to keep the position.
- Etc.

This is a non-trivial problem which will take you several days or weeks and several computer runs to solve with a third generation language. Even other fourth generation languages cannot handle this problem elegantly. The logical record approach is a non-procedural approach to the solution which can be developed in less than 15 minutes and will run first time. Notice, that once you have defined the way that you want the data, the job is over.

Using this logical record, we can construct any suitable screen trivially. Consider the display of all managers:

A DISPLAY OF ALL MANAGERS

```
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----
```

The screen definition looks like:

	<u>SEG</u>			<u>I/O</u>
screen	SD	A	M5XX	
	SS 1	1		
	SS 2	8		
	S0 2	mgr		0

What does that make it - the entire problem solved in some 25 to 30 statements (screen definition requiring only 4)?

Example 10 - The Suppliers of Red Parts

This example repeats the techniques illustrated in the previous examples. In this example, we contrast with the equivalent SQL statements required to achieve the same results. We use the famous SUPPLIER and PARTS tables frequently used in illustrations involving relational tables.

The tables are:

```
SUPPLIERS
SUPPLIER_NAME, SUPPLIER_NUMBER

SUPPLIER_PART
PART_NO, SUPPLIER_NO

PARTS
PART_NUMBER, PART_NAME, COLOR
```

The problem is quite simple: Find the supplier names for the suppliers of red parts.

The SQL code is:

```
SELECT SUPPLIER_NAME
FROM SUPPLIERS
WHERE SUPPLIER_NUMBER IN
  (SELECT SUPPLIER_NO
   FROM SUPPLIER_PART
   WHERE PART_NO IN
     (SELECT PART_NUMBER
      FROM PARTS
      WHERE COLOR = 'RED'))
```

This code will only obtain the raw data from the tables. It must be converted into embedded SQL, and imbedded into other procedural code together with establishing a cursor in order to generate a suitable application.

Our solution is to use the logical record capability and view the file as consisting of records that are structured so that the root segment is the (red) parts segment.

The logical record structure is:

File: redparts

Segments	Fields	TABLES
part	partno	PARTS
suppart	suppnumb	SUPPLIER_PART
supplier	supplier	SUPPLIERS

The definition is:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>	<u>EXTERNAL NAME</u>
redparts	FD	S				
	LS	part	10			PARTS
	LO	partno	10	1	1	PART_NUMBER
	LR	part			COLOR	EQ C,RED
	LS	suppart	20			SUPPLIER PART
	LO	suppnumb	20	2	1	SUPPLIER_NO
	LR	suppart			PART_NO	EQ partno
	LS	supplier	30			SUPPLIERS
	LO	suppname	30	3	1	SUPPLIER_NAME
	LR	supplier			SUPPLIER_NUMBER	EQ suppnumb

This definition produces a series of records where the root segments are all red parts. The suppliers of the relevant parts are included in these records. The VISION:Transact screen definition statements become trivial once the approach is taken of defining the data the way that is most convenient for solving the problem.

In this example, since there are multiple logical records created, it will be necessary to use the browse capability to gather them on to the same screen.

As with Example 8, the records can be gathered together into a single logical record by creating a higher level root segment COLOR equal to red (and suppressing duplicates). All the different red items then become subordinate segments of this logical record.

The application can be generalized so that it is not limited to red parts. This is achieved by including "color" as a field in the root segment and making it the key field. The data for this field can then be provided from the screen. The definition becomes:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>	<u>EXTERNAL NAME</u>
redparts	FD	S				
	LS	part	10			PARTS
	LO	partno	10	1		PART NUMBER
	LO	color	10	1	1	COLOR
	LS	suppart	20			SUPPLIER PART
	LO	suppnumb	20	2	1	SUPPLIER_NO
	LR	suppart			PART_NO	EQ partno
	LS	supplier	30			SUPPLIERS
	LO	suppname	30	3	1	SUPPLIER_NAME
	LR	supplier			SUPPLIER_NUMBER	EQ suppnumb

The browse capability is then used to display all of the relevant data.

While this example contrasts the SQL approach with the VISION:Transact approach to the problem, it is worthy of note that the SQL statements merely define the retrieval of the data. Construction of screen displays and manipulation (use of cursors) must be performed in order to finish the job. With VISION:Transact, the task is almost over with the definition of the logical record.

Notice also, that the VISION:Transact application is easily extended, generalized, and modified. The same is not true of the SQL statements shown.

Example 11 - Another SQL Comparison

This example is derived from the example that illustrates correlated subselects in the IBM manual, IBM DATABASE 2 Concepts and Facilities Guide (GG24-1582). The data is from the personnel file and the application is to list the employees who earn more than their managers.

The tables are:

```
EMPLOYEE
EMPNO, LASTNAME, SALARY, DEPTNO

DEPARTMENT
DEPTNO, MGRNO
```

(In order to retain the identity of the original solutions, we have not changed the names in this example. Note that the name DEPTNO is used in both tables.)

Two solutions are provided:

Solution 1:

```
SELECT EMPNO, LASTNAME
FROM EMPLOYEE E
WHERE E.SALARY >
  (SELECT SALARY
   FROM EMPLOYEE M
   WHERE M.EMPNO =
     ( SELECT MGRNO
       FROM DEPARTMENT
       WHERE DEPTNO = E.WORKDEPT ))
```

Solution 2:

```
SELECT EMPNO, LASTNAME
FROM EMPLOYEE E
WHERE E.SALARY >
  ( SELECT M.SALARY
    FROM DEPARTMENT D,
        EMPLOYEE M
    WHERE D.MGRNO = M.EMPNO
          AND D.DEPTNO = E.WORKDEPT)
```

It is not the purpose of this document to be a tutorial on SQL, so these SQL examples are not explained. You should simply look at them and decide whether you understand the solution.

As we have illustrated above, the VISION:Transact approach to the problem is not to start with what we have, but to start with what we would like to have and then let VISION:Transact create it.

In this case, we would like to have the following structure:

```
mgrname,mgrsal
empname
```

In other words, for every manager we would like to subordinate all employees who have a salary that is greater than that of their manager.

Since we want to start with the managers and we can only get the manager from the DEPARTMENT table, we must construct a logical record with the DEPARTMENT table as the root segment.

The logical record is:

Segment	Fields	TABLES
dept	deptno,mgrno	DEPARTMENT
mgr	mgrname,mgrsal	EMPLOYEE
employee	empname	EMPLOYEE

The file definition is:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>		<u>EXTERNAL NAME</u>
mgrs	FD	S					
	LS	dept	10				DEPARTMENT
	LO	deptno	10	1	1		DEPTNO
	LO	mgrno	10	1			MGRNO
	LS	mgr	20				EMPLOYEE
	LO	mgrname	20	2	1		LASTNAME
	LO	mgrsal	20	2			SALARY
	LR	mgr				EMPNO	EQ mgrno
	LS	employee	30				EMPLOYEE
	LO	empname	30	3	1		EMPNAME
	LR	employee			001	DEPTNO	EQ deptno
	LR	employee			002 A	SALARY	GT mgrsal

As far as the application is concerned, it is as if there existed a file of records with one record for each department containing a list of all employees that earn more than their manager.

Once again, the solution to the problem becomes trivial once the data is chosen in the most suitable format.

As with Example 10, the SQL code must be converted to embedded SQL with the use of cursors before it can be included into a COBOL application. At that point, application programming can begin. Again, with VISION:Transact the application programming is almost over with the definition of the logical record.

Example 12 - Multiple Column Browsing

In this example, we illustrate a method that can be used to efficiently browse a table based on a range of values for more than one column.

Assume we have the following relational table:

```
DIVISION TABLE
DIVISION_NAME, GROUP_CODE, CATEGORY_CODE, CATEGORY_NAME
```

We need to browse this table with a specific value for the DIVISION_NAME column and a range of values for the GROUP_CODE and CATEGORY_CODE columns. If the range of values for these columns are static, the browse can easily be implemented as constants can be used on the LR statements to specify the ranges for the GROUP_CODE and CATEGORY_CODE columns. The following file definition could be used:

STMT TYPE	FIELD/ SEG NAME	SEQ NO	LEV NO	KEY FLD	EXTERNAL NAME	
LS	divtabl	01			DIVISION TABLE	
L0	division	01	1	1	DIVISION NAME	
L0	group	01	1	2	GROUP CODE	
L0	category	01	1	3	CATEGORY CODE	
L0	name	01	1		CATEGORY NAME	
STMT TYPE	FIELD/ SEG NAME	SEQ NO	CON	RELATIONAL FIELD NAME	OP	PARENT FLD NAME/CONSTANT
LR	divtabl	001		GROUP_CODE	GE	C,value1
LR	divtabl	002	A	GROUP_CODE	LE	C,value2
LR	divtabl	003	A	CATEGORY_CODE	GE	C,value3
LR	divtabl	004	A	CATEGORY_CODE	LE	C,value4

The file would be processed as a DSDB in VISION:Transact and the synchronizing field would supply the desired DIVISION_NAME value. Note that an DSDB synchronizes a file only on the primary root key (key field 1) whereas an SSDB synchronizes on all key fields. An SSDB requires the exact key values to be supplied for all key fields.

If the range of values for the GROUP_CODE and CATEGORY_CODE columns are dynamic (e.g., based on operator input), the use of unmapped fields in the file definition can be used to accomplish the browse efficiently. Note that dynamic synchronization cannot be used to accomplish this browse because dynamic synchronization only supports range browsing (start and end value) for the primary root key (key field 1).

The following file definition can be used to achieve the multiple column browse:

<u>STMT</u> <u>TYPE</u>	<u>FIELD/</u> <u>SEG NAME</u>	<u>SEG</u> <u>NO</u>	<u>LEV</u> <u>NO</u>	<u>KEY</u> <u>FLD</u>	<u>EXTERNAL NAME</u>	
LS	root		01		DIVISION_TABLE	
L0	division	01	1	1	DIVISION_NAME	
L0	grouplow	01	1			
L0	grouphi	01	1			
L0	catlow	01	1			
L0	cathi		01	1		
LS	level2	02			DIVISION_TABLE	
L0	group	02	2	1	GROUP_CODE	
L0	category	02	2	2	CATEGORY_CODE	
L0	name	02	2		CATEGORY_NAME	
<u>STMT</u> <u>TYPE</u>	<u>FIELD/</u> <u>SEG NAME</u>	<u>SEQ</u> <u>NO</u>	<u>CON</u>	<u>RELATIONAL</u> <u>FIELD NAME</u>	<u>OP</u>	<u>PARENT FLD</u> <u>NAME/CONSTANT</u>
LR	level2	001		DIVISION_NAME	EQ	,division
LR	level2	002	A	GROUP_CODE	GE	,grouplow
LR	level2	003	A	GROUP_CODE	LE	,grouphi
LR	level2	004	A	CATEGORY_CODE	GE	,catlow
LR	level2	005	A	CATEGORY_CODE	LE	,cathi

The file would be processed as a direct DSDB or an SSDB in VISION:Transact and the synchronizing field (DSDB) or screen field (SSDB) would supply the desired DIVISION_NAME value. When the root segment is retrieved, only one row is actually read from the DIVISION_TABLE for the DIVISION_NAME value supplied although multiple rows for the DIVISION_NAME exist on the table.

After the root segment is retrieved and prior to the retrieval of the LEVEL2 segment, the programmer would procedurally set the contents of the grouplow, grouphi, catlow, and cathi unmapped fields. These fields are unmapped since they do not have an external column name entry. The contents of these fields are set to the desired ranges. When the LEVEL2 segment occurrences are retrieved by VISION:Transact, the LR statements would cause all rows in the table to be retrieved for the supplied DIVISION_NAME and for the range of values supplied for GROUP_CODE (grouplow and grouphi) and CATEGORY_CODE (catlow and cathi). This technique results in the execution of two SQL SELECT statements for the table; one for the ROOT segment and one for the LEVEL2 segment.

Summary

This chapter demonstrated examples of the logical record concept. In many of the examples, we did not generate screen definitions or postulate applications. There is no need to do this. Once the logical record is defined, the VISION:Transact application definition is identical to that used for VSAM or IMS.

We have shown techniques of generating logical records which, at first, might not have been obvious. These techniques are straightforward, once one realizes that they mirror the relational data base design.

The technique of defining a segment (table) to be subordinate to a parent segment (table) is the equivalent of pre-defining an outer join to VISION:Transact. The logical record, therefore, defines all of the joins that will be required by the application. As is stated many times throughout this chapter, these joins are not, in fact, made unless the data is actually required for the application. This results in VISION:Transact optimizing the outer join.

The logical record enables you to specify the file structure that is most suitable for solving the problem and construct your application just as if the appropriate file existed. When required, the data will be materialized for the application.

The logical record concept can be applied to screen synchronized data bases and to dynamically synchronized data bases.

Updating and Designing for Referential Integrity (DB2 UDB)

Introduction

[Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#) discussed the logical record and how it is used to generate screen displays of appropriate data. [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#) focused on the retrieval and display of data from logical records. In this chapter, we discuss the ramifications of updating the data base. We start by describing the process that VISION:Transact follows to update the data base. We then discuss the subject of referential integrity, what it is, and how to design for it. A series of simple examples are included to illustrate the topics of discussion.

The VISION:Transact Updating of Relational Tables

The process of updating a relational table is intuitively what you would expect it to be, namely, a row is obtained, updated, and returned from whence it came.

VISION:Transact maps (retrieves) a row from a table into a logical segment by applying the rules provided by the LR statements. All segments except the root segment must have LR statements in the definition, and so there is always a rule. The root segment may have LR statements provided, in which case, there is a rule. If the root segment has no LR statements, all of the rows of the table in question are available as root segments. The application of the LR rules will, in general, produce several rows that satisfy the rules, and consequently, these will be mapped into repeated segments of a logical record. Within the logical record, instances of the segments are distinguished by the key fields defined for the segment. The set of key fields defined should be chosen to establish uniqueness across the segments. Key fields are not an integral part of relational data bases, and so, any fields in the segment may be chosen as key fields. You may define up to nine key fields in each segment including the root segment.

The VISION:Transact process can be conceptualized as follows:

1. Rows are retrieved from various tables according to the LR rules to construct a set of logical records.

2. Screen synchronization is applied against the set of logical records to provide a screen synchronized set of records.
3. Updating via the screen input and then subsequently through the requests or procedures is performed.
4. The screen synchronized subset of the logical record (now updated) is dismantled with the segments being returned to the original rows of the original tables.

We emphasize that this is a conceptual approach for understanding the process. VISION:Transact does not build all of these logical records; it retrieves only the data from rows needed during an interaction.

The segments are returned to the tables by invoking the rule that was used to retrieve them and with the values of the key fields as originally retrieved from the tables. If the retrieved rows are uniquely identified by the keys, this process will return the segments to their original table rows. All field values that are used, are the values as originally retrieved from the tables and not as modified by application logic, provided unmapped fields are not used in Operand B [48-66] of the LR statement. Using the original values ensures that the original row will be located. Thus, it is not possible to retrieve a row and modify the keys or fields used as Operand B of the LR statement in order to redirect the row back to update some other row. Note that if unmapped fields are used in Operand B of the LR statement, then the current contents, as modified by application logic, are used to retrieve and update rows in the table. Thus, the contents of unmapped fields used on LR statements must not be changed after retrieval and prior to update of the table.

In the process of retrieval of rows into segments and the subsequent replacement of segments into table rows, the role of the LR statements should be clearly understood. The LR statements provide rules for retrieval of rows into a logical record only. They are not a constraint on the external tables and specifically have no effect on output. For example, we could have LR statements that constrain our view to individuals with salaries less than \$20,000. The logical record in the VISION:Transact application will bring in all relevant rows. If the application changes the salary of an individual to \$24,000, that segment will be replaced into the table at the end of the transaction. On the next transaction, that individual will not satisfy the LR rule and therefore will not now appear in the logical record.

It is important to understand that from the point of view of an interaction, the original value of a key is the value that is on the screen at the time that the user presses the enter key (or equivalent). It is not the value that was on the screen prior to being modified by the user. That value is the output of the previous transaction. A transaction starts with the user pressing the enter key (or equivalent) and the values on the screen, at that time, are the original keys.

The process of updating a row is just what you would expect. However, there is a consequence in the use of relational tables that requires extra consideration when designing applications that update tables. This is the fact that operations against a relational data base are set operations.

Set Operations

Operations against relational tables are set operations. Set operations apply to all rows that satisfy the selection or row identification criteria. If the segments of the logical records are not unique and one of them is updated, it is exactly equivalent to all of them being updated in the same manner. The update will be applied to all rows of the table that satisfy the LR rule ANDed with the original key values.

From the above description, it is important that a group of key fields be identified in the logical record that establishes the uniqueness of the segments. It is not sufficient that the original rows in the table be unique; it must be the subset of fields selected as key fields for the VISION:Transact logical record that are unique. In [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#), we illustrated the following example with the logical record:

```
WIDGETS
  A01  FINANCE
        45  M
        23  F
  B01  SALES
        23  M
        34  M
        29  F
        34  F
        23  M
  C02  ENGINEERING
        44  M
        44  M
```

In the file definition, we defined “age” as the key field. As can be seen, the keys are not unique.

When displayed on a screen, the application user would see duplicate line entries, just as in the record representation. As the output of one transaction the screen might look like:

```
DIVISION: WIDGETS      DEPARTMENT NO: B01
DEPARTMENT NAME: SALES

  ACTN      AGE      SEX
          23      M
          34      M
          29      F
          34      F
          23      M
```

In the example above, the ACTN column is for entering action codes.

If we were to start another interaction to update this record (x is the code for update):

```
DIVISION: WIDGETS      DEPARTMENT NO: B01
DEPARTMENT NAME: SALES

  ACTN      AGE      SEX
  x          23      f
             34      M
             29      F
             34      F
             23      M
```

Then the resulting record displayed on the next interaction would be:

```
WIDGETS
  A01  FINANCE
        45  M
        23  F
  B01  SALES
        23  F  <-----
        34  M
        29  F
        34  F
        23  F  <-----
  C02  ENGINEERING
        44  M
        44  M
```

Both of the arrowed segments would be updated. Although there was only one action on the screen, the action is applied against all table rows satisfying the LR rule and having the same key. Since “age” has been defined as the key, all of the rows with the value of age=23 would be located and updated in exactly the same way.

The same logic applies with respect to the deletion of rows from a table. In our example, an attempt to delete the first entry with age 23, thus (d is the code for delete):

```
DIVISION: WIDGETS      DEPARTMENT NO: B01
DEPARTMENT NAME: SALES

  ACTN      AGE      SEX
  d          23      M
             34      M
             29      F
             34      F
             23      M
```

would result in both the rows with age = 23 (in the sales department) disappearing.

On the next interaction, the logical record available would be:

```
WIDGETS
  A01  FINANCE
        45  M
        23  F
  B01  SALES
        34  M
        29  F
        34  F
```

```
C02  ENGINEERING
    44  M
    44  M
```

Set operations also affect insert actions. An insert action will cause VISION:Transact to insert a logical segment into the screen synchronized logical record. Since this is a new segment, it will be inserted into the appropriate table at the end of the interaction. This will occur regardless of any prior entries in the table. Duplicate rows are permitted in tables so the insert will always take place unless a UNIQUE index has been created for the table (see [Chapter 18, “Helpful Hints \(DB2 UDB\)”](#)).

None of the above is really unique to relational data base systems. In all forms of data base management, if the keys are not unique, the results are seldom desirable.

Set operations are a consequence of the way that relational data base managers work. Although there may be only one occurrence of a segment in a logical record, it is possible to affect more than one row of the corresponding table. This can occur when the “Suppress Duplication” specification is used to enforce a unique segment in the logical record or with the root segment, which, by definition, is unique and will be the first row available or satisfying any LR statements. The update (insert or delete) process is the same. All rows of the table satisfying the LR statements with the identified keys are affected.

It is, therefore, very important that the keys be defined to establish uniqueness across the segments of the logical record. VISION:Transact permits the definition of up to 9 keys per segment and this is sufficient to establish uniqueness.

In the course of the above discussion, it should be clear that VISION:Transact puts back only that which it has retrieved. This means that VISION:Transact can only change fields in the logical record and those fields, on output, can update fields in the physical table. VISION:Transact, therefore, cannot update fields in relational tables which have not been defined in a file definition or used as result fields in the application.

If a field (column) of a table is to be updated, it must be defined as a mapped field in the file definition and used in the application as a result field, i.e.:

- Named as Operand C of a PR statement in a request,
- Used as the result field of a LET statement in a procedure, or
- Specified as the destination of a non-null field action in the screen definition.

A mapped field is one for which a logical field has been defined and a physical field has been specified from which to transfer data; an unmapped field is a field defined in the logical record that has no physical counterpart, i.e., the field exists only in memory and not in a table on disk.

Creation of Tables With VISION:Transact

It is possible to create new tables in a VISION:Transact application. This capability exists because relational data base managers support the concept of Null tables, i.e., the existence of a table with no rows of data. VISION:Transact itself does not issue CREATE TABLE statements. The tables must be created by use of SQL outside of VISION:Transact. Once physically created, they are available for use with VISION:Transact in the same way as any other table. Logical records and segments may be created in memory by the use of the I (insert) action code in the screen definition (S0 statement). When the logical record is returned to the physical tables, newly created segments will be inserted by VISION:Transact into the table by the use of the SQL INSERT verb. In this way, VISION:Transact can be used to “fill” empty tables with data.

Referential Integrity

The power of relational tables is derived from the fact that the tables have no implicit relationships between them. All relationships are established explicitly. This is generally achieved by the matching of one or more fields in each of the tables. Those rows that have matching field values satisfy a relationship for an application. In VISION:Transact, the matching fields are established by means of the LR statements. It follows, therefore, that if the value of a field in one table is changed but the value of the corresponding field in the second table is not changed, then the original relationship is no longer satisfied. Thus, the linkage has been broken and the integrity of the data base has been damaged. For example, in our sample tables of [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#),

REGION_TABLE

REGION	DIV
EAST	WIDGETS
EAST	SPURS
WEST	LOCKS

DEPARTMENT_TABLE

DIVISION	DEPTNO	NAME	MANAGER_NO
WIDGETS	A01	FINANCE	123
WIDGETS	B01	SALES	456
WIDGETS	C02	ENGINEERING	789

EMPLOYEE_TABLE

EMPNO	NAME	AGE	SALARY	DEPT	LOC	SEX
231	JONES	23	25000	B01	L65	M
234	SMITH	34	45000	B01	L80	M
123	WILSON	45	75000	A01	L65	M
477	BROWN	29	35000	B01	L80	F
456	WHITE	34	45000	B01	L75	F
333	GREEN	44	25000	C02	L80	M
789	BLACK	44	67000	C02	L80	M
567	BLUE	23	25000	A01	L65	F
444	ADAMS	23	25000	B01	L65	M

A change to the DEPARTMENT_TABLE of the Sales department number from B01 to E01 would result in the following table:

DEPARTMENT_TABLE

	DIVISION	DEPTNO	NAME	MANAGER_NO
	WIDGETS	A01	FINANCE	123
-->	WIDGETS	E01	SALES	456
	WIDGETS	C02	ENGINEERING	789

with the original EMPLOYEE_TABLE:

EMPNO	NAME	AGE	SALARY	DEPT	LOC	SEX
231	JONES	23	25000	B01	L65	M
234	SMITH	34	45000	B01	L80	M
123	WILSON	45	75000	A01	L65	M
477	BROWN	29	35000	B01	L80	F
456	WHITE	34	45000	B01	L75	F
333	GREEN	44	25000	C02	L80	M
789	BLACK	44	67000	C02	L80	M
567	BLUE	23	25000	A01	L65	F
444	ADAMS	23	25000	B01	L65	M

Here, we see that the integrity of the data base has been destroyed. There is no one in the Sales department (nobody has department code of E01) and there are several employees with a department code of B01 for which there is no department in the DEPARTMENT_TABLE. Maintaining the linkages by changing the values in both of the tables is known as maintaining the referential integrity of the data base.

Referential integrity means that when a single field value is changed in a data base, then all fields with values that match that field and are used to establish relationships must also be changed to the same value. In the case above, corresponding values of DEPTNO in DEPARTMENT_TABLE and DEPT in EMPLOYEE_TABLE must be changed at the same time.

The referential integrity of the database is also affected by inserting a row in a table and deleting a row from a table.

If we delete the Sales department from the DEPARTMENT_TABLE we get the table:

DEPARTMENT_TABLE			
DIVISION	DEPTNO	NAME	MANAGER_NO
WIDGETS	A01	FINANCE	123
WIDGETS	C02	ENGINEERING	789

This is just as bad as changing the department number, since we now have no department B01 but do have employees in the EMPLOYEE_TABLE with this department code.

Similarly, if we add a new employee with a department code of G34, we have an employee without a department.

In the relational data base terminology, preserving referential integrity demands that values of key fields and foreign key fields be kept synchronized.

Within relational data bases, there are no fields defined as key fields to the data base manager. However, for the purposes of an application, it is convenient to *think* of certain fields as key fields. Key fields within relational data bases are, therefore, a concept rather than a reality. Thinking of some fields (columns) of a table as key fields means that these fields will be used to match against fields in another table. These matching fields in the other table are known as foreign keys.

Since key fields are not defined explicitly to the relational data base manager, any field may assume the role of a key field. It is purely a function of the application. Since logical key fields (segment keys) have a significance to the operation of VISION:Transact, segment key fields must be defined to VISION:Transact. Any suitable field for solving the current problem may be chosen as a segment key field.

Designing for Referential Integrity

In order to preserve the integrity of the data base, corresponding fields in different tables must be changed at the same time or be maintained in synchronization. This is not a task that can be achieved automatically by a program. The problem is that, not only must the keys and foreign keys as they pertain to the current application be synchronized, so must all fields used as keys or foreign keys in all applications. This is not information known to any single program. Consider our Department-Employee example above. Suppose that we change the MANAGER_NO of the Finance Department (A01) to 888. Does this mean that WILSON who used to be the manager should have his employee number changed to 888? Should we be demanding that a new employee with employee number 888

be added to the EMPLOYEE_TABLE or is this simply an error? And here, we are dealing with a field that was not used as a key or foreign key in any of our examples in this document. It is clear, therefore, that the preservation of referential integrity is the responsibility of the application and the application must concern itself not only with the current task but with the integrity of the data base.

The VISION:Transact logical record capability provides the user with powerful capabilities that assist the user in the preservation of referential integrity. There are three VISION:Transact techniques which will be discussed:

1. The maintenance of values of fields in separate tables.
2. The insertion of rows in a table.
3. The deletion of rows from a table.

In Example 1 of [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#), we illustrated the processing of three separate tables without the use of a structured logical record. In this example, the application designer elected to handle all aspects of the application independent of the logical record facility. The designer accepts an added level of complexity since the automatic capabilities provided by the VISION:Transact logical record cannot be used. The designer has to handle all of the aspects of referential integrity. Sometimes the circumstances of the application or the data base are such that the designer would prefer not to have automatic actions against the data base. In our discussions of insertion and deletion, we illustrate instances where it is desirable to treat a table as stand alone rather than part of a logical record.

Maintenance of Values of Fields in Separate Tables

When the values of fields in rows of one table are updated, you must consider what other fields in what other tables are to be updated at the same time. VISION:Transact provides simple straightforward capabilities for updating both keys (i.e., fields in the table being updated) and foreign keys (i.e., fields in a related table).

To preserve referential integrity, it is crucial to define all the fields to be maintained in the segments that represent the tables. VISION:Transact can only update those physical fields on the tables for which it has a logical representation in a segment.

In Example 2 of [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#), we illustrated a logical record from our Department-Employee example.

We had the structure:

File: product

Segments	Fields	TABLES
divisn	divname, region	REGION_TABLE
deprtmnt	dptno, deptname	DEPARTMENT_TABLE
emps	empnum, name, age, sal, locn	EMPLOYEE_TABLE

The definition is:

		<u>SEG</u>	<u>LEV</u>	<u>KEY</u>	<u>AUTH-ID</u>	<u>EXTERNAL NAME</u>
product	FD					
	LS	divisn	10		DVOWNR	REGION_TABLE
	L0	divname	10	1		DIV
	L0	region	10	1		REGION
	LS	deprtmnt	20		DPOWNR	DEPARTMENT_TABLE
	L0	dptno	20	2		DEPTNO
	L0	deptname	20	2		NAME
	LR	deprtmnt			DIVISION EQ	divname
	LS	emps	30		EMPOWNR	EMPLOYEE_TABLE
	L0	empnum	30	3		EMPNO
	L0	name	30	3		NAME
	L0	age	30	3		AGE
	L0	sal	30	3		SALARY
	L0	locn	30	3		LOC
	LR	emps			DEPT EQ	dptno

And this gave rise to the following logical record:

WIDGETS						
A01	FINANCE					
	123	WILSON	45	75000	L65	
	567	BLUE	23	25000	L65	
B01	SALES					
	231	JONES	23	25000	L65	
	234	SMITH	34	45000	L80	
	477	BROWN	29	35000	L80	
	456	WHITE	34	45000	L75	
	444	ADAMS	23	25000	L65	
C02	ENGINEERING					
	333	GREEN	44	25000	L80	
	789	BLACK	44	67000	L80	

Now let us assume logic in the application changes the department code of the Sales department code from B01 to E01. The record now becomes:

WIDGETS						
A01	FINANCE					
	123	WILSON	45	75000	L65	
	567	BLUE	23	25000	L65	
E01	SALES					
	231	JONES	23	25000	L65	
	234	SMITH	34	45000	L80	
	477	BROWN	29	35000	L80	
	456	WHITE	34	45000	L75	
	444	ADAMS	23	25000	L65	
C02	ENGINEERING					
	333	GREEN	44	25000	L80	
	789	BLACK	44	67000	L80	

When this data is returned to the tables from which it originated, we get:

```

REGION      DIV
EAST        WIDGETS
EAST        SPURS
WEST        LOCKS

DEPARTMENT_TABLE
DIVISION    DEPTNO    NAME        MANAGER_NO
WIDGETS     A01        FINANCE     123
WIDGETS     E01        SALES       456
WIDGETS     C02        ENGINEERING 789

EMPLOYEE_TABLE
EMPNO      NAME      AGE  SALARY  DEPT  LOC  SEX
231        JONES    23   25000  B01   L65  M
234        SMITH    34   45000  B01   L80  M
123        WILSON   45   75000  A01   L65  M
477        BROWN    29   35000  B01   L80  F
456        WHITE    34   45000  B01   L75  F
333        GREEN    44   25000  C02   L80  M
789        BLACK    44   67000  C02   L80  M
567        BLUE     23   25000  A01   L65  F
444        ADAMS    23   25000  B01   L65  M

```

The integrity of the data base has been destroyed as there is a department E01 in the DEPARTMENT_TABLE with no employees are assigned to it and there are employees in the EMPLOYEE_TABLE in department B01 which does not exist. The situation arises because VISION:Transact maps a table to a segment in a one-to-one manner. VISION:Transact can only update data fields that were retrieved and used as result fields in the application. Since DEPT from the EMPLOYEE_TABLE was not included in the logical record, it cannot be updated. The solution is obviously to put DEPT in the logical record. We include it thus:

File: product

Segments	Fields	TABLES
divisn	divname, region	REGION_TABLE
deprtmnt	dptno, deptname	DEPARTMENT_TABLE
emps	empnum, <u>dept</u> , name, age, sal, locn	EMPLOYEE_TABLE

The definition is:

```

                SEG  LEV  KEY      AUTH-ID  EXTERNAL NAME
product  FD
LS divisn      10
L0 divname     10  1  1      DVOWNR  REGION_TABLE
L0 region      10  1
                DIV
                REGION

LS deptmnt     20
L0 dptno       20  2  1      DPOWNR  DEPARTMENT_TABLE
L0 deptname    20  2                DEPTNO
                NAME

LR deptmnt                DIVISION EQ divname

LS emps        30
L0 empnum      30  3  1      EMPOWNR EMPLOYEE_TABLE
L0 dept        30  3                EMPNO
L0 name        30  3                DEPT
L0 age         30  3                NAME
L0 sal         30  3                AGE
L0 locn        30  3                SALARY
                LOC

LR emps                DEPT      EQ dptno

```

This will now give rise to the logical record:

```

WIDGETS
A01  FINANCE
      123  A01  WILSON    45  75000  L65
      567  A01  BLUE     23  25000  L65
B01  SALES
      231  B01  JONES    23  25000  L65
      234  B01  SMITH    34  45000  L80
      477  B01  BROWN    29  35000  L80
      456  B01  WHITE    34  45000  L75
      444  B01  ADAMS    23  25000  L65
C02  ENGINEERING
      333  C02  GREEN    44  25000  L80
      789  C02  BLACK    44  67000  L80

```

At first, this looks like redundant data with the department number occurring in each subordinate segment as well as the parent segment. However, remember that VISION:Transact only actually creates the data dictated by the screen synchronization. Thus, we are generally talking of only a handful of fields where the data is actually repeated.

The mere existence of the field in the definition will not, in itself, resolve the updating. The field must be updated by the application before the physical data base is updated. This is achieved by the single statement of either:

```

Request
PR      dptno      R                dept

Procedure
Let dept = dptno

```

in a type M request or procedure. This single statement will update all occurrences of "dept" in the lower level segment.

- If the segments in question have already been screen synchronized, only the synchronized segments (rows) will be updated.

- If the segments have not been synchronized, the statement itself will dynamically extend the synchronization to update all segments of the logical record (and subsequently the tables) that satisfy the LR rules.

It is recommended that statements which update keys and foreign keys be put into a separate request or procedure, one for each segment, even if there is only one statement. Obviously, the request(s) or procedure(s) must come after any that update the fields by means of PR or ASL statements.

Thus, referential integrity is easily maintained by including all of the keys and foreign keys in the segment definition and including the statements to update the fields in a type M request or procedure. In general, there will be only one PR or ASL statement per key or foreign key to be maintained.

Other rules of integrity can also be established in this manner. In our examples, we have not used the field `MANAGER_NO` in the `DEPARTMENT_TABLE`. It is reasonable to assume that someone may only be a manager if the person is actually a member of the department. Assume that "mgrno" is the representation of `MANAGER_NO` in the "dprtmnt" segment. In an interim between the departure of the current manager and the arrival of new manager, blank out the manager number.

The file structure and contents are:

File: product

Segments	Fields	TABLES
divisn	divname, region	REGION_TABLE
deprtmnt	dptno, deptname, mgrno	DEPARTMENT_TABLE
emps	empnum, dept, name, age, sal, locn	EMPLOYEE_TABLE

The relevant integrity check is achieved by the following simple statements:

Request

```
PR empnum EQ mgrno
PR A dptno NE dept
PR R C, mgrno
```

Procedure

```
IF empnum EQ mgrno AND,
  dptno NE dept
  LET mgrno = ' '
END
```

Modification of Keys and Foreign Keys

It is, of course, possible to set up a series of requests or procedures that actually modify the keys and foreign keys as part of the application. As a result of such actions, we could update the department numbers of our logical record thus (presupposing an appropriate screen definition and set of requests or procedures):

```
      WIDGETS
      A01  FINANCE
----->      123  B01  WILSON    45  75000  L65
              567  A01  BLUE     23  25000  L65
      B01  SALES
----->      231  C02  JONES     23  25000  L65
----->      234  A01  SMITH     34  45000  L80
              477  B01  BROWN    29  35000  L80
----->      456  A01  WHITE     34  45000  L75
              444  B01  ADAMS     23  25000  L65
      C02  ENGINEERING
----->      333  B01  GREEN     44  25000  L80
              789  C02  BLACK     44  67000  L80
```

The changes to the department numbers are made on the lines indicated with arrows.

These changes modify the field values but do not change the logical record structure as it is in memory. The application will “see” the logical record above. Upon update, the tables will be updated thus:

REGION_TABLE

REGION	DIV

EAST	WIDGETS
EAST	SPURS
WEST	LOCKS

DEPARTMENT_TABLE

DIVISION	DEPTNO	NAME	MANAGER_NO
WIDGETS	A01	FINANCE	123
WIDGETS	B01	SALES	456
WIDGETS	C02	ENGINEERING	789

EMPLOYEE_TABLE

EMPNO	NAME	AGE	SALARY	DEPT	LOC	SEX
231	JONES	23	25000	C02	L65	M
234	SMITH	34	45000	A01	L80	M
123	WILSON	45	75000	B01	L65	M
477	BROWN	29	35000	B01	L80	F
456	WHITE	34	45000	A01	L75	F
333	GREEN	44	25000	B01	L80	M
789	BLACK	44	67000	C02	L80	M
567	BLUE	23	25000	A01	L65	F
444	ADAMS	23	25000	B01	L65	M

On the next transaction therefore, the logical record becomes:

```

WIDGETS
  A01  FINANCE
        567  A01  BLUE    23   25000   L65
        234  A01  SMITH   34   45000   L80
        456  A01  WHITE   34   45000   L75
  B01  SALES
        123  B01  WILSON  45   75000   L65
        477  B01  BROWN  29   35000   L80
        444  B01  ADAMS   23   25000   L65
        333  B01  GREEN   44   25000   L80
  C02  ENGINEERING
        231  C02  JONES   23   25000   L65
        789  C02  BLACK   44   67000   L80

```

We started this discussion by an illustration that showed the structure:

File: product

Segments	Fields	TABLES
divisn	divname, region	REGION_TABLE
deptmnt	dptno, deptname, _____	DEPARTMENT_TABLE
emps	empnum, name, age, sal, locn	EMPLOYEE_TABLE

We commented that a change in deptno was not reflected in the EMPLOYEE_TABLE (“emps” segment) because the DEPT column of the table was not defined in the “emps” segment. This very capability is an important part of referential integrity. Referential integrity means controlling the integrity of the data base. One important aspect of controlling the integrity is inhibiting the update of fields. This is easily done in VISION:Transact by not defining them to VISION:Transact.

In this discussion, we have mentioned modifying keys by requests. Note that VISION:Transact keys cannot be modified directly from the screen by screen actions. This is simply because it is the data on the screen, when the user presses enter, that dictates the logical (screen synchronized) record available for processing. Key values on the screen do not change anything; they start the interaction.

Thus, we see that updates can be prevented or easily synchronized with VISION:Transact. In addition, sophisticated referential checks are easy to include in your application.

Insertion of Rows in a Table

The structured logical record greatly assists in preserving referential integrity with the insertion of rows into a table since the mechanics of the VISION:Transact transaction insertion capability require the existence of parent segments. You can not insert a segment without the appropriate parent segments existing. These parent segments must either exist or be inserted in the same transaction. Since parent segments in a logical record are simply other related tables, the VISION:Transact process ensures rows exist or are inserted into all related tables.

For example, assume that we had a logical structure of:

```
customer
|
order
|
item
```

It is simply not possible to define a series of actions to insert an item without having a customer and an order. Without the logical structure, it is perfectly possible via SQL or a COBOL application to add an item without an order, or an order without a customer. Such independent actions can serve to destroy the integrity of the data base.

Thus, the logical record, merely by its existence, provides a means of controlling referential integrity when rows are to be inserted into a table.

With a little imagination, the VISION:Transact capabilities can be employed to handle tables not used in the application and assist in preserving the integrity of the entire data base and not just the tables used in the application. In [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#), Example 2, we structured a logical record:

<u>Segment</u>	<u>Table</u>
divisn	REGION_TABLE
deptmnt	DEPARTMENT_TABLE
emps	EMPLOYEE_TABLE

Here, the insertion of a new employee requires that a division and department exist.

Now, suppose there is also a table that records the employees retirement plan contributions (RETIRE_TABLE) and another table that records the vacation hours (VAC_TABLE). As an employee is added to the EMPLOYEE_TABLE, we would like to add a row to each of the other two tables. To do this, we include logical segments that represent the tables into the logical record. The structure is:

File: product

Segments	Fields	TABLES
divisn	divname, region	REGION TABLE
deptmnt	dptno, deptname, mgrno	DEPARTMENT TABLE
emps	empnum, name, dept, age, sal, locn	EMPLOYEE TABLE
retire	empno	RETIRE TABLE
vac	empnumb	VAC TABLE

The definition is:

			<u>SEG</u>	<u>LEV</u>	<u>KEY</u>	<u>EXTERNAL NAME</u>
product	FD	S				
	LS	divisn	10			REGION_TABLE
	L0	divname	10	1	1	DIV
	L0	region	10	1		REGION
	LS	deprtmnt	20			DEPARTMENT_TABLE
	L0	dptno	20	2	1	DEPTNO
	L0	deptname	20	2		NAME
	LR	deprtmnt				DIVISION EQ divname
	LS	emps	22			EMPLOYEE_TABLE
	L0	empnum	22	3	1	EMPNO
	L0	dept	22	3		DEPT
	L0	name	22	3		NAME
	L0	age	22	3		AGE
	L0	sal	22	3		SALARY
	L0	locn	22	3		LOC
	LR	emps				DEPT EQ dptno
	LS	retire	24			RETIRE_TABLE
	L0	empno	24	4	1	RETIRE_EMPNO
	LR	retire				RETIRE_EMPNO EQ empnum
	LS	vac	26			VAC_TABLE
	L0	empnumb	26	5	1	VAC_EMPNO
	LR	vac				VAC_EMPNO EQ empnum

The retirement and vacation segments are somewhat simple as they contain only the employee number. The application can now be set up to ensure that whenever an insert is made for the “emps” segment, inserts are also made for the “retire” and “vac” segments. The net result is to update three tables with a single screen action. The screen would be the same screen that we have used in other examples:

DIVISION: -----		DEPARTMENT NO: ---			
DEPARTMENT NAME: -----REGION: ----					
ACTN	EMPNO	NAME	AGE	SALARY	LOC
-	---	-----	--	-----	---
-	---	-----	--	-----	---
-	---	-----	--	-----	---
-	---	-----	--	-----	---
-	---	-----	--	-----	---
-	---	-----	--	-----	---

The screen definition would look like:

```

                SEG      VSA      I/O  ACT
employee  SD          A  M5XX

          SS   1   1

          S0   1      divname  B   M
          S0   1      dptno   B   M
          S0   1      deptname O
          S0   1      region  O

          SS   2   6

          S0   2              I   N      NM action
          S0   2      empnum  B   X
          S1              I
          S0   2      S  empno  I   Q      IF S, action EQ C,N
          S0   2      S  empnumb I   Q
          S0   2      name   B   P
          S0   2      age    B   P
          S0   2      sal    B   P
          S0   2      locn   B   P

```

We are using the same screen field to control three different segments.

Our RF statement is:

```

M5XX      RC
          RF,product  FILE              U  DBOWNER

```

Notice that we must define “empnum” and “dept” in the “emps” segment in the same way that we were required to define them under the discussion on modification of values. As with that example, values for “dept” must be inserted into the logical record via a request. Thus, the inserted segment will give rise to a new row on the table with data provided for the following fields:

Logical Field	Physical
empnum	EMPNO
dept	DEPT
name	NAME
age	AGE
sal	SALARY
locn	LOC

The field SEX, which exists in the table, cannot be updated by the insert action since it has no representation in the logical segment. This field will be set to the value of NULL by the data base manager. If, when the CREATE TABLE was established, the field SEX were defined to be NOT NULL, then the VISION:Transact APPGEN will issue an error if the field is not defined and not used as a result field or key field in the application, thus, preventing the insert from failing. *Note that only those fields defined as key fields or used as result fields in the application are included on an insert.*

In the above example, we were concerned with the referential integrity of the data base and wanted to make sure that whenever a row was inserted in one table, corresponding rows were inserted into other tables. We achieved this by use of the logical record capability and employing the VISION:Transact requirements for a hierarchical structure.

But suppose that we wanted to insert a lower level segment (table) and did not want to update the parent segments, or even care if they existed. For example, in the following structure, we may well wish to insert the item segment (table) without concern for the customer or order segments (tables).

```
customer
  |
order
  |
item
```

The truth is, that you should not be using this application to insert rows into the item table. If you must update/insert customers, orders and items in the same application, and you do not want automatic referential integrity on the item segment then you must use a different set of logical records. An appropriate set might be:

```
customer      item
  |
order
```

This will demand two RF statements to establish synchronization between the tables. With this structure, inserts may be performed on each logical record without affecting (the tables in) the other.

The point is that the VISION:Transact support of relational data bases does not constrain you to work with a specific set of structures as dictated by the data base manager. Using relational data bases, you can select the appropriate logical structures to solve the problem at hand. Part of this involves selecting the structures that support the needs of referential integrity. If it is desired (as in Example 1, [Chapter 16, "Details of Operation and Examples \(DB2 UDB\)"](#)), each of the tables can be handled separately without the use of the logical record capability. In this case, processing considerations would be similar to those required of a COBOL application. The application designer gains control at the expense of the automatic functions provided by the logical record. On the other hand, as we have shown above, either single or multiple logical records can be used. It all depends on what you want to control and what you want VISION:Transact to do for you.

Deletion of Rows From a Table

The logical record capability can be used to preserve the referential integrity of the data base when a row is removed from a table. In this case, it may well be that when a row is deleted from a table, it should be deleted from several other tables.

The VISION:Transact rules for the deletion of a segment automatically cause the deletion of all subordinate segments. With the appropriate logical structure, therefore, a single delete action from a screen can delete several different rows from different tables. Using the example above where we wished to insert into several tables, we may assume that we wish to delete from several tables. We have the logical structure:

File: product

Segments	TABLES
divisn	REGION TABLE
deprtmnt	DEPARTMENT TABLE
emps	EMPLOYEE TABLE
retire	RETIRE TABLE
vac	VAC_TABLE

The deletion of an employee by a single screen entry to delete the “emps” occurrence will delete the relevant “emps” occurrence and the subordinate “retire” and “vac” segments (each of which occurs only once in this example). This single screen entry will remove the relevant employee record from the EMPLOYEE_TABLE, RETIRE_TABLE, and VAC_TABLE. Thus, three related rows are removed from three separate tables by a single screen action.

The delete action is also a set operation and if the keys are not uniquely established, the delete action will remove all rows that satisfy the LR rules and have the same VISION:Transact key value.

While the logical record greatly assists in the preservation of referential integrity when rows are to be deleted from tables, there are always situations where it is not desired to cascade the delete to lower level segments.

Consider again the logical record structure:

```
customer
|
order
|
item
```

Here, in the course of the deletion of a “customer” record, or perhaps more realistically an “order” record, we may not want to delete the items from the “item” table. As with the update situation, the solution lies in selecting the logical records that provide support for the application. In the case cited, we may wish to have a the following structures:

```
customer      item
|
order
```

Once again, this requires two RF statements to synchronize the files (tables). If desired, each of the tables can be handled separately. This requires three RF statements for synchronization of the tables, but in this case, deletions against one table will not affect the other tables.

The Order of Updating Rows

In this chapter, the term *update* is used in its broadest sense to mean any change to the table. Update includes the functions of DELETE (removal of rows) and INSERT (creation of rows).

The logical record concept used within VISION:Transact involves mapping records from relational tables into logical in-memory records, applying screen transactions, and returning the updated rows to the tables from where they came. We have explained how the rows are returned to their original position by the use of the LR rules used for retrieval and with the VISION:Transact key fields. By using the original data, we ensure that the same rows are located.

The above process assumes that none of the tables are changed between retrieving the rows and returning them. If any of the values of the keys or foreign keys in the rows of the tables are changed within an interaction (i.e., after screen data has been entered and before the output screen is displayed), it follows that VISION:Transact will not be able to find the original rows when attempting to return the data. Keys and foreign key fields (columns) can be changed by actually changing the values or by deletion of the rows.

Tables can be changed between retrieval and update in five ways:

1. Another application changes the table.
2. A table is represented in different FILES of a logical data view and is updated in more than one of the files.
3. A table is represented by different segments at different levels in the same logical view.
4. Repeated segments (rows) of the screen synchronized record change the values of the keys.
5. An unmapped field is used in Operand B [48-66] of an LR statement and the contents of the unmapped field are changed after retrieval and prior to update of the segment (table).

Deletions and insertions can only be performed by screen actions but updates can be performed both by screen actions and requests or procedures.

Case 1 can be prevented by invoking the appropriate relational data base locking facility. The remaining four cases are application design considerations.

These situations may be caused by the fact that as VISION:Transact dismantles a logical record, returning the updated segment to the relational table, it does it one segment at a time. After each segment is returned, the table is now modified for the subsequent segments. This means that if a row is updated via two separate segments, the second update will be impacted by the results of the first update. If the first update does not affect keys or foreign keys, then the second update will override the results of the first update. This is exactly the same as providing two

transactions against a conventional file. If, however, the first update changes the keys or foreign keys, then the second update will not be able to find its original table row. The data base will be considered inconsistent and the transaction will be rejected with the appropriate data base rollback (see [Chapter 18, "Helpful Hints \(DB2 UDB\)"](#)).

Cases 2 and 3 are similar. Consider what would happen if a delete were performed against one segment representing a table while an update were performed on a different segment representing the same table. The deletion gets performed and then VISION:Transact requests the relational data base manager to do the update. But the row is no longer there to update! The data base will be deemed inconsistent and the transaction terminated (see [Chapter 18, "Helpful Hints \(DB2 UDB\)"](#)). You can easily avoid problems of the nature of cases 2 and 3 by ensuring that updates to a table occur only via one segment representing the table. So, when a table is represented by more than one segment in an application, all of the updates should be made against only one of these segment types.

Case 4 is a little different. The actions and data are specified by the applications user. The user is generally viewing several repeated segment occurrences (rows of a table). If, on the first row of the screen, the user specifies a delete action and a key value, while on the second row the user specifies an update action with the same key value, then we have the same situation as for cases 2 and 3 above. First, the delete takes place, then the update is attempted. The results are the same as described above. You can also prevent this situation very simply by not permitting updates on key fields. Display the key fields as output fields.

Case 5 is very easy to prevent. If unmapped fields are used on LR statements, their contents should be established prior to retrieval and, if updates are to occur, their contents should not be modified prior to the update (type 2 and M requests and procedures execute prior to an update).

In some applications, it is necessary to be able to update key fields. In this case, you should include a type 1 request or procedure to monitor for this eventuality. Various techniques exist, but essentially the new key data must be supplied on the screen as a non-key field. After the appropriate screen synchronized segments are retrieved, then the type 1 request or procedure can move the new key data to the correct field.

When VISION:Transact is "dismantling" a record, the sequence of events is as follows:

Update and Insert Actions:

The logical record segments are processed from the top down.

Delete Actions:

The logical segments are removed (deleted) from the bottom up.

Summary

We have discussed the VISION:Transact updating of relational data bases and pointed out the powerful assistance that VISION:Transact can provide to support referential integrity. Basically, VISION:Transact puts the rows of data back from where it got them. Since relational data bases perform set operations, if the rows are not uniquely identified, more than one row may be affected by an update or delete.

VISION:Transact also updates only those fields used as result fields in the application. For inserts, only those fields defined as key fields or used as result fields are included.

A field is a result field in the application if it is:

- Named as Operand C of a PR statement in a request,
- Used as a result field of a LET statement in a procedure, or
- Specified as the destination of a non-null field action in the screen definition.

VISION:Transact has a very powerful capability that enables *logical views* of the data base. These views can involve a table taking part in the view at different levels of the logical record. When this occurs, care must be taken that the updates or deletes of one representation do not interfere with the updates or deletes of the other representations.

Helpful Hints (DB2 UDB)

Introduction

In this chapter, we discuss a variety of miscellaneous topics to round out your understanding of the relational support of VISION:Transact. In some cases, we repeat information that is discussed in other parts of this document. We repeat the information here since it is sometimes easy to overlook a point that is made while imbedded in a larger chapter of a document.

Setting Up and Running a Job for DB2

Here, we provide a summary and overview on how to run a job. A complete description of the JCL and all of the steps is contained in the *Advantage VISION:Transact for CICS Environment Guide*. We assume that you are familiar with setting up a VISION:Transact application using IMS TM or CICS. In addition, you should be familiar with the standard process of installing a DB2 application under a terminal monitor.

The use of relational data bases requires a new technique for an application whether it be generated by VISION:Transact or COBOL. An application must be bound together with a plan that describes the relational tables and relational activity to DB2. In running a VISION:Transact job, we are concerned with two relational tasks.

The first of these is the APPGEN step, which, since it validates against the relational catalog, is a relational application itself. The plan for the VISION:Transact APPGEN is actually created as part of the VISION:Transact installation process at which time the BIND operation is also performed. The application developer only needs to be authorized to use the plan. All users must be authorized by the DB2 administrator to use the appropriate plan and relevant tables, otherwise DB2 will prohibit access to the tables. The VISION:Transact installation process will, therefore, set up the APPGEN step properly.

The second relational task is that of the application being developed. Here, we must concern ourselves with all of the steps necessary to install the application and this includes the BIND operation.

A conventional COBOL (PL/I, Assembly Language) application is developed in source code including imbedded SQL statements. The source code is passed through a pre-processor which removes the SQL statements placing them into a DBRM (Data Base Request Module) and replaces them with conventional Call statements for the language processor being used. After the pre-pass, the source is compiled into an object module which is then link edited with other modules and interface routines to become an executable application. This application must then be installed under the terminal monitor in the usual way. Meanwhile (and before application execution), the DBRM must be put through the BIND process to generate information for DB2. The standard IBM or installation procedures should be used for each of the above operations.

All of the above must be done for VISION:Transact applications. The pre-compilation and bind operation parallel a conventional Assembly language application.

When used with IMS/DB or VSAM under the IMS TM or CICS terminal monitors, the VISION:Transact APPGEN step generates an object module together with the terminal mapping module. The object module is link edited and installed under the terminal monitor using the standard IBM procedure. In this respect, the object module generated by VISION:Transact is no different from any object module generated by any of the compilers and VISION:Transact is no different from any of the conventional compilers.

When the application involves relational tables, the APPGEN step generates an additional assembly language source module as well as the usual object module and mapping tables. It is this additional module that must be pre-compiled, used to generate the DBRM, assembled, and (the DBRM) put through the bind operation. When the link edit is performed, the original VISION:Transact object module and the object module resulting from the pre-processed assembler module must be included.

There is nothing exceptional about installing a VISION:Transact relational application. A standard DB2 process is added to the usual VISION:Transact process as an extra step.

Here is a quick review of the steps involved. We have also added some reminders that should be checked as you reach these steps.

1. Execute VISION:Transact to provide the file definition for the DB2 tables involved, remembering to include the necessary entries for DB2 (i.e., the table name, column name, authorization ID (optional), and definition of the logical relationships between the segments or tables).
2. Define the screen definitions, which have no special considerations for DB2 tables.

3. Before running the APPGEN, you must have authorization to use the VISION:Transact generator plan (or be the original creator of the VISION:Transact generator plan, i.e., the person who did the VISION:Transact installation). Access to the tables may also have to be authorized depending on how the VISION:Transact FIVESQL module was installed (see the *VISION:Transact for CICS Installation Guide*).
4. Develop the VISION:Transact application remembering to include the authorization ID of the creator of the table or view on the RF statement if it is not included on the LS statement for that file and if you are not the creator of the table.
5. To run the APPGEN, you will execute the program IKJEFT01 which is a TSO utility that will invoke the TSO Attach facility so that there can be a connection to DB2 during the APPGEN. Note that the input to the APPGEN will consist of the M5INPUT and the control statements to the IKJEFT01 utility (SYSTSIN DD).
6. Output from the APPGEN will be the M5PREP file which contains the SQL statements used to create the application plan for DB2. The M5PREP file must be run through the pre-compile program and Assembler as part of the process of creating the DB2 plan. Following this, the object modules produced from the M5PREP assembly and the APPGEN must be link edited. There is an IBM provided PROC, DSNHASM, which will perform all three of these steps. Included in the PROC are three programs:
 - DSNHPC — The pre-compile.
 - ASMA90 — The assembly.
 - IEWL — The link edit.

You must provide to the PROC the user ID of the creator of the application plan and a member name. The member name will usually be the same as the program name/plan name (but that is not a requirement). The member name creates a member in a library called "user id.dbrmlib.data" so the user ID specified on the PROC should match the one on the dbrmlib or you will need to modify it.
7. Process the MFS or BMS statements output for the screen definitions to the M5PUNCH data set from the definition run and/or the M5MAP data set from the APPGEN run.
8. For IMS TM, add the APPLCTN and TRANSACT statements to the IMS system. For CICS, add the PPT, PCT, and RCT entries to the CICS tables.
9. Run the PSBGEN and ACBGEN using the recommended PSB. This is only required for CICS if the application uses IMS data bases.
10. The final step before executing is to execute the BIND of the new application plan to DB2.

Some Additional Hints

The following are some things which you should be aware of and watch out for. Some of these are installation dependent and may not apply to everyone:

1. Every time you perform an APPGEN you must also re-run the PROC to perform the pre-process, assemble, and link edit of the application. You must also perform a BIND REPLACE on the plan. This is because the assembled modules and the results of the bind are time-date stamped and checked by DB2.
2. Before running the APPGEN or using the application you must be granted authorization to use the VISION:Transact generator plan and the application plan. You may also have to be granted access to all relevant tables depending on how the FIVESQL module was installed at your site (see the *VISION:Transact for CICS Installation Guide*).
3. You should always provide an authorization ID on the LS or the RF statement for each of the tables or views (by now you will have noticed that we use authorization ID, creator ID, user ID, and owner ID interchangeably). The exception to this is when an unqualified table, view, or synonym name is used. In this case, no authorization ID is specified on the LS or RF statement. VISION:Transact will qualify the unqualified name with the authorization ID of the person who submitted the APPGEN run. DB2 will qualify the unqualified name with the authorization ID of the person who submitted the BIND run. If the IDs used are different, the table columns and data types must be the same.
4. For IMS TM, if your installation has multiple DB2 subsystem IDs (which are associated with specific MPP regions) then when defining the TRANSACT macro to IMS you will want to make sure you specify the right class in order to execute the application in the correct MPP region.
5. For IMS TM, when executing in BTS it may be necessary to include a "/P " statement in your BTS input in order to specify your DB2 subsystem ID if it is not the same as the default ID. (/P MBR=BTSCOM00 PA=000C14 PC=???)
6. For CICS, when an RCT entry is added to the CICS table, be sure the RCT specifies the correct authorization ID (AUTH parameter) to be used by CICS and DB2. This can be the CICS operator ID, user ID, terminal ID, or transaction code.
7. When you are performing an insert segment action on lower level segments, it is usually desirable to define the foreign key (from the LR statement) on an L0 statement. If this field is not provided with data from the screen, it will be necessary to fill the value of this field in a request. For inserts, all NOT NULL columns with no default value as well as those columns to be included on the insert must be defined and used as a key or result field in the application.
8. When performing updates, each field (column) to be updated must be defined and used as a result field in the application.

9. The application plan name is the same name as the program name. In IMS™, it will also be the same as the PSB name.
10. If you are not specified as the creator of the application plan when the bind is performed, then you must be granted authorization to access that application plan before executing the application.

Performance Considerations

The bulk of the resources that are consumed in a VISION:Transact relational application are consumed by the data base manager, DB2 or SQL/DS. It is appropriate to be aware that VISION:Transact generates applications that utilize imbedded SQL statements. VISION:Transact will generate statements based upon the specifications provided in the file definition and the fields referenced within the application. It should be noted that the Suppress Duplication and the Segment Order specifications on the LS statement, while only one character and easy to invoke, cause VISION:Transact to generate SELECT DISTINCT or ORDER clauses which require DB2 to perform extra processing and sort the results of the LR selection criteria. For screen synchronized data bases, this is generally a small manageable amount of data and therefore not significant. However, it is possible to create output screens or dynamically synchronized data bases that contain many repeated segments (rows). If the number is excessive, the processing time will increase.

VISION:Transact does not perform expensive SQL joins. Since screen synchronization effectively limits the retrieval to a handful of rows, VISION:Transact issues tailored SELECTs. The correct combination of SELECTs within a logical record has the same effect as if an outer join were performed. The logical record makes the capability of defining outer joins a trivial process.

VISION:Transact only retrieves or maps those fields (columns) into the logical record that are referenced or actually used in the application.

- For update operations, only those fields used as result fields are included in the update.
- For insert operations, only those fields defined as keys or used as result fields are included in the insert.

A field is used as a result field if it is:

- Named as Operand C of a PR statement in a request,
- Used as a result field of a LET statement in a procedure, or
- Specified as the destination of a non-null field action in the screen definition.

The definition of indexes is external to VISION:Transact. DB2 (or SQL/DS) will always honor a request for data whether or not an index exists. If the index does not exist, an entire data base scan will be performed. This can be very costly. You are advised to review whether foreign keys (i.e., fields referenced on the left hand side of LR statements) should be indexed. Fields defined as VISION:Transact key fields are also candidates for an index. These decisions must be made in the light of the application and the nature of the tables involved. Too many indexes will impact the application in a negative way as they are updated.

The browse capability is supported for relational data bases. However, it is implemented with different internal logic than that used for IMS, although the results to the application appear to be the same. In IMS, a data base call is made per segment retrieved and the retrieved segment is tested against the browse limits or the availability of a slot on the screen. In this manner, only the minimum number of segments is retrieved from the data base. With relational data bases, the browse capability is implemented by generating a single SQL statement with the limits of the browse incorporated into the SQL statement. This has two performance implications:

- First, if there is no index on the table, an entire data base scan will be performed to execute the request for data. This will happen regardless of the bounds placed on the browse. This can be very expensive. It is recommended that an index be created for the field against which the browse is taking place.
- Second, the consequence of SQL is that DB2 (SQL/DS) will return all qualifying rows in the range regardless of the availability of display on the screen. You cannot stop SQL once a request is made. It is therefore incumbent on the application to make sure that the upper bound of a browse is specified as accurately as possible. Careless specification of the upper bound of a browse will result in wasted activity by DB2.

Commits and Rollbacks

VISION:Transact does not generate any COMMITs to the data base. An automatic COMMIT is made by the terminal monitor at the end of the transaction. If the transaction experiences an abnormal system termination, the data will not be committed and the specific interaction will not have modified the data base.

In the event the error cycle is invoked, the input screen is redisplayed and by default data base updates are backed out. An IMS or CICS rollback is issued which causes DB2 to perform a backout of any updates that occurred within the interaction.

Note: An ROLB call is issued under IMS TM and a SYNCPOINT ROLLBACK command is issued under CICS.

Creating a Logical Record

In [Chapter 14, “A Conceptual Overview \(DB2 UDB\)”](#), we described the process of building a logical record and showed several examples in [Chapter 16, “Details of Operation and Examples \(DB2 UDB\)”](#) and [Chapter 17, “Updating and Designing for Referential Integrity \(DB2 UDB\)”](#). It is important to remember that if more than one row of a table qualifies to be a root segment, (e.g. the key provided from the screen is duplicated in the table), only one of the candidate rows will be selected. Within the concept of the logical record, only a single occurrence of a root segment is permitted.

Validation Against the DB2 Catalog

No validation against the DB2 (SQL/DS) catalog takes place during the VISION:Transact file definition run. Thus, during file definition, DB2 does not need to be available. Relational table, view, and synonym names are validated against the catalog during the APPGEN step. In this step, DB2 or SQL/DS must be available and the tables, views, or synonyms must have been previously CREATED with all columns defined.

Uniqueness of Tables

Normal operation against relational tables will permit duplicate rows to be entered into the table (by perhaps the INSERT action). You can prevent duplicate rows by using the DB2 capability of creating a UNIQUE INDEX.

Null Fields

VISION:Transact does not support the NULL property on a field basis. Whenever a NULL field is mapped into a logical record, it will be converted to a blank or zero, as appropriate. The null property will be preserved for a field only if the field is not used as a result in the application. For inserts, if a field is not a key or result field, it will not be included in the insert and DB2 will assign it the null value if the column can be null.

Performance and the Generation of Output Screens From a Browse File

The VISION:Transact capability of browsing files enables the generation of several output screens which the user may browse utilizing the paging features of the terminal monitor (PA1 for IMS TM). However, the nature of relational tables compared to hierarchical files is such that frequently much more data is available in a table than would be available in a record or set of records. This is simply a practical consideration of the difference in the way that one tends to build and use relational data bases. The consequence of this is, that with large tables the number of output screens may overflow the storage allocated within IMS TM to hold the screen images. This will cause termination of the run. Obviously, the problem can be avoided by increasing the amount of screen buffer space. However, this may well not be the most desirable solution.

The generation of many screens implies a lot of data. It might well be that the desired results are located within the first few screens. In which case, the data base accesses and the generation of the remaining screens is a waste of resources. Thus, a solution which makes better use of the resources may be more appropriate.

Effectively, this means achieving the results by the use of several transactions, each viewing a portion of the output data. It should be noted that DB2 operates differently from IMS/DB:

- Access of the data from IMS is on a segment by segment basis and can be stopped when enough data has been retrieved.
- Access of the data from a DB2 table is on a set basis by means of one SELECT statement. DB2 will retrieve all of the qualifying rows from the data base regardless of how many you want. Thus, on a very large data base, perhaps 1,000,000 rows would be processed just to put 10 onto a screen. This is a characteristic of DB2; once you start a SELECT you can't stop it. Since each transaction will trigger a SELECT, you must compromise between the number of pages generated for output and the total number of transactions. Because of the characteristic of the SELECT, it pays to generate a reasonable number of pages of output and minimize the number of transactions to view all of the output. Only you will know what a reasonable number of output pages is since it is a function of the data base and the probability of finding the data before viewing all of the screens.

The OUTLIM parameter on the SC statement enables you to control the number of pages generated. However, if OUTLIM is used and some pages are to be generated by a subsequent interaction, logic must be included in your application to set the lower limit of the browse capability (sync field 1) so that subsequent transactions will start the output screen displays with the right data.

A summary of the technique is as follows:

1. Establish a browse file.

2. Set sync field 1 low (below first record).
3. Set sync field 2 high (adequate to retrieve the appropriate number of rows).
4. Set OUTLIM for the desired number of screens.
5. Include a type 1 request or procedure to retrieve the last field displayed on the screen (i.e., the previous interaction) and set it as sync field 1 for the next interaction. You may also have to have a type O request or procedure to position the last key at a fixed position on the screen. This is a common technique with VISION:Transact and both fields are often used to communicate data between interactions via the screen.

Each interaction will issue a SELECT to the table governed by a "BETWEEN sync field 1 and sync field 2." Sync field 1 can be determined from the previous interaction. There is no practical method to determine sync field 2 and it is dependent on knowledge of the data base. You should employ index capabilities on the sync fields for optimum performance. Without an INDEX, DB2 will perform a full data base scan.

Tracing SQL Calls

SQL calls issued by your VISION:Transact generated application can be displayed during execution when the SSDB and/or DSDB I/O call tracing level is greater than 0. The SQL call trace displays the SQL function, return code, precompiler source statement number, and variables used in the WHERE clause, if any. Optionally, the input/output variables for the SQL call can be displayed. For more information on execution tracing, refer to [Chapter 12, "Execution Tracing"](#).

Order/Item Application Screen and Database Definition Code

This appendix contains the complete application system code for the ORDER/ITEM application system introduced in [Chapter 2, “A VISION:Transact Generated Application System”](#). The code has been compiled using VISION:Transact and has been executed online. The application system is coded in a variety of styles to demonstrate the flexibility of implementing applications with VISION:Transact and to show you alternative methods of performing application functions with VISION:Transact. For instance, most of screen control in this application is driven by the use of function keys. This is neither the primary nor a necessarily recommended use of function key support in VISION:Transact; nor is this the only way to control the selection of output screen displays in a VISION:Transact generated application. Therefore, we have shown other uses of function keys (to control dynamic selection of actions on many of the screens), and another method to implement your screen control (the ITMBRWS4 screen requests use temporary fields to drive the selection of screens output from interactions when ITMBRWS4 is the input screen). Most of these seemingly inconsistent coding techniques have been used to give you a feel for the breadth and power of implementation design and coding available to you when you use VISION:Transact. This section is intended more as a VISION:Transact sampler than as an application design guide.

The division of Appendix B into three sections denotes the three steps involved in generating the application system. First, we defined the data bases; next we defined the screens; and lastly, we performed the application generations. [Appendix B, “Order/Item Application Code Design Manual”](#) contains an example of a Design Manual. This Design Manual applies to the multiple screen transaction, M501. The Design Manual consists of a number of field, screen, and data base cross references for all screens referenced in that transaction. To get a complete Design Manual, like the one you see in the back of the application generation section, combine these listings with the glossary and request listings of all the screens referenced in the transaction. See [Appendix B, “Order/Item Application Code Design Manual”](#) and the *Advantage VISION:Transact Reference Summary* for more information on the Design Manual.

[Figure A-1](#) is a more detailed application system overview than the one presented in [Chapter 2, “A VISION:Transact Generated Application System”](#). It includes the screen control for the application and shows how the screen requests fit into the application flow. The screen requests are listed in the order in which they are executed when a screen display is output to, or input from, the terminal. The character to the left of each screen request name is its request type.

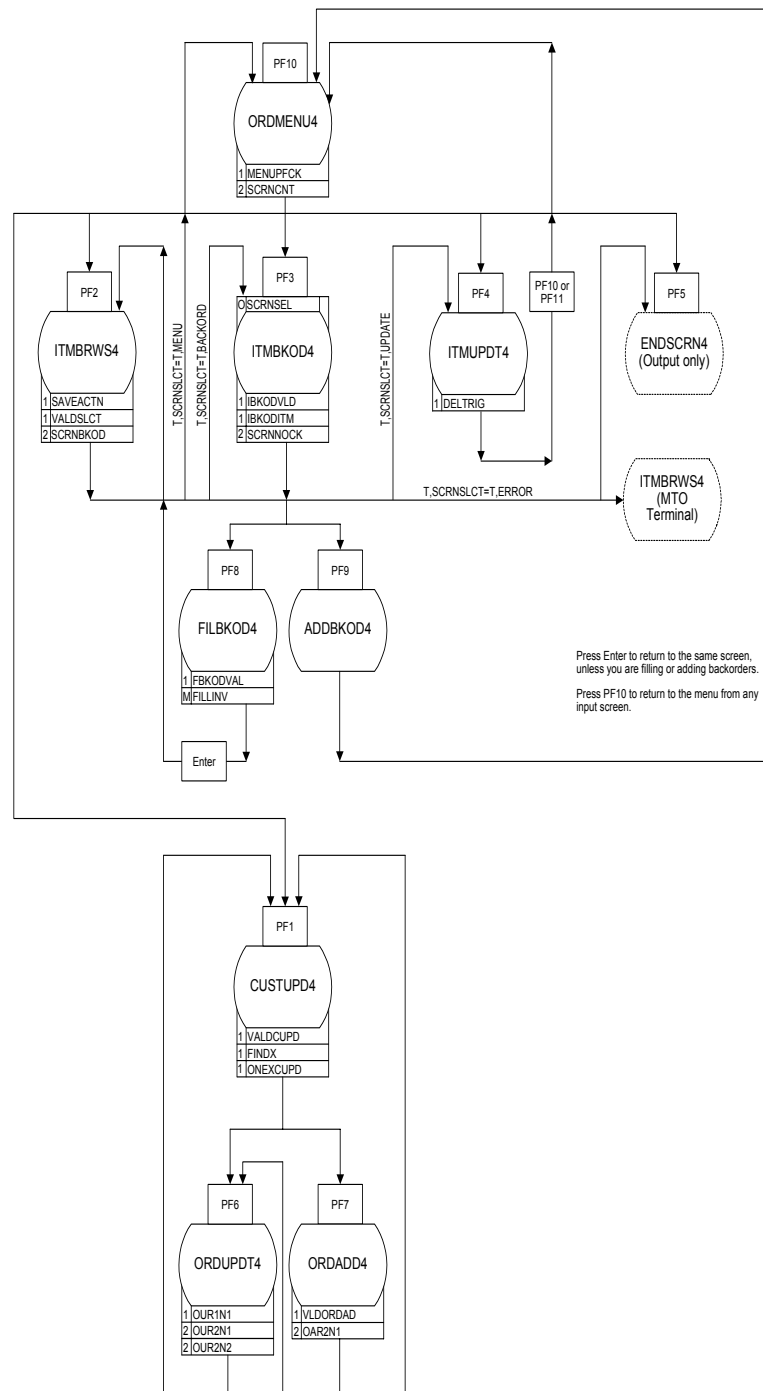


Figure A-1 Order/Item Application Flow

Program Listings for Sample Applications

Figure A-2 shows the program listings for sample applications.

```

DEC 04, 2003   19.04.12
*****
* Advantage VISION:Transact for z/OS - Version 7.8 *
*          *
* (C) 2003 Computer Associates International, Inc. (CA) *
*****

M5 RELEASE 7.8, 12/04/2003                                         Build Number = 103338-14.18.32.9

Advantage VISION:Transact APARs and PTFs Applied

NONE.

Common Library Subsystem Information

COMLIB Version C4.5      LIBRARY ACCESS METHOD = B/V      RESERVE = NO      ITEM TRACKING = 0

COMLIB System Modifications Applied

201,202,203,204,205,206,207,208,209,210,212,213,214,215,216,218
=====
=====
=
=
=
=
= M5PARM INSTALLATION PARAMETERS (FIVEPARM)
=
=
= ASTORSZ-TYPE A STORAGE SIZE.: 262,144      TEMON -DEFAULT TP MONITOR...
= CONFREQ-CONNECTOR REQUIRED?.. N             TEREQ -TF SIMTS. REQUIRED?.. N             LIBOPNA-LIB UPDT/READ IN APG: U
= DELIM -SYSTEM DELIMITER..... #            DEOCHAR-DECIMAL POINT CHAR... :           GRECHAR-NUMERIC GROUPING CHR: ,
= INVALID-INVALID FIELD CHAR.. *             NOEXIST-MISSING FIELD CHAR... -           OVERFLW-OVERFLOWED FLD CHAR.: +
= LEFTMRK-LEFT SEPARATOR..... (             SNGLSEP-SINGLE SEPARATOR..... ,           RIGHIMK-RIGHT SEPARATOR..... )
= LSTHGHT-MSLIST LINES/PAGE.... 66           LSTWDTH-M5LIST WIDH..... 132           PRICL -M5LIST LINE SPACING.: 1
= FLDSGN-FIELD DESIG CHAR.....              HILDQSN-HIGHLIGHT DESIG CHAR.: @           ERDEST-ERROR DESTINATION....
= LKEDCIS-CICS LKEDCALL PGMS?: N             CICKEY-CICS CLR/PA HANDLING: N           XLOAD -IMS EXECUTOR LOADED?: Y
= LKEDIMS-IMS LKEDCALL PGMS?: Y             MSGPROC-IMS MSG PROC OPTION: M
= TODAY -TODAY FLAG FORMAT...: MDDYY        DATEMT -DATE FORMAT.....: MM DD, YYYY
= MNHTAB-MONTH ABBREVIATIONS.: JANFEBMARAPRMAJUNJULAGSEPOCTNOVDEC
=====
=====

DEVICE TABLE -- GENERATED BY THE M5TERMDV MACRO IN M5PARM

<===== PARAMETERS FOR IMS/DC MFS GENERATION =====>
DEVICE-ID HEIGHT WIDTH MFS DEVICE TYPE NUMPK XCOLOR? XHLI? XPSYM? MODPAGE? MODIGN? MIDIGN? MIDNMOD?
D3270 24 80 (3270,2) 24 N N N Y Y Y N
DP24X80 24 80 (3270,2) 24 N N N Y Y Y N
DP24X80 55 120 (3270P,2) 0 N N N Y Y Y N
D24X80 24 80 (3270,2) 24 N N N Y Y Y N
D24X80 24 80 3270-A2 24 Y Y Y Y Y Y N
D24X80 27 132 3270-A7 24 N N N Y Y Y N
D24X80 43 80 3270-A4 24 N N N Y Y Y N
D24X80 24 80 (3270P,2) 0 N N N Y Y Y N
P24X80 55 132 SC51 0 N N N Y Y Y N
P24X80 55 80 SC52 0 N N N Y Y Y N
P55X120 55 120 (3270P,2) 0 N N N Y Y Y N
P55X120 55 132 SC51 0 Y Y Y N Y Y N

DEFAULT TERMINAL DEVICE-ID: D3270

-----
RUN STMT LIB SYS DEBUG SRC LKED SYS CICS MSG POS RUN TRG TRG PSB PROGRAM SEQUENCE
NAME TYPE OPEN DELIM LEVEL LST PGMS ROLLB KEYS PROC PCB TYPE MCN SYS NAME NAME 73 TO 80
-----
(FILEDEF ,RC) (#) (D) (C)

** CLSAA00 TYPE 0 DEFINITION CUSTOMER IS BEING PROCESSED.
```

FILE NAME	STMT TYPE	FILE IDENT	DELETE FILE	GLOSSARY LEVEL	RECORD FORMAT	RECORD SIZE	RECORDS PER BLK	BUFFER SIZE	MAPPED FILE	MAPPING INIT	MAPPING TERM	MAPPED RECSIZ	SEQUENCE 73 TO 80	LIST SEQ
(CUSTOMER)	(FD)	(CUSTOMDB)	(R)	(X)	(D)			(5000)						

FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMR	VRT SEG	SUP DPL	SEG KEY	KEY CHAR	AUTH ID	EXTERNAL TABLE NAME	SEQUENCE 73 TO 80	LIST SEQ
(CUSTOMER)	(LS)	(CUSTOMER)			(1)							

FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC	CNT SEG	OCRS NTIM	OUTP EDIT	EDIT LNTH	DATA BASE NAME/ COLUMN HEADING	SEQUENCE 73 TO 80	LIST SEQ
(CUSTOMER)	(LO)	(CUSTNO)		(1)	(1)	(1)	(5)	(C)	(1)									
(CUSTOMER)	(LO)	(CUSTNAME)		(1)	(1)	(6)	(30)	(C)	(S)									
(CUSTOMER)	(LO)	(CUSTEH)		(1)	(1)	(36)	(10)	(C)										

FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMR	VRT SEG	SUP DPL	SEG KEY	KEY CHAR	AUTH ID	EXTERNAL TABLE NAME	SEQUENCE 73 TO 80	LIST SEQ
(CUSTOMER)	(LS)	(ORDER)			(2)							

FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC	CNT SEG	OCRS NTIM	OUTP EDIT	EDIT LNTH	DATA BASE NAME/ COLUMN HEADING	SEQUENCE 73 TO 80	LIST SEQ
(CUSTOMER)	(LO)	(ORDERNO)		(2)	(2)	(1)	(5)	(C)	(1)									
(CUSTOMER)	(LO)	(ORDRDATE)		(2)	(2)	(8)	(6)	(C)										
(CUSTOMER)	(LO)	(ORDPERSON)		(2)	(2)	(14)	(3)	(C)										
(CUSTOMER)	(LO)	(ORDPONUM)		(2)	(2)	(17)	(5)	(C)										
(CUSTOMER)	(LO)	(ORDDUAT)		(2)	(2)	(22)	(6)	(C)										
(CUSTOMER)	(LO)	(ORDINVG)		(2)	(2)	(28)	(1)	(C)										
(CUSTOMER)	(LO)	(ORDCOMPLT)		(2)	(2)	(29)	(1)	(C)										

FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMR	VRT SEG	SUP DPL	SEG KEY	KEY CHAR	AUTH ID	EXTERNAL TABLE NAME	SEQUENCE 73 TO 80	LIST SEQ
(CUSTOMER)	(LS)	(SHIPINV)			(3)							

FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC	CNT SEG	OCRS NTIM	OUTP EDIT	EDIT LNTH	DATA BASE NAME/ COLUMN HEADING	SEQUENCE 73 TO 80	LIST SEQ
(CUSTOMER)	(LO)	(SHIPNO)		(3)	(3)	(1)	(4)	(C)	(1)									
(CUSTOMER)	(LO)	(SHIPDATE)		(3)	(3)	(5)	(6)	(C)										
(CUSTOMER)	(LO)	(SHIPPER)		(3)	(3)	(11)	(3)	(C)										
(CUSTOMER)	(LO)	(FRTCOST)		(3)	(3)	(23)	(4)	(P)										
(CUSTOMER)	(LO)	(INVNO)		(3)	(3)	(41)	(5)	(C)										
(CUSTOMER)	(LO)	(INVDATE)		(3)	(3)	(48)	(6)	(C)										
(CUSTOMER)	(LO)	(INVT)		(3)	(3)	(54)	(4)	(P)										
(CUSTOMER)	(LO)	(PAYDATE)		(3)	(3)	(70)	(6)	(C)										
(CUSTOMER)	(LO)	(CHECKNO)		(3)	(3)	(76)	(7)	(C)										
(CUSTOMER)	(LO)	(PAIDDATE)		(3)	(3)	(83)	(6)	(C)										

FILE NAME	STMT TYPE	SEGMENT NAME	DEL ETE	SGMT NMR	VRT SEG	SUP DPL	SEG KEY	KEY CHAR	AUTH ID	EXTERNAL TABLE NAME	SEQUENCE 73 TO 80	LIST SEQ
(CUSTOMER)	(LS)	(ITEMSHIP)			(4)							

FILE NAME	STMT TYPE	FIELD NAME	DEL ETE	SGMT NMR	LEV NUM	FIELD LOCATN	FIELD LNTH	FLD TYP	SEG KEY	FLD RND	DEC PLC	CNT SEG	OCRS NTIM	OUTP EDIT	EDIT LNTH	DATA BASE NAME/ COLUMN HEADING	SEQUENCE 73 TO 80	LIST SEQ
(CUSTOMER)	(LO)	(ITEMSHIP)		(4)	(4)	(1)	(7)	(C)	(1)									
(CUSTOMER)	(LO)	(QTYSHIP)		(4)	(4)	(8)	(7)	(Z)										

Figure A-2 Program Listings for Sample Applications (Page 2 of 84)

```

FILE STMT SEGMENT DEL SGMT VRT SUP SEG KEY AUTH EXTERNAL SEQUENCE LIST
NAME TYPE NAME ETE NMR SEG DPL KEY CHAR ID TABLE NAME 73 TO 80 SEQ
-----
(CUSTOMER) (LS) (ITEMORD ) ( 5)
-----
FILE STMT FIELD DEL SGMT LEV FIELD FIELD FLD SEG FLD DEC CNT OCRS OUTP EDIT DATA BASE NAME/ SEQUENCE LIST
NAME TYPE NAME ETE NMR NUM LOCATN INGTN TYP KEY RND PLC SEG NTIM EDIT INGH COLUMN HEADING 73 TO 80 SEQ
-----
(CUSTOMER) (LO) (ITEMORD ) ( 5) (3) ( 1) ( 7) (C) (1)
(CUSTOMER) (LO) (QTYBKORD) ( 5) (3) ( 21) ( 7) (Z)
(CUSTOMER) (LO) (ITMPRICE) ( 5) (3) ( 28) ( 6) (Z) (2)
(CUSTOMER) (LO) (ITMQTYOR) ( 5) (3) ( 34) ( 7) (Z)
-----
FILE STMT SEGMENT DEL SGMT VRT SUP SEG KEY AUTH EXTERNAL SEQUENCE LIST
NAME TYPE NAME ETE NMR SEG DPL KEY CHAR ID TABLE NAME 73 TO 80 SEQ
-----
(CUSTOMER) (LS) (INSTALL ) ( 6)
-----
FILE STMT FIELD DEL SGMT LEV FIELD FIELD FLD SEG FLD DEC CNT OCRS OUTP EDIT DATA BASE NAME/ SEQUENCE LIST
NAME TYPE NAME ETE NMR NUM LOCATN INGTN TYP KEY RND PLC SEG NTIM EDIT INGH COLUMN HEADING 73 TO 80 SEQ
-----
(CUSTOMER) (LO) (INSTNO ) ( 6) (2) ( 1) ( 4) (C) (1)
(CUSTOMER) (LO) (INSTCONT) ( 6) (2) ( 5) ( 3) (C)
(CUSTOMER) (LO) (INSTOPS ) ( 6) (2) ( 8) ( 8) (C)
(CUSTOMER) (LO) (INSTOSRL) ( 6) (2) ( 16) ( 2) (C)
(CUSTOMER) (LO) (INPRODID) ( 6) (2) ( 18) ( 2) (C)
(CUSTOMER) (LO) (INPRODRL) ( 6) (2) ( 20) ( 2) (C)
(CUSTOMER) (LO) (ADDRESS ) ( 6) (2) ( 22) ( 50) (C)
(CUSTOMER) (LO) (ADDLIN1 ) ( 6) (2) ( 22) ( 25) (C)
(CUSTOMER) (LO) (ADDLIN2 ) ( 6) (2) ( 47) ( 25) (C)
** CLSUN0 TYPE 1 SEGMENT 6 LEVEL 2 CONTAINS OVERLAPPING FIELDS.
ADDRESS OVERLAPS ADDLIN1 ADDLIN2
** CLSDC02 TYPE 0 THE DEFINITION IS VALID AND HAS BEEN CATALOGED.

FILE STRUCTURE DIAGRAM FOR
FILE DEFINITION - CUSTOMER

RECORD FORMAT = DL/I
BUFFER SIZE = 5000

DIAGRAM PAGE 1
FILE IDENTIFICATION = CUSTOMDB
NUMBER OF SEGMENTS IN FILE = 6

      +-----+
      °CUSTOMER °
      +-----+
      °
      +-----+
      °
      +-----+ +-----+
      °ORDER ° °INSTALL °
      +-----+ +-----+
      °
      +-----+
      °
      +-----+ +-----+
      °SHIPIN ° °ITEMORD °
      +-----+ +-----+
      °
      °
      °
      +-----+
      °ITEMSHIP °
      +-----+

*****
* File Definition for File: CUSTOMER *
* Glossary by Location *
*****

Format: DL/I Buffer Size: 5000 DBD Name: CUSTOMDB Segments Defined: 6
***** Number: 01 Level: 1 Order: UNORDERED Fields in Segment: 3
* Segment Name: CUSTOMER * Key Fields: CUSTNO - UNIQUE KEYS ASSUMED
***** Segment Size: 45
CUSTNO
----- CHARACTER Loc. 1 Length: 5 Dec. Places: 0 Col. Width: 5
Special Field Attributes
-----
```

Figure A-2 Program Listings for Sample Applications (Page 3 of 84)


```

This is Key Field 1
This is a Data Base Search Field
----- CUSTNAME CHARACTER Loc. 6 Length: 30 Dec. Places: 0 Col. Width: 30
Special Field Attributes
-----
This is a Data Base Search Field
----- CUSTPH CHARACTER Loc. 36 Length: 10 Dec. Places: 0 Col. Width: 10

*****
* File Definition for File: CUSTOMER *
* Glossary by Location *
*****
Glossary Page 2

***** Number: 02 Level: 2 Order: UNORDERED Fields in Segment: 7
* Segment Name: ORDER * Key Fields: ORDERNO - UNIQUE KEYS ASSUMED
***** Parent Segment: CUSTOMER (Number=01) Count Field: **NONE** Segment Size: 29
ORDERNO
----- CHARACTER Loc. 1 Length: 5 Dec. Places: 0 Col. Width: 5
Special Field Attributes
-----
This is Key Field 1
This is a Data Base Search Field
----- ORDDATE CHARACTER Loc. 8 Length: 6 Dec. Places: 0 Col. Width: 6
ORPERSON CHARACTER Loc. 14 Length: 3 Dec. Places: 0 Col. Width: 3
ORDPONUM CHARACTER Loc. 17 Length: 5 Dec. Places: 0 Col. Width: 5
ORDDUAT CHARACTER Loc. 22 Length: 6 Dec. Places: 0 Col. Width: 6
ORDINVCN CHARACTER Loc. 28 Length: 1 Dec. Places: 0 Col. Width: 1
ORDCOMPLT CHARACTER Loc. 29 Length: 1 Dec. Places: 0 Col. Width: 1

*****
* File Definition for File: CUSTOMER *
* Glossary by Location *
*****
Glossary Page 3

***** Number: 03 Level: 3 Order: UNORDERED Fields in Segment: 10
* Segment Name: SHIPINV * Key Fields: SHIPNO - UNIQUE KEYS ASSUMED
***** Parent Segment: INSTALL (Number=06) Count Field: **NONE** Segment Size: 88
SHIPNO
----- CHARACTER Loc. 1 Length: 4 Dec. Places: 0 Col. Width: 4
Special Field Attributes
-----
This is Key Field 1
This is a Data Base Search Field
----- SHIPDATE CHARACTER Loc. 5 Length: 6 Dec. Places: 0 Col. Width: 6
SHIPPER CHARACTER Loc. 11 Length: 3 Dec. Places: 0 Col. Width: 3
FRTOCST PACKED Loc. 23 Length: 4 Dec. Places: 2 Col. Width: 10 Pattern: S99,999.99
INVNO CHARACTER Loc. 41 Length: 5 Dec. Places: 0 Col. Width: 5
INVDATE CHARACTER Loc. 48 Length: 6 Dec. Places: 0 Col. Width: 6
INPMT PACKED Loc. 54 Length: 4 Dec. Places: 2 Col. Width: 10 Pattern: S99,999.99
PAYDATE CHARACTER Loc. 70 Length: 6 Dec. Places: 0 Col. Width: 6
CHECKNO CHARACTER Loc. 76 Length: 7 Dec. Places: 0 Col. Width: 7
PAIDDATE CHARACTER Loc. 83 Length: 6 Dec. Places: 0 Col. Width: 6

*****
* File Definition for File: CUSTOMER *
* Glossary by Location *
*****
Glossary Page 4

***** Number: 04 Level: 4 Order: UNORDERED Fields in Segment: 2
* Segment Name: ITEMSHIP * Key Fields: ITEMSHIP - UNIQUE KEYS ASSUMED
***** Parent Segment: ITEMORD (Number=05) Count Field: **NONE** Segment Size: 14
ITEMSHIP

```

Figure A-2 Program Listings for Sample Applications (Page 4 of 84)

```

----- CHARACTER Loc. 1 Length: 7 Dec. Places: 0 Col. Width: 7
Special Field Attributes
-----
This is Key Field 1
This is a Data Base Search Field

QTYSHIP
----- ZONED Loc. 8 Length: 7 Dec. Places: 0 Col. Width: 10 Pattern: S9,999,999

*****
* File Definition for File: CUSTOMER *
* Glossary by Location *
*****
Glossary Page 5

***** Number: 05 Level: 3 Order: UNORDERED Fields in Segment: 4
* Segment Name: ITEMORD * Key Fields: ITEMORD - UNIQUE KEYS ASSUMED
***** Parent Segment: INSTALL (Number=06) Count Field: **NONE** Segment Size: 40
ITEMORD
----- CHARACTER Loc. 1 Length: 7 Dec. Places: 0 Col. Width: 7
Special Field Attributes
-----
This is Key Field 1
This is a Data Base Search Field

QTYBKORD
----- ZONED Loc. 21 Length: 7 Dec. Places: 0 Col. Width: 10 Pattern: S9,999,999
ITMPRICE
----- ZONED Loc. 28 Length: 6 Dec. Places: 2 Col. Width: 9 Pattern: S9,999.99
ITMQTYOR
----- ZONED Loc. 34 Length: 7 Dec. Places: 0 Col. Width: 10 Pattern: S9,999,999

*****
* File Definition for File: CUSTOMER *
* Glossary by Location *
*****
Glossary Page 6

***** Number: 06 Level: 2 Order: UNORDERED Fields in Segment: 9
* Segment Name: INSTALL * Key Fields: INSTNO - UNIQUE KEYS ASSUMED
***** Parent Segment: CUSTOMER (Number=01) Count Field: **NONE** Segment Size: 71
INSTNO
----- CHARACTER Loc. 1 Length: 4 Dec. Places: 0 Col. Width: 4
Special Field Attributes
-----
This is Key Field 1
This is a Data Base Search Field

INSTOCONT
----- CHARACTER Loc. 5 Length: 3 Dec. Places: 0 Col. Width: 3
INSTOPS
----- CHARACTER Loc. 8 Length: 8 Dec. Places: 0 Col. Width: 8
INSTOSRL
----- CHARACTER Loc. 16 Length: 2 Dec. Places: 0 Col. Width: 2
INPRODID
----- CHARACTER Loc. 18 Length: 2 Dec. Places: 0 Col. Width: 2
INPRODR1
----- CHARACTER Loc. 20 Length: 2 Dec. Places: 0 Col. Width: 2
ADDRESS
----- CHARACTER Loc. 22 Length: 50 Dec. Places: 0 Col. Width: 50
ADDLIN1
----- CHARACTER Loc. 22 Length: 25 Dec. Places: 0 Col. Width: 25
ADDLIN2
----- CHARACTER Loc. 47 Length: 25 Dec. Places: 0 Col. Width: 25
** CLSCB00 TYPE 0 END OF FILE DEFINITION GLOSSARY.

** CLSAA00 TYPE 0 DEFINITION ITEMX IS BEING PROCESSED.

-----
FILE STMT FILE DELETE GLOSSARY RECORD RECORD RECORDS BUFFER MAPPED MAPPING MAPPED SEQUENCE LIST
NAME TYPE IDENT FILE LEVEL FORMAT SIZE PER BLK SIZE FILE INIT TERM RECSIZ 73 TO 80 SEQ
-----
(ITEMX ) (FD) (ITEMDB ) (R) (X) (D) ( 2K)

-----
FILE STMT SEGMENT DEL SGMT VRT SUP SEG KEY AUTH EXTERNAL SEQUENCE LIST
NAME TYPE NAME ETE NMR SEG DPL KEY CHAR ID TABLE NAME 73 TO 80 SEQ
-----
(ITEMX ) (LS) (ITEMROOT) ( 1)

-----
FILE STMT FIELD DEL SGMT LEV FIELD FIELD FLD SEG FLD DEC CNT OCRS OUTP EDIT DATA BASE NAME/ SEQUENCE LIST

```

Figure A-2 Program Listings for Sample Applications (Page 5 of 84)

```
NAME      TYPE      NAME      ETE NMR NUM LOCATN LNTH TYP KEY RND PLC SEG NTIM EDIT LNTH      COLUMN HEADING      73 TO 80      SEQ
-----
(ITEMX   ) (LO) (ITEMNO )      ( 1) (1) ( 3) ( 7) (C) (1)
(ITEMX   ) (LO) (ITMPRICE)      ( 1) (1) ( 10) ( 6) (Z)
(ITEMX   ) (LO) (PACKBOX )      ( 1) (1) ( 16) ( 3) (Z)
(ITEMX   ) (LO) (QANONHND)      ( 1) (1) ( 19) ( 5) (Z)
(ITEMX   ) (LO) (SHIPYTD )      ( 1) (1) ( 24) ( 6) (Z)
(ITEMX   ) (LO) (ORDYTD )      ( 1) (1) ( 30) ( 6) (Z)
(ITEMX   ) (LO) (QTYRKORD)      ( 1) (1) ( 36) ( 5) (Z)
(ITEMX   ) (LO) (REORDPT )      ( 1) (1) ( 41) ( 4) (Z)
(ITEMX   ) (LO) (PRODCODE)      ( 1) (1) ( 45) ( 2) (C)
(ITEMX   ) (LO) (ITEMNAME)      ( 1) (1) ( 47) ( 20) (C)
```

```
FILE STMT SEGMENT DEL SGMT VRT      SUP SEG KEY      AUTH      EXTERNAL      SEQUENCE      LIST
NAME TYPE NAME ETE NMR SEG      DPL KEY CHAR      ID      TABLE NAME      73 TO 80      SEQ
```

```
(ITEMX   ) (LS) (EKORDERS)      ( 2)
```

```
FILE STMT FIELD DEL SGMT LEV FIELD FIELD FLD SEG FLD DEC CNT OORS OUTP EDIT      DATA BASE NAME/      SEQUENCE      LIST
NAME TYPE NAME ETE NMR NUM LOCATN LNTH TYP KEY RND PLC SEG NTIM EDIT LNTH      COLUMN HEADING      73 TO 80      SEQ
```

```
(ITEMX   ) (LO) (EKORDRNO)      ( 2) (2) ( 3) ( 7) (C) (1)
(ITEMX   ) (LO) (CUSTNO )      ( 2) (2) ( 10) ( 5) (C)
(ITEMX   ) (LO) (ORDRNO )      ( 2) (2) ( 17) ( 5) (C)
```

** CLSDC02 TYPE 0 THE DEFINITION IS VALID AND HAS BEEN CATALOGED.

FILE STRUCTURE DIAGRAM FOR
FILE DEFINITION - ITEMX

DIAGRAM PAGE 1

RECORD FORMAT = DL/I
BUFFER SIZE = 2K

FILE IDENTIFICATION = ITEMDB
NUMBER OF SEGMENTS IN FILE = 2

```
+-----+
°ITEMROOT °
+-----+
  °
  °
  °
+-----+
°EKORDERS °
+-----+
```

* File Definition for File: ITEMX *
* Glossary by Location *

Glossary Page 1

Format: DL/I Buffer Size: 2048

DBD Name: ITEMDB

Segments Defined: 2

***** Number: 01 Level: 1 Order: UNORDERED
* Segment Name: ITEMROOT * Key Fields: ITEMNO - UNIQUE KEYS ASSUMED

Fields in Segment: 10

Segment Size: 66

```
ITEMNO
----- CHARACTER Loc. 3 Length: 7 Dec. Places: 0 Col. Width: 7
```

Special Field Attributes

This is Key Field 1
This is a Data Base Search Field

```
ITMPRICE
----- ZONED Loc. 10 Length: 6 Dec. Places: 0 Col. Width: 8 Pattern: S999,999
```

```
PACKBOX
----- ZONED Loc. 16 Length: 3 Dec. Places: 0 Col. Width: 4 Pattern: S999
```

```
QANONHND
----- ZONED Loc. 19 Length: 5 Dec. Places: 0 Col. Width: 7 Pattern: S99,999
```

```
SHIPYTD
----- ZONED Loc. 24 Length: 6 Dec. Places: 0 Col. Width: 8 Pattern: S999,999
```

```
ORDYTD
----- ZONED Loc. 30 Length: 6 Dec. Places: 0 Col. Width: 8 Pattern: S999,999
```

```
QTYRKORD
----- ZONED Loc. 36 Length: 5 Dec. Places: 0 Col. Width: 7 Pattern: S99,999
```

```
REORDPT
----- ZONED Loc. 41 Length: 4 Dec. Places: 0 Col. Width: 6 Pattern: S9,999
```

```
PRODCODE
----- CHARACTER Loc. 45 Length: 2 Dec. Places: 0 Col. Width: 2
```

```
ITEMNAME
----- CHARACTER Loc. 47 Length: 20 Dec. Places: 0 Col. Width: 20
```

* File Definition for File: ITEMX *

Glossary Page 2

Figure A-2 Program Listings for Sample Applications (Page 6 of 84)

```

*          Glossary by Location          *
*****

***** Number: 02   Level: 2   Order: UNORDERED                      Fields in Segment:   3
* Segment Name: BKORDERS * Key Fields: BKORDRNO - UNIQUE KEYS ASSUMED
***** Parent Segment: ITEMROOT (Number=01)   Count Field: **NONE**      Segment Size:   21
BKORDRNO
----- CHARACTER Loc.   3   Length:   7   Dec. Places:   0       Col. Width:   7
Special Field Attributes
This is Key Field 1
This is a Data Base Search Field

CUSTNO
----- CHARACTER Loc.  10   Length:   5   Dec. Places:   0       Col. Width:   5
ORDERNO
----- CHARACTER Loc.  17   Length:   5   Dec. Places:   0       Col. Width:   5
** CLSGB00 TYPE 0 END OF FILE DEFINITION GLOSSARY.

** M5W204 TYPE 0 NUMBER OF MESSAGES PRINTED IS   8.
M5INPUT -   59 RECORDS INPUT
M5PUNCH -    0 RECORDS OUTPUT
M5LIST  -  429 RECORDS OUTPUT
5 TRACKS ASSIGNED TO M5LIB - 4 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 272.
** M5W209 TYPE 0 0 BYTES OF TYPE A STORAGE USED DURING DICTIONARY MAINTENANCE
DEC 04, 2003 20:55.00

*****
* Advantage VISION:Transact for z/OS - Version 7.8 *
*
* (C) 2003 Computer Associates International, Inc. (CA) *
*****

M5 RELEASE 7.8, 12/04/2003                                     Build Number = 103338-14.18.32.9

Advantage VISION:Transact APARs and PTFs Applied

NONE.

Common Library Subsystem Information

COMLIB Version C4.5 LIBRARY ACCESS METHOD = B/V RESERVE = NO ITEM TRACKING = 0

COMLIB System Modifications Applied

201,202,203,204,205,206,207,208,209,210,212,213,214,215,216,218

```

DEVICE TABLE -- GENERATED BY THE M5TERMDV MACRO IN M5PARM

DEVICE-ID	HEIGHT	WIDTH	PARAMETERS FOR IMS/DC MFS GENERATION								
			MFS DEVICE TYPE	NUMPFK	XCOLOR?	XHIL?	XPSYM?	MODPAGE?	MODIGN?	MIDIGN?	MIDMOD?
D3270	24	80	(3270,2)	24	N	N	N	Y	Y	Y	N
DP24X80	24	80	(3270,2)	24	N	N	N	Y	Y	Y	N
DP24X80	55	120	(3270P,2)	0	N	N	N	Y	Y	Y	N
D24X80	24	80	(3270,2)	24	N	N	N	Y	Y	Y	N
D24X80	24	80	3270-A2	24	Y	Y	Y	Y	Y	Y	N
D24X80	27	132	3270-A7	24	N	N	N	Y	Y	Y	N
D24X80	43	80	3270-A4	24	N	N	N	Y	Y	Y	N
P24X80	24	80	(3270P,2)	0	N	N	N	N	Y	Y	N
P24X80	55	132	SCS1	0	N	N	N	N	Y	Y	N
P24X80	55	80	SCS2	0	N	N	N	N	Y	Y	N
P55X120	55	120	(3270P,2)	0	N	N	N	N	Y	Y	N

Figure A-2 Program Listings for Sample Applications (Page 7 of 84)

P55X120 55 132 SCS1 0 Y Y Y N Y Y N

DEFAULT TERMINAL DEVICE-ID: D3270

RUN NAME	STMT TYPE	LIB OPEN	SYS DELIM	SRC LEVEL	DEBUG LST	SYS PGMS	LIKED ROLLS	SYS KEYS	CICS MSG PROC	POS PCB	RUN TYPE	TRG MON	TRG SYS	PSB NAME	PROGRAM NAME	SEQUENCE 73 TO 80

(SCRNDEF ,RC) (#) (D) (C)

** M5WC03 TYPE 0 SCREEN DEFINITION ORDMENU4 CATALOGED PRIOR TO VALIDATION.

SCREEN NAME	STMT TYPE	CAT ACIN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

(ORDMENU4) (SD) (R) () (e) (M510) (M510)

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	TEXT (1-57)	SEQUENCE 73 TO 80

(ORDMENU4,SL) (1) (0) (ORDMENU4)
(ORDMENU4,SL) (3) (0) (CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS:)
(ORDMENU4,SL) (5) (0) (PF1---CUSTOMER ADD/UPDATE)
(ORDMENU4,SL) (7) (0) (PF2---ITEM BROWSE BY PRODUCT KEY)
(ORDMENU4,SL) (9) (0) (PF3---BACK-ORDERED ITEM)
(ORDMENU4,SL) (11) (0) (PF4---ITEM ADD/UPDATE)
(ORDMENU4,SL) (13) (0) (PF5---END TRANSACTION)
(ORDMENU4,SL) (16) (0) (ENTER CUSTOMER OR ITEM NUMBER)
(ORDMENU4,SL) (18) (0) ((ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2))
(ORDMENU4,SL) (18) (1) (IS USED)

** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

(ORDMENU4,SS) (1) (1)

SCREEN NAME	STMT TYPE	SEG S	NO. A	V	SOURCE/DEST	IR	CE	PE	PM	N	A	L	C	OPERAND B	PARTIAL FIELD	SEQUENCE 73 TO 80

(ORDMENU4,S0) (1) (N,CUSTNO) (I) (X) (NM) (ITEMNO)
(ORDMENU4,S1) (V) (IF) (S,PFKEY) (EQ) (C,PF001)
(ORDMENU4,S0) (1) (S) (1,ITEMNO) (I) (X)
(ORDMENU4,S1) (M) (IF) (S,PFKEY) (EQ) (C,PF003)
(ORDMENU4,S1) (V) (IF) (S,PFKEY) (EQ) (C,PF004)
(ORDMENU4,S0) (1) (V) (I) (N) (NM) (PFKEY)
(ORDMENU4,S1) (IN) (5))
(ORDMENU4,S1) (EQ) (C,PF001)
(ORDMENU4,S1) (1,O,EQ) (C,PF002)
(ORDMENU4,S1) (1,O,EQ) (C,PF003)
(ORDMENU4,S1) (1,O,EQ) (C,PF004)
(ORDMENU4,S1) (1,O,EQ) (C,PF005)
(ORDMENU4,S0) (1) (A) (F,MSGLINE) (H) (*SYSTEM*)

GLOSSARY FOR SCREEN - ORDMENU4
SCREEN FORMAT
DEC 04, 2003
PAGE 1

-----1-----2-----3-----4-----5-----6-----7-----8		
1	ORDMENU4	1
2		2
3	CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS:	3
4		4
5	PF1---CUSTOMER ADD/UPDATE	5
6		6
7	PF2---ITEM BROWSE BY PRODUCT KEY	7
8		8
9	PF3---BACK-ORDERED ITEM	9
10		10

Figure A-2 Program Listings for Sample Applications (Page 8 of 84)

11 | PF4---ITEM ADD/UPDATE | 11
12 | | 12
13 | PF5---END TRANSACTION | 13
14 | | 14
15 | | 15
16 | ENTER CUSTOMER OR ITEM NUMBER | 16
17 | | 17
18 | (ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2 IS USED) | 18
19 | | 19
20 | | 20
21 | | 21
22 | | 22
23 | | 23
24 | | 24

-----1-----2-----3-----4-----5-----6-----7-----8
SCREEN IDENTIFIER.. M510 FIELD DESIGNATOR..... INITIAL CURSOR POSITION
TRANSACTION CODE... M510 HIGHLIGHT DESIGNATOR.. @ LINE: 16 COLUMN: 50
OUTPUT CONTROL..... S

GLOSSARY FOR SCREEN - ORDMENU4
SCREEN CONTENT

DEC 04, 2003
PAGE 1

DATE CATALOGED 12/04/03 20:55:00
CATALOGED BY

DATE LAST USED
EXPIRATION DATE

DATE LAST UPDATED 12/04/03 20:55:00
LAST UPDATED BY

* SEGMENT 1 *

NUMBER OF SEGMENT OCCURRENCES = 1
NUMBER OF FIELDS IN SEGMENT = 3

POSITION OF FIRST FIELD
LINE: 16 COLUMN: 50

SCREEN	I	R	E	P	M	N	A	C		
FIELD	.SOURCE/DEST.	/	E	H	R	R	O	U	C	O E/V
NAME	QLF FLD NAME	O	Q	I	R	T	D	M	T	N OP

ITEMNO	N,CUSTNO	I		H		N	N		X	
										NM
									V	IF
	1,ITEMNO	I		H		N	N		X	S,PFKEY
									M	IF
									V	IF
	LOC/XATTRS: LINE	16, COL	50	=(COLOR=G, HIGHLIGHT=, PGM SYMBOL=)						
PFKEY		I		H		N			N	
										NM
										IN
										EQ
									O	EQ
									O	EQ
									O	EQ
									O	EQ
									O	EQ

VIRTUAL SCREEN FIELD

NONAME F,MSGLINE O H
LOC/XATTRS: LINE 24, COL 2=(COLOR=N, HIGHLIGHT=, PGM SYMBOL=)

** MK5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** MK5WC03 TYPE 0 SCREEN DEFINITION CUSTUPD4 CATALOGED PRIOR TO VALIDATION.

SCREEN	SIMT	CAT	GLOSS	FIELD	HILITE	AUTO	OUTPUT	TRANSACT	SCREEN	PROC	TERMINAL	SEQUENCE
NAME	TYPE	ACTN	TYPE	DESIG	DESIG	FORMAT	CONTROL	CODE	ID	NULL	DEVICE	73 TO 80

(CUSTUPD4) (SD) (R) () (E) (M501) (CUSTSET)

SCREEN SIMT LINE CONT- EXTN 1.....2.....3.....4.....5..... <-TEXT (1-57) SEQUENCE

Figure A-2 Program Listings for Sample Applications (Page 9 of 84)

NAME TYPE NO. INUE ATTR ..6....+...7....+...8....+...9....+...0....+...1.... <-TEXT (58-114) 73 TO 80

```
(CUSTUPD4,SL) ( 1) (0) ( CUSTUPD4 )
(CUSTUPD4,SL) ( 3) (0) ( CUSTOMER NUMBER )
(CUSTUPD4,SL) ( 4) (0) ( CUSTOMER NAME )
(CUSTUPD4,SL) ( 5) (0) ( CUSTOMER PHONE )
(CUSTUPD4,SL) ( 7) (0) ( ACT ORDER P.O. DUE INVOICE )
(CUSTUPD4,SL) ( 7) (1) ( ORDER )
(CUSTUPD4,SL) ( 8) (0) ( ION NUMBER DATE SELLER NUMBER DATE GENERATED )
(CUSTUPD4,SL) ( 8) (1) ( COMPL )
(CUSTUPD4,SL) ( 9) (0) ( - )
(CUSTUPD4,SL) ( 9) (1) ( - )
(CUSTUPD4,SL) ( 11) (0) ( - )
(CUSTUPD4,SL) ( 11) (1) ( - )
(CUSTUPD4,SL) ( 13) (0) ( - )
(CUSTUPD4,SL) ( 13) (1) ( - )
(CUSTUPD4,SL) ( 15) (0) ( - )
(CUSTUPD4,SL) ( 15) (1) ( - )
(CUSTUPD4,SL) ( 17) (0) ( - )
(CUSTUPD4,SL) ( 17) (1) ( - )
(CUSTUPD4,SL) ( 19) (0) ( - )
(CUSTUPD4,SL) ( 19) (1) ( - )
(CUSTUPD4,SL) ( 21) (0) ( NEXT SCREEN OF ORDERS=PA1 UPDATE=U ACT + ENTER )
(CUSTUPD4,SL) ( 22) (0) ( ORDER DETAILS=X ACT + PF6 ADD ORDER=PF7 )
(CUSTUPD4,SL) ( 22) (1) ( RETURN TO MENU=PF10 )
```

** M5WML14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

(CUSTUPD4,SS) (1) (1)

SCREEN NAME	SIMT TYPE	SEG S	NO. A	V	IR	CE	PMN	A	LC	C	LO	E/V		OPERAND B.....	PARTIAL FIELD	SEQUENCE	
NAME	TYPE	NO.	A	QLF	FLD	NAME	OD	IR	RT	DM	T	V	N	OP	QLF	FIELD NAME OR CONSTANT	ST NO OP 73 TO 80

```
(CUSTUPD4,S0) ( 1) (N,CUSTNO ) (B) (X) (NM) (CUSTNO )
(CUSTUPD4,S1) (M) (IF) (F,ATNID ) (EQ) (C,PF006 )
(CUSTUPD4,S1) (M) (IF) (F,ATNID ) (EQ) (C,PF007 )
(CUSTUPD4,S1) (U) (IF) (F,ATNID ) (EQ) (C,ENTER )
(CUSTUPD4,S0) ( 1) (N,CUSTNAME) (B) (P) (NM) (CUSTNAME )
(CUSTUPD4,S0) ( 1) (N,CUSTPH ) (B) (P) (NM) (CUSTPH )
(CUSTUPD4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)
```

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

(CUSTUPD4,SS) (2) (5)

SCREEN NAME	SIMT TYPE	SEG S	NO. A	V	IR	CE	PMN	A	LC	C	LO	E/V		OPERAND B.....	PARTIAL FIELD	SEQUENCE	
NAME	TYPE	NO.	A	QLF	FLD	NAME	OD	IR	RT	DM	T	V	N	OP	QLF	FIELD NAME OR CONSTANT	ST NO OP 73 TO 80

```
(CUSTUPD4,S0) ( 2) (I) (N) (NM) (ACTION )
(CUSTUPD4,S0) ( 2) (N,ORDERNO ) (B) (X) (NM) (ORDERNO )
(CUSTUPD4,S1) (U) (IF) (S,ACTION ) (EQ) (C,U )
(CUSTUPD4,S1) (M) (IF) (S,ACTION ) (EQ) (C,X )
(CUSTUPD4,S0) ( 2) (N,ORDRDATE) (B) (P) (NM) (ORDRDATE )
(CUSTUPD4,S0) ( 2) (N,ORPERSON) (B) (P) (NM) (ORPERSON )
(CUSTUPD4,S0) ( 2) (N,ORDPONUM) (B) (P) (NM) (ORDPONUM )
(CUSTUPD4,S0) ( 2) (N,ORDDUTAT) (B) (P) (NM) (ORDDUTAT )
(CUSTUPD4,S0) ( 2) (N,ORDINVGNN) (B) (P) (NM) (ORDINVGNN )
(CUSTUPD4,S0) ( 2) (N,ORDCMPLT) (B) (P) (NM) (ORDCMPLT )
```

GLOSSARY FOR SCREEN - CUSTUPD4
SCREEN FORMAT

DEC 04, 2003
PAGE 1

```
*-----1-----2-----3-----4-----5-----6-----7-----8*
1 | CUSTUPD4 | 1
2 | | 2
3 | CUSTOMER NUMBER | 3
```

Figure A-2 Program Listings for Sample Applications (Page 10 of 84)

```

4 |      CUSTOMER NAME _____ | 4
5 |      CUSTOMER PHONE _____ | 5
6 |      _____ | 6
7 |      ACT ORDER _____ | 7
8 |      ION NUMBER DATE SELLER P.O. DUE INVOICE ORDER | 8
9 |      - - - - - - - - - - - - - - - - - - - - - - - | 9
10 |      _____ | 10
11 |      _____ | 11
12 |      _____ | 12
13 |      _____ | 13
14 |      _____ | 14
15 |      _____ | 15
16 |      _____ | 16
17 |      _____ | 17
18 |      _____ | 18
19 |      _____ | 19
20 |      _____ | 20
21 | NEXT SCREEN OF ORDERS=PA1      UPDATE=U ACT + ENTER | 21
22 | ORDER DETAILS=X ACT + PF6      ADD ORDER=PF7      RETURN TO MENU=PF10 | 22
23 |      _____ | 23
24 |      _____ | 24

*-----1-----2-----3-----4-----5-----6-----7-----8*

SCREEN IDENTIFIER.. M510      FIELD DESIGNATOR.....      INITIAL CURSOR POSITION
TRANSACTION CODE... M501      HIGHLIGHT DESIGNATOR.. @      LINE: 11 COLUMN: 10
                                OUTPUT CONTROL..... S

```

```

                                GLOSSARY FOR SCREEN - CUSTUPD4
                                SCREEN CONTENT
                                DEC 04, 2003
                                PAGE 1

DATE CATALOGED 12/04/03 20:55:00      DATE LAST USED      DATE LAST UPDATED 12/04/03 20:55:00
CATALOGED BY      EXPIRATION DATE      LAST UPDATED BY

*****
* SEGMENT 1 *
*****

NUMBER OF SEGMENT OCCURRENCES = 1      POSITION OF FIRST FIELD
NUMBER OF FIELDS IN SEGMENT = 4      LINE: 3 COLUMN: 28

```

```

SCREEN      I R      E P M N      A      C
FIELD      .SOURCE/DEST. / E H R R O U      C      O E/V .....OPERAND B.....      PARTIAL FIELD
NAME      QLF FLD NAME O Q I R T D M      T      N OP      QLF FIELD NAME / CONSTANT      STRT NUMB OPR

CUSTNO      N,CUSTNO      O      H      N N      X      NM      CUSTNO
N,CUSTNO      I      H      N N      M      IF      F,ATINID      EQ C,PF006
M      IF      F,ATINID      EQ C,PF007
U      IF      F,ATINID      EQ C,ENTER
LOC/XATTRS: LINE 3,COL 28 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

CUSTNAME      N,CUSTNAME      O      H      N N      P      NM      CUSTNAME
N,CUSTNAME      I      H      N N      LOC/XATTRS: LINE 4,COL 28 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

CUSTPH      N,CUSTPH      O      H      N N      P      NM      CUSTPH
N,CUSTPH      I      H      N N      LOC/XATTRS: LINE 5,COL 28 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

*NONAME*      F,MSGLINE      O      H
LOC/XATTRS: LINE 24,COL 2 =(COLOR=N,HIGHLIGHT= ,PGM SYMBOL= )

```

```

                                GLOSSARY FOR SCREEN - CUSTUPD4
                                SCREEN CONTENT
                                DEC 04, 2003
                                PAGE 2

*****
* SEGMENT 2 *
*****

NUMBER OF SEGMENT OCCURRENCES = 5      POSITION OF FIRST FIELD
NUMBER OF FIELDS IN SEGMENT = 8      LINE: 11 COLUMN: 10

```

Figure A-2 Program Listings for Sample Applications (Page 11 of 84)


```

-----
SCREEN          I   R   E   P   M   N           A   C
FIELD .SOURCE/DEST. / E H R R O U           C   O E/V .....OPERAND B.....
NAME QLF FLD NAME O Q I R T D M           T   N OP QLF FIELD NAME / CONSTANT PARTIAL FIELD
                                           STRT NUMB OPR

ACTION          I   H   N   N           N           NM ACTION
                LOC/XATTRS: LINE 11,COL 10 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ORDERNO N,ORDERNO O
N,ORDERNO I   H   N   N           X           NM ORDERNO
                                           U IF S,ACTION EQ C,U
                                           M IF S,ACTION EQ C,X
                LOC/XATTRS: LINE 11,COL 13 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ORDRDATE N,ORDRDATE O
N,ORDRDATE I   H   N   N           P           NM ORDRDATE
                LOC/XATTRS: LINE 11,COL 20 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ORPERSON N,ORPERSON O
N,ORPERSON I   H   N   N           P           NM ORPERSON
                LOC/XATTRS: LINE 11,COL 28 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ORDPONUM N,ORDPONUM O
N,ORDPONUM I   H   N   N           P           NM ORDPONUM
                LOC/XATTRS: LINE 11,COL 35 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ORDDUDAT N,ORDDUDAT O
N,ORDDUDAT I   H   N   N           P           NM ORDDUDAT
                LOC/XATTRS: LINE 11,COL 42 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ORDINVGN N,ORDINVGN O
N,ORDINVGN I   H   N   N           P           NM ORDIVGN
                LOC/XATTRS: LINE 11,COL 53 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ORDCOMPLT N,ORDCOMPLT O
N,ORDCOMPLT I   H   N   N           P           NM ORDCOMPLT
                LOC/XATTRS: LINE 11,COL 61 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

** MK5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** MK5WC03 TYPE 0 SCREEN DEFINITION ITMERWS4 CATALOGED PRIOR TO VALIDATION.
-----
SCREEN STMT CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN PROC TERMINAL
NAME TYPE ACTN TYPE DESIG DESIG FORMAT CONTROL CODE ID NULL DEVICE SEQUENCE
                                           73 TO 80

(ITMERWS4) (SD) (R)          ( ) (0)          (M502 ) (M502MAP)

SCREEN STMT LINE CONT- EXIN ....+...1....+...2....+...3....+...4....+...5....+.. <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6....+...7....+...8....+...9....+...0....+...1.... <-TEXT (58-114) 73 TO 80
-----

(ITMERWS4,SL) ( 1) (0) ( ITMERWS4 NEXT SCREEN OF ITEMS=PA1 )
(ITMERWS4,SL) ( 2) (0) ( 'U' ACTION=ITEM UPDATE )
(ITMERWS4,SL) ( 3) (0) ( 'B' ACTION=BACK-ORDERS )
(ITMERWS4,SL) ( 4) (0) ( 'M' ACTION=MENU )
(ITMERWS4,SL) ( 5) (0) ( PRODUCT RANGE: _____ TO _____ )
(ITMERWS4,SL) ( 5) (1) ( QUANTITY )
(ITMERWS4,SL) ( 6) (0) ( ACT ITEM ITEM QUANTITY SHIPPED ORDERED )
(ITMERWS4,SL) ( 6) (1) ( BACK )
(ITMERWS4,SL) ( 7) (0) ( ION NUMBER PRICE ON-HAND YTD YTD )
(ITMERWS4,SL) ( 7) (1) ( ORDERED )
(ITMERWS4,SL) ( 9) (0) ( )
(ITMERWS4,SL) ( 9) (1) ( )
(ITMERWS4,SL) (10) (0) ( )
(ITMERWS4,SL) (10) (1) ( )
(ITMERWS4,SL) (11) (0) ( )
(ITMERWS4,SL) (11) (1) ( )

```

Figure A-2 Program Listings for Sample Applications (Page 12 of 84)

```

(ITMRRWS4,SL) ( 12) (0) ( )
(ITMRRWS4,SL) ( 12) (1) ( )
(ITMRRWS4,SL) ( 13) (0) ( )
(ITMRRWS4,SL) ( 13) (1) ( )
(ITMRRWS4,SL) ( 14) (0) ( )
(ITMRRWS4,SL) ( 14) (1) ( )
(ITMRRWS4,SL) ( 15) (0) ( )
(ITMRRWS4,SL) ( 15) (1) ( )
(ITMRRWS4,SL) ( 16) (0) ( )
(ITMRRWS4,SL) ( 16) (1) ( )
(ITMRRWS4,SL) ( 17) (0) ( )
(ITMRRWS4,SL) ( 17) (1) ( )
(ITMRRWS4,SL) ( 18) (0) ( )
(ITMRRWS4,SL) ( 18) (1) ( )
(ITMRRWS4,SL) ( 19) (0) ( )
(ITMRRWS4,SL) ( 19) (1) ( )
(ITMRRWS4,SL) ( 20) (0) ( )
(ITMRRWS4,SL) ( 20) (1) ( )
(ITMRRWS4,SL) ( 21) (0) ( )
(ITMRRWS4,SL) ( 21) (1) ( )
(ITMRRWS4,SL) ( 22) (0) ( )
(ITMRRWS4,SL) ( 22) (1) ( )
(ITMRRWS4,SL) ( 23) (0) ( )
(ITMRRWS4,SL) ( 23) (1) ( )

```

** MSGWL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

(ITMRRWS4,SS) (1) (1)

SCREEN NAME	SIMT TYPE	SEG NO.	S A	V QLF	DEST FLD NAME	IR OD	CE DIR	PM RRT	MN TDM	ALC T	LOC VN	OP OP	OPERAND B. QLF FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO	SEQUENCE 73 TO 80
(ITMRRWS4,S0)	(1)		(T,SYNCFD1)	(B)						(P)	(NM)	(SYNCFD1)	)		
(ITMRRWS4,S0)	(1)		(T,SYNCFD2)	(B)						(P)	(NM)	(SYNCFD2)	)		
(ITMRRWS4,S0)	(1)	(A)	(F,MSGLINE)						(H)						(*SYSTEM*)

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

(ITMRRWS4,SS) (2) (15)

SCREEN NAME	SIMT TYPE	SEG NO.	S A	V QLF	DEST FLD NAME	IR OD	CE DIR	PM RRT	MN TDM	ALC T	LOC VN	OP OP	OPERAND B. QLF FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO	SEQUENCE 73 TO 80
(ITMRRWS4,S0)	(2)					(I)	(Y)			(N)	(NM)	(ACTION	)		
(ITMRRWS4,S0)	(2)		(1,ITEMNO)			(I)				(X)	(NM)	(ITEMNO	)		
(ITMRRWS4,S1)										(M)	(IF)	(S,ACTION)	(EQ) (C,U)		
(ITMRRWS4,S1)										(M)	(IF)	(S,ACTION)	(EQ) (C,M)		
(ITMRRWS4,S0)	(2)	(S)	(2,ITEMNO)			(O)									
(ITMRRWS4,S0)	(2)		(2,ITEMPRICE)			(O)				(NM)	(ITEMPRICE	)			
(ITMRRWS4,S0)	(2)		(2,QANONHND)			(O)				(NM)	(QANONHND	)			
(ITMRRWS4,S0)	(2)		(2,SHIPYTD)			(O)				(NM)	(SHIPYTD	)			
(ITMRRWS4,S0)	(2)		(2,ORDYTD)			(O)				(NM)	(ORDYTD	)			
(ITMRRWS4,S0)	(2)		(2,QTYBKORD)			(O)				(NM)	(QTYBKORD	)			

GLOSSARY FOR SCREEN - ITMRRWS4
SCREEN FORMAT

DEC 04, 2003
PAGE 1

*	1	2	3	4	5	6	7	8*
1	ITMRRWS4				NEXT SCREEN OF ITEMS=PA1			1
2					'U' ACTION=ITEM UPDATE			2
3					'B' ACTION=BACK-ORDERS			3
4					'M' ACTION=MENU			4
5	PRODUCT RANGE:	TO			QUANTITY			5
6	ACT ITEM	ITEM	QUANTITY	SHIPPED	ORDERED	BACK		6
7	ION NUMBER	PRICE	ON-HAND	YTD	YTD	ORDERED		7
8								8
9								9

Figure A-2 Program Listings for Sample Applications (Page 13 of 84)

```
10 | | | | | | | | | | 10
11 | | | | | | | | | | 11
12 | | | | | | | | | | 12
13 | | | | | | | | | | 13
14 | | | | | | | | | | 14
15 | | | | | | | | | | 15
16 | | | | | | | | | | 16
17 | | | | | | | | | | 17
18 | | | | | | | | | | 18
19 | | | | | | | | | | 19
20 | | | | | | | | | | 20
21 | | | | | | | | | | 21
22 | | | | | | | | | | 22
23 | | | | | | | | | | 23
24 | | | | | | | | | | 24

*-----1-----2-----3-----4-----5-----6-----7-----8*

```

```
SCREEN IDENTIFIER.. M510      FIELD DESIGNATOR.....      INITIAL CURSOR POSITION
TRANSACTION CODE... M502      HIGHLIGHT DESIGNATOR.. @      LINE: 9 COLUMN: 13
                                OUTPUT CONTROL..... S

```

```

                                GLOSSARY FOR SCREEN - ITIMRWS4      DEC 04, 2003
                                SCREEN CONTENT                        PAGE 1

DATE CATALOGED 12/04/03 20:55:00      DATE LAST USED      DATE LAST UPDATED 12/04/03 20:55:00
CATALOGED BY      EXPIRATION DATE      LAST UPDATED BY

                                *****
                                * SEGMENT 1 *
                                *****

NUMBER OF SEGMENT OCCURRENCES = 1      POSITION OF FIRST FIELD
NUMBER OF FIELDS IN SEGMENT = 3      LINE: 5 COLUMN: 17

```

```
-----
SCREEN      I R      E P M N      A C
FIELD .SOURCE/DEST. / E H R R O U      C O E/V .....OPERAND B..... PARTIAL FIELD
NAME QLF FLD NAME O Q I R T D M      T N OP QLF FIELD NAME / CONSTANT      STRT NUMB OPR
-----

```

```
SYNCFD1 T,SYNCFD1 O
T,SYNCFD1 I H N N P NM SYNCFD1
LOC/XATTRS: LINE 5,COL 17 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

SYNCFD2 T,SYNCFD2 O
T,SYNCFD2 I H N N P NM SYNCFD2
LOC/XATTRS: LINE 5,COL 28 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

*NONAME* F,MSGLINE O H
LOC/XATTRS: LINE 24,COL 2 =(COLOR=N,HIGHLIGHT= ,PGM SYMBOL= )

```

```

                                GLOSSARY FOR SCREEN - ITIMRWS4      DEC 04, 2003
                                SCREEN CONTENT                        PAGE 2

                                *****
                                * SEGMENT 2 *
                                *****

NUMBER OF SEGMENT OCCURRENCES = 15      POSITION OF FIRST FIELD
NUMBER OF FIELDS IN SEGMENT = 7      LINE: 9 COLUMN: 13

```

```
-----
SCREEN      I R      E P M N      A C
FIELD .SOURCE/DEST. / E H R R O U      C O E/V .....OPERAND B..... PARTIAL FIELD
NAME QLF FLD NAME O Q I R T D M      T N OP QLF FIELD NAME / CONSTANT      STRT NUMB OPR
-----

```

```
ACTION      I H N N N NM ACTION
LOC/XATTRS: LINE 9,COL 13 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ITEMNO 2,ITEMNO O
1,ITEMNO I H N N X NM ITEMNO

```

Figure A-2 Program Listings for Sample Applications (Page 14 of 84)

```

M      IF S,ACTION EQ C,U
M      IF S,ACTION EQ C,M
LOC/XATTRS: LINE 9,COL 15 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ITMPRICE 2,ITMPRICE 0
LOC/XATTRS: LINE 9,COL 23 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

QANONHND 2,QANONHND 0
LOC/XATTRS: LINE 9,COL 33 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

SHIPYTD 2,SHIPYTD 0
LOC/XATTRS: LINE 9,COL 41 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

ORDYTD 2,ORDYTD 0
LOC/XATTRS: LINE 9,COL 50 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

QTYBKORD 2,QTYBKORD 0
LOC/XATTRS: LINE 9,COL 59 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

```

** M5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** M5WC03 TYPE 0 SCREEN DEFINITION ITMEKOD4 CATALOGED PRIOR TO VALIDATION.

SCREEN NAME	STMT TYPE	CAT ACIN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

(ITMEKOD4) (SD)	(R)		()	(@)			(M503)		(ITMEKOD)			

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80

```

(ITMEKOD4,SL) ( 1) (0) ( ITMEKOD4 )
(ITMEKOD4,SL) ( 3) (0) ( ITEM NUMBER _ PRICE _ QUANTITY )
(ITMEKOD4,SL) ( 3) (1) (/BOX _ )
(ITMEKOD4,SL) ( 4) (0) ( QUANTITY ON HAND _ SHIPPED YTD _ )
(ITMEKOD4,SL) ( 4) (1) ( )
(ITMEKOD4,SL) ( 5) (0) ( ORDERED YTD _ QUANTITY BACK-ORDERED )
(ITMEKOD4,SL) ( 5) (1) ( )
(ITMEKOD4,SL) ( 6) (0) ( REORDER POINT _ PRODUCT CODE )
(ITMEKOD4,SL) ( 7) (0) ( ***** )
(ITMEKOD4,SL) ( 7) (1) ( ***** )
(ITMEKOD4,SL) ( 8) (0) ( * SPECIFIED SCREEN OF BACK-ORDERS=SCREEN# + )
(ITMEKOD4,SL) ( 8) (1) ( ENTER * )
(ITMEKOD4,SL) ( 9) (0) ( * NEXT SCREEN OF BACK-ORDERS=ENTER )
(ITMEKOD4,SL) ( 9) (1) ( * )
(ITMEKOD4,SL) ( 10) (0) ( * FILL A BACK-ORDER='X' + PF8 )
(ITMEKOD4,SL) ( 10) (1) ( * )
(ITMEKOD4,SL) ( 11) (0) ( * ADD BACK-ORDER=PF9 RETURN TO MENU=PF10 )
(ITMEKOD4,SL) ( 11) (1) ( * )
(ITMEKOD4,SL) ( 12) (0) ( ***** )
(ITMEKOD4,SL) ( 12) (1) ( ***** )
(ITMEKOD4,SL) ( 13) (0) ( FILL BACK CUSTOMER ORDER )
(ITMEKOD4,SL) ( 13) (1) ( SCREEN _ )
(ITMEKOD4,SL) ( 14) (0) ( BACK ORDER NUMBER NUMBER )
(ITMEKOD4,SL) ( 14) (1) ( _ OF _ )
(ITMEKOD4,SL) ( 15) (0) ( ORDERS NUMBER )
(ITMEKOD4,SL) ( 16) (0) ( )
(ITMEKOD4,SL) ( 17) (0) ( _ )
(ITMEKOD4,SL) ( 18) (0) ( _ )
(ITMEKOD4,SL) ( 19) (0) ( _ )
(ITMEKOD4,SL) ( 20) (0) ( _ )
(ITMEKOD4,SL) ( 21) (0) ( _ )
(ITMEKOD4,SL) ( 22) (0) ( _ )
(ITMEKOD4,SL) ( 23) (0) ( )

```

** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

(ITMEKOD4,SS)	(1)	(1)		

SCREEN NAME	STMT TYPE	SEG NO.	SOURCE/DEST QLF FLD NAME	I R C E P M N / Q H U R R O U O D I R R T D M	A L C C L O E/V T V N O P	OPERAND B..... QLF FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80

Figure A-2 Program Listings for Sample Applications (Page 15 of 84)

V		I R C E P M N		A I C		PARTIAL	
SCREEN	STMT	SEG S	SOURCE/DEST	/ Q H U R R O U	C L O E/V	OPERAND B.....	FIELD
NAME	TYPE	NO. A	QLF FLD NAME	O D I R R T D M	T V N O P	QLF FIELD NAME OR CONSTANT	SEQUENCE
							ST NO OP 73 TO 80
(TIMEKOD4,S0)	(1)		(1,ORDYTD)	(O)		(NM) (ORDYTD)	
(TIMEKOD4,S0)	(1)		(1,QTYEKORD)	(O)		(NM) (QTYEKORD)	
(TIMEKOD4,S0)	(1)		(1,REORDPT)	(O)		(NM) (REORDPT)	
(TIMEKOD4,S0)	(1)		(1,PRODCODE)	(O)		(NM) (PRODCODE)	
(TIMEKOD4,S0)	(1)			(I)	(N)	(NM) (REQSCRN)	
(TIMEKOD4,S0)	(1)		(T,SCREENNO)	(B)	(Y)	(P) (NM) (SCREENNO)	
(TIMEKOD4,S0)	(1)		(T,NOSCREEN)	(B)	(Y)	(P) (NM) (NOSCREEN)	
(TIMEKOD4,S0)	(1)	(A)	(F,MSGLINE)				(*SYSTEM*)

(ITMBKOD4,SS) (2) (8)

SCREEN	V	I R C E P M N	A I C	PARTIAL
NAME	SIMT SEG S	SOURCE/DEST / Q H U R R O U	C L O E/V OPERAND B.....	FIELD SEQUENCE
NAME	TYPE NO. A	QLF FLD NAME O D I R T D M	T V N O P QLF FIELD NAME OR CONSTANT	ST NO OF 73 TO 80
(ITEMKOD4,S0) (2)		(I) (Y)	(X) (NM) (XFIELD))	
(ITEMKOD4,S0) (2)		(1,BKORDRNO) (B)	(X) (NM) (BKORDRNO	
(ITEMKOD4,S1)			(M) (IF) (S,XFIELD) (EQ) (C,X))	
(ITEMKOD4,S0) (2)		(N,CUSTNO) (I)	(X) (NM) (CUSTNO))	
(ITEMKOD4,S1)			(M) (IF) (S,XFIELD) (EQ) (C,X))	
(ITEMKOD4,S0) (2) (S)		(1,CUSTNO) (O)		
(ITEMKOD4,S0) (2)		(N,ORDERNO) (I)	(X) (NM) (ORDERNO))	
(ITEMKOD4,S1)			(M) (IF) (S,XFIELD) (EQ) (C,X))	
(ITEMKOD4,S0) (2) (S)		(1,ORDERNO) (O)		
(ITEMKOD4,S0) (2) (V)		(N,ITEMORD) (I)	(X) (NM) (ITEMORD))	
(ITEMKOD4,S1)			(M) (IF) (S,XFIELD) (EQ) (C,X))	

```

1 1-----1-----2-----3-----4-----5-----6-----7-----8-----
2 |
3 | ITEMKOD4 |
4 |
5 | ITEM NUMBER PRICE QUANTITY/BOX
6 | QUANTITY ON HAND SHIPPED YTD
7 | ORDERED YTD QUANTITY BACK-ORDERED
8 | REORDER POINT PRODUCT CODE
9 | *****
10 | * SPECIFIED SCREEN OF BACK-ORDERS=SCREEN# + ENTER *
11 | * NEXT SCREEN OF BACK-ORDERS=ENTER *
12 | * FILL A BACK-ORDER='X' + PF8 *
13 | * ADD BACK-ORDER=PF9 RETURN TO MENU=PF10 *
14 | *****
15 | FILL BACK CUSTOMER ORDER SCREEN
16 | BACK ORDER NUMBER NUMBER OF
17 | ORDERS NUMBER
18 |
19 |
20 |
21 |
22 |
23 |
24 |
24 1-----1-----2-----3-----4-----5-----6-----7-----8-----

```

Order/Item Application Screen and Database Definition Code A-19

```

                                SCREEN IDENTIFIER... M510      FIELD DESIGNATOR.....      INITIAL CURSOR POSITION
                                TRANSACTION CODE... M503        HIGHLIGHT DESIGNATOR.. @      LINE: 16 COLUMN: 26
                                                                OUTPUT CONTROL..... S

                                GLOSSARY FOR SCREEN - ITMBKOD4      DEC 04, 2003
                                SCREEN CONTENT                      PAGE 1

DATE CATALOGED 12/04/03 20:55:00      DATE LAST USED      DATE LAST UPDATED 12/04/03 20:55:00
CATALOGED BY      EXPIRATION DATE      LAST UPDATED BY

                                *****
                                * SEGMENT 1 *
                                *****

NUMBER OF SEGMENT OCCURRENCES = 1      POSITION OF FIRST FIELD
NUMBER OF FIELDS IN SEGMENT = 13      LINE: 3 COLUMN: 26

```

```

-----
SCREEN      I R E P M N      A C
FIELD .SOURCE/DEST. / E H R R O U      C O E/V .....OPERAND B..... PARTIAL FIELD
NAME QLF FLD NAME O Q I R T D M      T N OP QLF FIELD NAME / CONSTANT      STRT NUMB OPR
-----

```

```

ITEMNO 1,ITEMNO O
1,ITEMNO I H Y N N M
                                NM ITEMNO
                                X IF F,ATINID EQ C,PF010
LOC/XATTRS: LINE 3,COL 26 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

ITEMPRICE 1,ITEMPRICE O
LOC/XATTRS: LINE 3,COL 40 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

PACKBOX 1,PACKBOX O
LOC/XATTRS: LINE 3,COL 63 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

QANONHND 1,QANONHND O
LOC/XATTRS: LINE 4,COL 31 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

SHIPYTD 1,SHIPYTD O
LOC/XATTRS: LINE 4,COL 51 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

CRDYTD 1,CRDYTD O
LOC/XATTRS: LINE 5,COL 26 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

QTYBKORD 1,QTYBKORD O
LOC/XATTRS: LINE 5,COL 57 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

REORDPT 1,REORDPT O
LOC/XATTRS: LINE 6,COL 28 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

PRODCODE 1,PRODCODE O
LOC/XATTRS: LINE 6,COL 48 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

REQSCRN I H N N N
                                NM REQSCRN
LOC/XATTRS: LINE 13,COL 69 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

SCREENNO T,SCREENNO O
T,SCREENNO I H Y N N P
                                NM SCREENNO
LOC/XATTRS: LINE 14,COL 62 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

NOSCREEN T,NOSCREEN O

```

```

                                GLOSSARY FOR SCREEN - ITMBKOD4      DEC 04, 2003
                                SCREEN CONTENT                      PAGE 2
                                SEGMENT 1 CONTINUED

```

```

-----
SCREEN      I R E P M N      A C
FIELD .SOURCE/DEST. / E H R R O U      C O E/V .....OPERAND B..... PARTIAL FIELD
NAME QLF FLD NAME O Q I R T D M      T N OP QLF FIELD NAME / CONSTANT      STRT NUMB OPR
-----

```

```

NOSCREEN T,NOSCREEN I H Y N N P
                                NM NOSCREEN
LOC/XATTRS: LINE 14,COL 69 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

```

Figure A-2 Program Listings for Sample Applications (Page 17 of 84)

NONAME F,MSGLINE O H
LOC/XATTRS: LINE 24,COL 2 =(COLOR=N,HIGHLIGHT= ,PGM SYMBOL=)

GLOSSARY FOR SCREEN - ITMBKOD4
SCREEN CONTENT

DEC 04, 2003
PAGE 3

* SEGMENT 2 *

NUMBER OF SEGMENT OCCURRENCES = 8
NUMBER OF FIELDS IN SEGMENT = 5

POSITION OF FIRST FIELD
LINE: 16 COLUMN: 26

SCREEN		I	R	E	P	M	N		A	C			
FIELD	.SOURCE/DEST.	/	E	H	R	R	O	U	C	O	E/V	OPERAND B.....	PARTIAL FIELD
NAME	QLF FLD NAME	O	Q	I	R	T	D	M	T	N	OP	QLF FIELD NAME / CONSTANT	SIRT NUMB OPR

XFIELD I H N N N NM XFIELD
LOC/XATTRS: LINE 16,COL 26 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)

BKORDRNO 1,BKORDRNO O
1,BKORDRNO I H N N X
NM BKORDRNO
IF S,XFIELD EQ C,X
M
LOC/XATTRS: LINE 16,COL 32 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)

CUSTNO 1,CUSTNO O
N,CUSTNO I H N N X
NM CUSTNO
IF S,XFIELD EQ C,X
M
LOC/XATTRS: LINE 16,COL 41 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)

ORDERNO 1,ORDERNO O
N,ORDERNO I H N N X
NM ORDERNO
IF S,XFIELD EQ C,X
M
LOC/XATTRS: LINE 16,COL 51 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)

ITEMORD N,ITEMORD I H N X
NM ITEMORD
IF S,XFIELD EQ C,X
M
VIRTUAL SCREEN FIELD

** MK5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** MK5WC03 TYPE 0 SCREEN DEFINITION ITMUPDT4 CATALOGED PRIOR TO VALIDATION.

SCREEN	SIMT	CAT	GLOSS	FIELD	HILITE	AUTO	OUTPUT	TRANSACT	SCREEN	PROC	TERMINAL	SEQUENCE
NAME	TYPE	ACIN	TYPE	DESIG	DESIG	FORMAT	CONTROL	CODE	ID	NULL	DEVICE	73 TO 80

(ITMUPDT4) (SD) (R) () (0) (M504) (ITMUPDT)

SCREEN	SIMT	LINE	CONT-	EXTN	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE
NAME	TYPE	NO.	INUE	ATTR	..6.....7.....8.....9.....0.....1.....	<-TEXT (58-114)	73 TO 80

(ITMUPDT4,SL)	(1)	(0)	(ITMUPDT4	)
(ITMUPDT4,SL)	(1)	(1)	(*****	)
(ITMUPDT4,SL)	(2)	(0)	(	)
(ITMUPDT4,SL)	(2)	(1)	(* MODIFY RECORD=ENTER *	)
(ITMUPDT4,SL)	(3)	(0)	(ITEM NUMBER	)
(ITMUPDT4,SL)	(3)	(1)	(* RECORD DELETE=PF11 *	)
(ITMUPDT4,SL)	(4)	(0)	(PRICE	)
(ITMUPDT4,SL)	(4)	(1)	(* RETURN TO MENU=PF10 *	)
(ITMUPDT4,SL)	(5)	(0)	(QUANTITY/BOX	)
(ITMUPDT4,SL)	(5)	(1)	(*****	)
(ITMUPDT4,SL)	(6)	(0)	(QUANTITY ON HAND	)
(ITMUPDT4,SL)	(7)	(0)	(SHIPPED YTD	)
(ITMUPDT4,SL)	(8)	(0)	(ORDERED YTD	)
(ITMUPDT4,SL)	(9)	(0)	(QUANTITY BACK-ORDERED	)

Figure A-2 Program Listings for Sample Applications (Page 18 of 84)

SCREEN NAME	SMT TYPE	SEG NO.	S NO.	OF OCCURRENCES	SEQUENCE 73 TO 80
(ITMUPDT4,SS)	(1)	(1)			
SCREEN NAME	SMT TYPE	SEG NO.	S NO.	OF OCCURRENCES	SEQUENCE 73 TO 80
(ITMUPDT4,S0)	(1)	(T,DELETE)	(B)	(D)	(Y,Y)
(ITMUPDT4,S0)	(1)	(1,ITEMNO)	(B)	(Y)	(X)
(ITMUPDT4,S1)					(D)
(ITMUPDT4,S1)					(X)
(ITMUPDT4,S0)	(1)	(1,TIMPRICE)	(B)		(P)
(ITMUPDT4,S0)	(1)	(1,PACKBOX)	(B)		(P)
(ITMUPDT4,S0)	(1)	(1,QANONHND)	(B)		(P)
(ITMUPDT4,S0)	(1)	(1,SHIPYTD)	(B)		(P)
(ITMUPDT4,S0)	(1)	(1,ORDYTD)	(B)		(P)
(ITMUPDT4,S0)	(1)	(1,QTYEKORD)	(C)		(P)
(ITMUPDT4,S0)	(1)	(1,REORDPT)	(B)		(P)
(ITMUPDT4,S0)	(1)	(1,PRDCODE)	(B)		(P)
(ITMUPDT4,S0)	(1)	(1,A)	(F,MSGLINE)	(H)	

```

*-----1-----2-----3-----4-----5-----6-----7-----8*
1 | ITM/PDT4                                     ***** 1
2 |                                                * MODIFY RECORD=ENTER * 2
3 | ITEM NUMBER _____ * RECORD DELETE=PF11 * 3
4 | PRICE _____ * RETURN TO MENU=PF10 * 4
5 | QUANTITY/BOX _____ ***** 5
6 | QUANTITY ON HAND _____ 6
7 | SHIPPED YTD _____ 7
8 | ORDERED YTD _____ 8
9 | QUANTITY BACK-ORDERED _____ 9
10 | RE-ORDER POINT _____ 10
11 | PRODUCT CODE _____ 11
12 | 12
13 | 13
14 | 14
15 | 15
16 | 16
17 | 17
18 | 18
19 | 19
20 | 20
21 | 21
22 | 22
23 | 23
24 | _____ 24

*-----1-----2-----3-----4-----5-----6-----7-----8*

REEN IDENTIFIER.. M510          FIELD DESIGNATOR..... INITIAL CURSOR POSITION
ANSACTION CODE... M504          HIGHLIGHT DESIGNATOR.. @   LINE: 4 COLUMN: 25
                                OUTPUT CONTROL..... S

```

Figure A-2 Program Listings for Sample Applications (Page 19 of 84)

SCREEN FIELD NAME	.SOURCE/DEST. QLF FLD NAME	I O	R Q	E I	P R	M T	N D	A C	C O	E/V	OPERAND B.....	PARTIAL FIELD STRT NUMB OFR
DELETE	T,DELETE	O		D								
	T,DELETE	I			Y	Y	N					
	LOC/XATTRS: LINE											
ITEMNO	1,ITEMNO	O										
	1,ITEMNO	I			H	Y	N	N				
	LOC/XATTRS: LINE											
ITMPRICE	1,ITMPRICE	O										
	1,ITMPRICE	I			H		N	N				
	LOC/XATTRS: LINE											
PACKBOX	1,PACKBOX	O										
	1,PACKBOX	I			H		N	N				
	LOC/XATTRS: LINE											
QANONHND	1,QANONHND	O										
	1,QANONHND	I			H		N	N				
	LOC/XATTRS: LINE											
SHIPYTD	1,SHIPYTD	O										
	1,SHIPYTD	I			H		N	N				
	LOC/XATTRS: LINE											
ORDYTD	1,ORDYTD	O										
	1,ORDYTD	I			H		N	N				
	LOC/XATTRS: LINE											
QTYBKORD	1,QTYBKORD	O										
	LOC/XATTRS: LINE											
REORDPT	1,REORDPT	O										

GLOSSARY FOR SCREEN - ITMUPDT4
SCREEN CONTENT
SEGMENT 1 CONTINUED

DEC 04, 2003
PAGE 2

SCREEN FIELD NAME	.SOURCE/DEST. QLF FLD NAME	I O	R Q	E I	P R	M T	N D	A C	C O	E/V	OPERAND B.....	PARTIAL FIELD STRT NUMB OFR
REORDPT	1,REORDPT	I			H		N	N				
	LOC/XATTRS: LINE											
PRODCODE	1,PRODCODE	O										
	1,PRODCODE	I			H		N	N				
	LOC/XATTRS: LINE											
NONAME	F,MSGLINE	O			H							
	LOC/XATTRS: LINE											

** MK5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** MK5WC03 TYPE 0 SCREEN DEFINITION ENDSR4 CATALOGED PRIOR TO VALIDATION.

SCREEN NAME	STMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

Figure A-2 Program Listings for Sample Applications (Page 20 of 84)

```

-----
(ENDSCRN4) (SD) (R)          ( ) (8)
-----

SCREEN STMT LINE CONT- EXTN ....+...1...+...2...+...3...+...4...+...5...+.. <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6...+...7...+...8...+...9...+...0...+...1... <-TEXT (58-114) 73 TO 80
-----

(ENDSCRN4,SL) ( 1) (0)      ( ENDSRN4                                )
(ENDSCRN4,SL) ( 4) (0)      (                                         )
(ENDSCRN4,SL) ( 6) (0)      (                                         )
(ENDSCRN4,SL) ( 8) (0)      (                                         )
(ENDSCRN4,SL) (10) (0)      (                                         )
(ENDSCRN4,SL) (13) (0)      (***** )
(ENDSCRN4,SL) (13) (1)      (***** )
(ENDSCRN4,SL) (14) (0)      (                                         )
(ENDSCRN4,SL) (15) (0)      (                                         )
** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----

SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES
-----

(ENDSCRN4,SS) ( 1) ( 1)
-----

SCREEN STMT SEG S V I R C E P M N A L C PARTIAL
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP FIELD SEQUENCE
-----

(ENDSCRN4,S0) ( 1) (F,DATE ) (0) (NM) (DATE )
(ENDSCRN4,S0) ( 1) (F,TIME ) (0) (NM) (TIME )
(ENDSCRN4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)
-----

GLOSSARY FOR SCREEN - ENDSRN4 DEC 04, 2003
SCREEN FORMAT PAGE 1

*-----1-----2-----3-----4-----5-----6-----7-----8*
1 | ENDSRN4 | 1
2 | | 2
3 | | 3
4 | | 4
5 | | 5
6 | | 6
7 | | 7
8 | | 8
9 | | 9
10 | | 10
11 | | 11
12 | | 12
13 | ***** | 13
14 | | 14
15 | | 15
16 | | 16
17 | | 17
18 | | 18
19 | | 19
20 | | 20
21 | | 21
22 | | 22
23 | | 23
24 | | 24
*-----1-----2-----3-----4-----5-----6-----7-----8*

SCREEN IDENTIFIER.. M510 FIELD DESIGNATOR..... INITIAL CURSOR POSITION
TRANSACTION CODE... HIGHLIGHT DESIGNATOR.. @ LINE: 1 COLUMN: 1
OUTPUT CONTROL..... S
-----

GLOSSARY FOR SCREEN - ENDSRN4 DEC 04, 2003
SCREEN CONTENT PAGE 1

DATE CATALOGED 12/04/03 20:55:00 DATE LAST USED DATE LAST UPDATED 12/04/03 20:55:00
CATALOGED BY EXPIRATION DATE LAST UPDATED BY

*****
* SEGMENT 1 *

```

Figure A-2 Program Listings for Sample Applications (Page 21 of 84)

```

*****
NUMBER OF SEGMENT OCCURRENCES = 1
NUMBER OF FIELDS IN SEGMENT = 3
POSITION OF FIRST FIELD
LINE: 14 COLUMN: 11
-----
SCREEN          I R   E P M N          A   C
FIELD .SOURCE/DEST. / E H R R O U      C   O E/V .....OPERAND B..... PARTIAL FIELD
NAME   QLF FLD NAME O Q I R T D M      T   N OP   QLF FIELD NAME / CONSTANT  STRT NUMB OPR
-----

DATE      F,DATE      O
LOC/XATRS: LINE 14,COL 11 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

TIME      F,TIME      O
LOC/XATRS: LINE 15,COL 11 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

*NONAME* F,MSGLINE    O   H
LOC/XATRS: LINE 24,COL 2 =(COLOR=N,HIGHLIGHT= ,PGM SYMBOL= )

** MK5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** MK5WC03 TYPE 0 SCREEN DEFINITION ORDUPDT4 CATALOGED PRIOR TO VALIDATION.

-----
SCREEN  SIMT CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN  PROC TERMINAL
NAME    TYPE ACIN TYPE DESIG DESIG FORMAT CONTROL CODE    ID    NULL  DEVICE
-----
(ORDUPDT4) (SD) (R)          ( ) ( @)          (M501 )

SCREEN  SIMT LINE CONT- EXIN .....1.....2.....3.....4.....5..... <-TEXT (1-57) SEQUENCE
NAME    TYPE NO.  INUE  ATTR ..6.....7.....8.....9.....0.....1.... <-TEXT (58-114) 73 TO 80
-----

(ORDUPDT4,SL) ( 1) (0)      ( ORDUPDT4 )
(ORDUPDT4,SL) ( 1) (1)      ( UPDATE=U ACT + ENTER )
(ORDUPDT4,SL) ( 2) (0)      ( CUSTOMER NUMBER )
(ORDUPDT4,SL) ( 2) (1)      ( CUST UPDT SCRN=PFT )
(ORDUPDT4,SL) ( 3) (0)      ( ORDER NUMBER )
(ORDUPDT4,SL) ( 3) (1)      ( RETURN TO MENU=PFT0 )
(ORDUPDT4,SL) ( 4) (0)      ( CUSTOMER NAME )
(ORDUPDT4,SL) ( 5) (1)      ( QTY )
(ORDUPDT4,SL) ( 6) (0)      ( ACT ITEM          ITEM          PRICE          QUANTITY )
(ORDUPDT4,SL) ( 6) (1)      ( BACK          ON )
(ORDUPDT4,SL) ( 7) (0)      ( ION NUMBER          NAME          ORDERED )
(ORDUPDT4,SL) ( 7) (1)      ( ORDERED HAND )
(ORDUPDT4,SL) ( 8) (0)      ( - )
(ORDUPDT4,SL) ( 8) (1)      ( - )
(ORDUPDT4,SL) ( 9) (0)      ( - )
(ORDUPDT4,SL) ( 9) (1)      ( - )
(ORDUPDT4,SL) (10) (0)      ( - )
(ORDUPDT4,SL) (10) (1)      ( - )
(ORDUPDT4,SL) (11) (0)      ( - )
(ORDUPDT4,SL) (11) (1)      ( - )
(ORDUPDT4,SL) (12) (0)      ( - )
(ORDUPDT4,SL) (12) (1)      ( - )
(ORDUPDT4,SL) (13) (0)      ( - )
(ORDUPDT4,SL) (13) (1)      ( - )
(ORDUPDT4,SL) (14) (0)      ( - )
(ORDUPDT4,SL) (14) (1)      ( - )
(ORDUPDT4,SL) (15) (0)      ( - )
(ORDUPDT4,SL) (15) (1)      ( - )
(ORDUPDT4,SL) (16) (0)      ( - )
(ORDUPDT4,SL) (16) (1)      ( - )
(ORDUPDT4,SL) (17) (0)      ( - )
(ORDUPDT4,SL) (17) (1)      ( - )
(ORDUPDT4,SL) (18) (0)      ( - )
(ORDUPDT4,SL) (18) (1)      ( - )
(ORDUPDT4,SL) (19) (0)      ( - )
(ORDUPDT4,SL) (19) (1)      ( - )
(ORDUPDT4,SL) (20) (0)      ( - )
(ORDUPDT4,SL) (20) (1)      ( - )
(ORDUPDT4,SL) (21) (0)      ( - )
(ORDUPDT4,SL) (21) (1)      ( - )
(ORDUPDT4,SL) (22) (0)      ( - )

```

Figure A-2 Program Listings for Sample Applications (Page 22 of 84)

```

(ORDUPDT4,SL) ( 22) (1) ( _____ )
(ORDUPDT4,SL) ( 23) (0) ( _____ )
(ORDUPDT4,SL) ( 23) (1) ( _____ )
** M50M14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----
SCREEN  SIMT  SEGMENT  NO. OF          SEQUENCE
NAME  TYPE  NO.    OCCURRENCES          73 TO 80
-----
(ORDUPDT4,SS) ( 1)      ( 1)
-----
          V          I R   C E P M N      A L C          PARTIAL
SCREEN  SIMT  SEG S  SOURCE/DEST / Q H U R R O U  C L O E/V .....OPERAND B.....  FIELD  SEQUENCE
NAME  TYPE NO. A  QLF FLD NAME O D I R R T D M  T V N O P  QLF FIELD NAME OR CONSTANT  ST NO OP 73 TO 80
-----
(ORDUPDT4,S0) ( 1)      (N,CUSTNO ) (B)      (H,Y)      (X) (NM) (CUSTNO )
(ORDUPDT4,S1)          (M) (IF) (F,ATINID ) (EQ) (C,PF001 )
(ORDUPDT4,S1)          (M) (IF) (F,ATINID ) (EQ) (C,ENTER )
(ORDUPDT4,S0) ( 1)      (N,ORDERNO ) (B)      (Y)      (X) (NM) (ORDERNO )
(ORDUPDT4,S1)          (Q) (IF) (F,ATINID ) (EQ) (C,ENTER )
(ORDUPDT4,S0) ( 1)      (N,CUSTNAME) (O)          (NM) (CUSTNAME )
(ORDUPDT4,S0) ( 1) (A) (F,MSGLINE ) (H)                                     (*SYSTEM*)
-----
SCREEN  SIMT  SEGMENT  NO. OF          SEQUENCE
NAME  TYPE  NO.    OCCURRENCES          73 TO 80
-----
(ORDUPDT4,SS) ( 2)      ( 15)
-----
          V          I R   C E P M N      A L C          PARTIAL
SCREEN  SIMT  SEG S  SOURCE/DEST / Q H U R R O U  C L O E/V .....OPERAND B.....  FIELD  SEQUENCE
NAME  TYPE NO. A  QLF FLD NAME O D I R R T D M  T V N O P  QLF FIELD NAME OR CONSTANT  ST NO OP 73 TO 80
-----
(ORDUPDT4,S0) ( 2)          (I)      (H)      (N) (NM) (ACTION )
(ORDUPDT4,S0) ( 2)      (N,ITEMORD ) (B)      (X) (NM) (ITEMORD )
(ORDUPDT4,S1)          (V) (IF) (S,ACTION ) (EQ) (C,U )
(ORDUPDT4,S0) ( 2) (S) (1,ITEMNO ) (I)      (X)
(ORDUPDT4,S1)          (U) (IF) (S,ACTION ) (EQ) (C,U )
(ORDUPDT4,S0) ( 2)      (2,ITEMNAME) (O)          (NM) (ITEMNAME )
(ORDUPDT4,S0) ( 2)      (N,ITEMPRICE) (B)      (P) (NM) (ITEMPRICE )
(ORDUPDT4,S0) ( 2)      (N,ITEMQTYOR) (B)      (P) (NM) (ITEMQTYOR )
(ORDUPDT4,S0) ( 2)      (N,QTYEKORD) (B)      (P) (NM) (QTYEKORD )
(ORDUPDT4,S0) ( 2)      (2,QANONHND) (O)          (NM) (QANONHND )
-----
GLOSSARY FOR SCREEN - ORDUPDT4          DEC 04, 2003
SCREEN FORMAT          PAGE 1
-----
*-----1-----2-----3-----4-----5-----6-----7-----8*
1 | ORDUPDT4          UPDATE=U ACT + ENTER | 1
2 | CUSTOMER NUMBER  _____ CUST UPDT SCR=PF1 | 2
3 | ORDER NUMBER    _____ RETURN TO MENU=PF10 | 3
4 | CUSTOMER NAME    _____ | 4
5 | _____ | 5
6 | ACT ITEM        ITEM      PRICE  QUANTITY  BACK  QTY
7 | ION NUMBER      NAME      ORDERED ORDERED ON
8 | _____ | 8
9 | _____ | 9
10 | _____ | 10
11 | _____ | 11
12 | _____ | 12
13 | _____ | 13
14 | _____ | 14
15 | _____ | 15
16 | _____ | 16
17 | _____ | 17
18 | _____ | 18
19 | _____ | 19
20 | _____ | 20
21 | _____ | 21
22 | _____ | 22
23 | _____ | 23
24 | _____ | 24
*-----1-----2-----3-----4-----5-----6-----7-----8*
SCREEN IDENTIFIER.. M510          FIELD DESIGNATOR.....          INITIAL CURSOR POSITION
TRANSACTION CODE...          HIGHLIGHT DESIGNATOR.. 0          LINE: 9 COLUMN: 3

```

Figure A-2 Program Listings for Sample Applications (Page 23 of 84)

OUTPUT CONTROL..... S

GLOSSARY FOR SCREEN - ORDUPDT4
SCREEN CONTENT
DEC 04, 2003
PAGE 1
DATE CATALOGED 12/04/03 20:55:00
DATE LAST USED
CATALOGED BY
EXPIRATION DATE
DATE LAST UPDATED 12/04/03 20:55:00
LAST UPDATED BY

* SEGMENT 1 *

NUMBER OF SEGMENT OCCURRENCES = 1
NUMBER OF FIELDS IN SEGMENT = 4
POSITION OF FIRST FIELD
LINE: 2 COLUMN: 18

SCREEN	I	R	E	P	M	N	A	C					
FIELD	.SOURCE/DEST.	/	E	H	R	R	O	E/V		OPERAND B.....		PARTIAL FIELD	
NAME	QLF FLD NAME	O	Q	I	R	T	D	M	T	N	OP	QLF FIELD NAME / CONSTANT	STRT NUMB OPR

CUSTNO N,CUSTNO O
N,CUSTNO I H Y N N X
M NM CUSTNO
M IF F,ATTNID EQ C,PF001
M IF F,ATTNID EQ C,ENTER
LOC/XATTRS: LINE 2,COL 18 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL=)
ORDERNO N,ORDERNO O
N,ORDERNO I H Y N N X
Q NM ORDERNO
IF F,ATTNID EQ C,ENTER
LOC/XATTRS: LINE 3,COL 18 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL=)
CUSTNAME N,CUSTNAME O
LOC/XATTRS: LINE 4,COL 18 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL=)
NONAME F,MSGLINE O H
LOC/XATTRS: LINE 24,COL 2 =(COLOR=N,HIGHLIGHT= ,PGM SYMBOL=)

GLOSSARY FOR SCREEN - ORDUPDT4
SCREEN CONTENT
DEC 04, 2003
PAGE 2

* SEGMENT 2 *

NUMBER OF SEGMENT OCCURRENCES = 15
NUMBER OF FIELDS IN SEGMENT = 7
POSITION OF FIRST FIELD
LINE: 9 COLUMN: 3

SCREEN	I	R	E	P	M	N	A	C					
FIELD	.SOURCE/DEST.	/	E	H	R	R	O	E/V		OPERAND B.....		PARTIAL FIELD	
NAME	QLF FLD NAME	O	Q	I	R	T	D	M	T	N	OP	QLF FIELD NAME / CONSTANT	STRT NUMB OPR

ACTION I H N N N
NM ACTION
LOC/XATTRS: LINE 9,COL 3 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)
ITEMORD N,ITEMORD O
N,ITEMORD I H N N X
V NM ITEMORD
IF S,ACTION EQ C,U
X
U IF S,ACTION EQ C,U
LOC/XATTRS: LINE 9,COL 6 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)
ITEMNAME 2,ITEMNAME O
LOC/XATTRS: LINE 9,COL 14 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL=)
ITMPRICE N,ITMPRICE O
N,ITMPRICE I H N N P
NM ITMPRICE
LOC/XATTRS: LINE 9,COL 35 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)
ITMQUYOR N,ITMQUYOR O

Figure A-2 Program Listings for Sample Applications (Page 24 of 84)

```

N,ITMQTYOR I      H      N      N      P      NM      ITMQTYOR
LOC/XATTRS: LINE  9,COL  45 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

QTYBKORD N,QTYBKORD O
N,QTYBKORD I      H      N      N      P      NM      QTYBKORD
LOC/XATTRS: LINE  9,COL  56 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

QANONHND 2,QANONHND O
LOC/XATTRS: LINE  9,COL  67 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

```

** M5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** M5WC03 TYPE 0 SCREEN DEFINITION ORDADD4 CATALOGED PRIOR TO VALIDATION.

SCREEN NAME	STMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

(ORDADD4) (SD) (R) () (M501)

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80

```

(ORDADD4 ,SL) ( 1) (0) ( ORDADD4 )
(ORDADD4 ,SL) ( 1) (1) (ADD NEW DATA=ENTER )
(ORDADD4 ,SL) ( 2) (1) (CUST UPDT SCR=PF1 )
(ORDADD4 ,SL) ( 3) (0) ( CUSTOMER NUMBER )
(ORDADD4 ,SL) ( 3) (1) (VIEW ADDED ITEMS=PF6 )
(ORDADD4 ,SL) ( 4) (0) ( CUSTOMER NAME )
(ORDADD4 ,SL) ( 4) (1) (RETURN TO MENU=PF10 )
(ORDADD4 ,SL) ( 5) (0) ( CUSTOMER PHONE )
(ORDADD4 ,SL) ( 7) (0) ( ORDER NUMBER )
(ORDADD4 ,SL) ( 8) (0) ( P.O. NUMBER )
(ORDADD4 ,SL) ( 10) (0) ( ITEM PRICE )
(ORDADD4 ,SL) ( 11) (0) ( NUMBER ORDERED )
(ORDADD4 ,SL) ( 12) (0) ( )
(ORDADD4 ,SL) ( 13) (0) ( )
(ORDADD4 ,SL) ( 14) (0) ( )
(ORDADD4 ,SL) ( 15) (0) ( )
(ORDADD4 ,SL) ( 16) (0) ( )
(ORDADD4 ,SL) ( 17) (0) ( )
(ORDADD4 ,SL) ( 18) (0) ( )
(ORDADD4 ,SL) ( 19) (0) ( )
(ORDADD4 ,SL) ( 20) (0) ( )
(ORDADD4 ,SL) ( 21) (0) ( )
(ORDADD4 ,SL) ( 22) (0) ( )
(ORDADD4 ,SL) ( 23) (0) ( )

```

** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

(ORDADD4 ,SS) (1) (1)

SCREEN NAME	STMT TYPE	SEG NO.	S A	V QLF	DEST FLD NAME	IR O D	CE H I	PM R R	LN T D M	A T	L V N	C O P	E QLF	OPERAND B..... FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO	SEQUENCE 73 TO 80

```

(ORDADD4 ,S0) ( 1) (N,CUSTNO ) (B) (Y) (X) (NM) (CUSTNO )
(ORDADD4 ,S1) (M) (IF (F,ATTNID ) (EQ) (C,PF001 )
(ORDADD4 ,S1) (M) (IF (F,ATTNID ) (EQ) (C,PF006 )
(ORDADD4 ,S1) (M) (IF (F,ATTNID ) (EQ) (C,ENTER )
(ORDADD4 ,S0) ( 1) (N,CUSTNAME) (O) (NM) (CUSTNAME )
(ORDADD4 ,S0) ( 1) (N,CUSTPH ) (O) (NM) (CUSTPH )
(ORDADD4 ,S0) ( 1) (N,ORDERNO ) (B) (X) (NM) (ORDERNO )
(ORDADD4 ,S1) (M) (IF (F,ATTNID ) (EQ) (C,PF006 )
(ORDADD4 ,S1) (V) (IF (F,ATTNID ) (EQ) (C,ENTER )
(ORDADD4 ,S0) ( 1) (N,ORPERSON) (B) (P) (NM) (ORPERSON )
(ORDADD4 ,S0) ( 1) (N,ORDPONUM) (B) (P) (NM) (ORDPONUM )

```

Figure A-2 Program Listings for Sample Applications (Page 25 of 84)

(ORDADD4 ,SO) (1) (N,ORDDUDAT) (B) (P) (NM) (ORDDUDAT) ()
(ORDADD4 ,SO) (1) (N,ORDRDATE) (B) (P) (NM) (ORDRDATE) ()
(ORDADD4 ,SO) (1) (A) (F,MSGLINE) (H) () (*SYSTEM*)

SCREEN SIMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES 73 TO 80

(ORDADD4 ,SS) (2) (11)

V I R C E P M N A L C PARTIAL
SCREEN SIMT SEG S SOURCE/DEST / Q H U R R O U C L O E / V OPERAND B. FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ORDADD4 ,SO) (2) (N,ITEMORD) (I) (X) (NM) (ITEMORD) ()
(ORDADD4 ,S1) (I) (IF) (F,ATINID) (EQ) (C,ENTER) ()
(ORDADD4 ,SO) (2) (S) (1,ITEMNO) (I) (X)
(ORDADD4 ,S1) (U) (IF) (F,ATINID) (EQ) (C,ENTER) ()
(ORDADD4 ,SO) (2) (N,ITEMPRICE) (I) (R) (NM) (ITEMPRICE) ()
(ORDADD4 ,SO) (2) (N,ITEMQTYOR) (I) (R) (NM) (ITEMQTYOR) ()
(ORDADD4 ,SO) (2) (S) (1,QANONHND) (I) (S)
(ORDADD4 ,S1) (LE) (1,QANONHND) ()
(ORDADD4 ,S1) (MS) (C,NOT ENOUGH IN STOCK) ()
(ORDADD4 ,SO) (2) (N,QTYBKORD) (I) (R) (NM) (QTYBKORD) ()

GLOSSARY FOR SCREEN - ORDADD4
SCREEN FORMAT

DEC 04, 2003
PAGE 1

-----1-----2-----3-----4-----5-----6-----7-----8
1 | ORDADD4 | ADD NEW DATA=ENTER | 1
2 | | CUST UPDT SCR=PF1 | 2
3 | CUSTOMER NUMBER | | VIEW ADDED ITEMS=PF6 | 3
4 | CUSTOMER NAME | | RETURN TO MENU=PF10 | 4
5 | CUSTOMER PHONE | | | 5
6 | | | | 6
7 | ORDER NUMBER | SELLER | | 7
8 | P.O. NUMBER | DUE DATE | ORDER DATE | 8
9 | | | | 9
10 | ITEM | PRICE | QUANTITY | BACK | 10
11 | NUMBER | ORDERED | ORDERED | 11
12 | | | | | 12
13 | | | | | 13
14 | | | | | 14
15 | | | | | 15
16 | | | | | 16
17 | | | | | 17
18 | | | | | 18
19 | | | | | 19
20 | | | | | 20
21 | | | | | 21
22 | | | | | 22
23 | | | | | 23
24 | | | | | 24
-----1-----2-----3-----4-----5-----6-----7-----8

SCREEN IDENTIFIER.. M510 FIELD DESIGNATOR..... INITIAL CURSOR POSITION
TRANSACTION CODE... HIGHLIGHT DESIGNATOR.. @ LINE: 13 COLUMN: 7
OUTPUT CONTROL..... S

GLOSSARY FOR SCREEN - ORDADD4
SCREEN CONTENT

DEC 04, 2003
PAGE 1

DATE CATALOGED 12/04/03 20:55:00
CATALOGED BY

DATE LAST USED
EXPIRATION DATE

DATE LAST UPDATED 12/04/03 20:55:00
LAST UPDATED BY

* SEGMENT 1 *

NUMBER OF SEGMENT OCCURRENCES = 1
NUMBER OF FIELDS IN SEGMENT = 9

POSITION OF FIRST FIELD
LINE: 3 COLUMN: 18

Figure A-2 Program Listings for Sample Applications (Page 26 of 84)

SCREEN		I	R	E	P	M	N		A	C		
FIELD	.SOURCE/DEST.	/	E	H	R	R	O	U	C	O	E/V	OPERAND B.....
NAME	QLF FLD NAME	O	Q	I	R	T	D	M	T	N	OP	QLF FIELD NAME / CONSTANT
												PARTIAL FIELD STRT NUMB OPR

CUSTNO	N,CUSTNO	O										
	N,CUSTNO	I		H	Y	N	N		X			
										NM	CUSTNO	
									M	IF	F,ATTNID	EQ C,PF001
									M	IF	F,ATTNID	EQ C,PF006
									M	IF	F,ATTNID	EQ C,ENTER
	LOC/XATTRS:	LINE	3,	COL	18	=(COLOR=B,HIGHLIGHT= ,PGM SYMBOL=)						
CUSTNAME	N,CUSTNAME	O										
	LOC/XATTRS:	LINE	4,	COL	18	=(COLOR=B,HIGHLIGHT= ,PGM SYMBOL=)						
CUSTPH	N,CUSTPH	O										
	LOC/XATTRS:	LINE	5,	COL	18	=(COLOR=B,HIGHLIGHT= ,PGM SYMBOL=)						
ORDERNO	N,ORDERNO	O										
	N,ORDERNO	I		H		N	N		X			
										NM	ORDERNO	
									M	IF	F,ATTNID	EQ C,PF006
									V	IF	F,ATTNID	EQ C,ENTER
	LOC/XATTRS:	LINE	7,	COL	15	=(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)						
ORPERSON	N,ORPERSON	O										
	N,ORPERSON	I		H		N	N		P			
										NM	ORPERSON	
	LOC/XATTRS:	LINE	7,	COL	28	=(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)						
ORDPONUM	N,ORDPONUM	O										
	N,ORDPONUM	I		H		N	N		P			
										NM	ORDPONUM	
	LOC/XATTRS:	LINE	8,	COL	14	=(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)						
ORDDUDAT	N,ORDDUDAT	O										
	N,ORDDUDAT	I		H		N	N		P			
										NM	ORDDUDAT	
	LOC/XATTRS:	LINE	8,	COL	29	=(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)						
ORDRDATE	N,ORDRDATE	O										
	N,ORDRDATE	I		H		N	N		P			
										NM	ORDRDATE	
	LOC/XATTRS:	LINE	8,	COL	47	=(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)						

GLOSSARY FOR SCREEN - ORDADD4
SCREEN CONTENT
SEGMENT 1 CONTINUED

DEC 04, 2003
PAGE 2

SCREEN		I	R	E	P	M	N		A	C		
FIELD	.SOURCE/DEST.	/	E	H	R	R	O	U	C	O	E/V	OPERAND B.....
NAME	QLF FLD NAME	O	Q	I	R	T	D	M	T	N	OP	QLF FIELD NAME / CONSTANT
												PARTIAL FIELD STRT NUMB OPR

F,MSGLINE	O	H										
LOC/XATTRS:	LINE	24,	COL	2	=(COLOR=N,HIGHLIGHT= ,PGM SYMBOL=)							

GLOSSARY FOR SCREEN - ORDADD4
SCREEN CONTENT

DEC 04, 2003
PAGE 3

* SEGMENT 2 *

NUMBER OF SEGMENT OCCURRENCES = 11
NUMBER OF FIELDS IN SEGMENT = 4

POSITION OF FIRST FIELD
LINE: 13 COLUMN: 7

SCREEN		I	R	E	P	M	N		A	C		
FIELD	.SOURCE/DEST.	/	E	H	R	R	O	U	C	O	E/V	OPERAND B.....
NAME	QLF FLD NAME	O	Q	I	R	T	D	M	T	N	OP	QLF FIELD NAME / CONSTANT
												PARTIAL FIELD STRT NUMB OPR

ITEMORD	N,ITEMORD	I		H		N	N		X			

Figure A-2 Program Listings for Sample Applications (Page 27 of 84)


```

NM      ITEMORD
IF      F,ATTNID  EQ C,ENTER
1,ITEMNO  I      H      N      N      I      X      U      IF      F,ATTNID  EQ C,ENTER
LOC/XATTRS: LINE 13,COL 7 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ITEMPRICE N,ITEMPRICE I      H      N      N      R      NM      ITEMPRICE
LOC/XATTRS: LINE 13,COL 15 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

ITEMQTYOR N,ITEMQTYOR I      H      N      N      R      NM      ITEMQTYOR
1,QANONEND I      H      N      N      S      LE      1,QANONEND
MS      C,NOT ENOUGH IN STOCK
LOC/XATTRS: LINE 13,COL 25 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

QTYEKORD N,QTYEKORD I      H      N      N      R      NM      QTYEKORD
LOC/XATTRS: LINE 13,COL 36 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

```

** M5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** M5WC03 TYPE 0 SCREEN DEFINITION FILEKOD4 CATALOGED PRIOR TO VALIDATION.

SCREEN NAME	SMTI TYPE	CAT ACIN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80
(FILEKOD4)	(SD)	(R)		(_)	(@)		(M508)					

SCREEN NAME	SMTI TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5..... <-TEXT (1-57)	SEQUENCE 73 TO 80
					..6.....7.....8.....9.....0.....1.... <-TEXT (58-114)	

```

(FILEKOD4,SL) ( 1) (0) ( FILEKOD4 DO YOU WANT TO FILL )
(FILEKOD4,SL) ( 1) (1) (FILL BACK-ORDER='Y' OR )
(FILEKOD4,SL) ( 2) (0) ( THIS BACK-ORDER (Y OR N)? _ )
(FILEKOD4,SL) ( 2) (1) ( 'N' + ENTER )
(FILEKOD4,SL) ( 3) (1) (RETURN TO MENU=PF10 )
(FILEKOD4,SL) ( 4) (0) ( )
(FILEKOD4,SL) ( 5) (0) ( )
(FILEKOD4,SL) ( 6) (0) ( )
(FILEKOD4,SL) ( 7) (0) ( )
(FILEKOD4,SL) ( 8) (0) ( )
(FILEKOD4,SL) ( 9) (0) ( )
(FILEKOD4,SL) (10) (0) ( )
(FILEKOD4,SL) (11) (0) ( )
(FILEKOD4,SL) (11) (1) ( )
(FILEKOD4,SL) (12) (0) ( )
(FILEKOD4,SL) (13) (0) ( )
(FILEKOD4,SL) (14) (0) ( )
(FILEKOD4,SL) (15) (0) ( )
(FILEKOD4,SL) (16) (0) ( )
** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

```

SCREEN NAME	SMTI TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80
(FILEKOD4,SS)	(1)	(1)		

SCREEN NAME	SMTI TYPE	SEG NO.	V A	SOURCE/DEST QLF FLD NAME	I R C E P M N / Q H U R R O U O D I R R T D M	A L C C L O E / V T V N O P	OPERAND B..... QLF FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80
(FILEKOD4,S0)	(1)			(I)		(N)	(NM) (CHOICE)		
(FILEKOD4,S0)	(1)			(1,ITEMNO)	(B)	(Y)	(U) (NM) (ITEMNO)		
(FILEKOD4,S0)	(1)			(1,BKORDRNO)	(B)		(D) (NM) (BKORDRNO)		
(FILEKOD4,S1)							(X) (IF (S,CHOICE) (NE) (C,Y))		
(FILEKOD4,S0)	(1)			(1,ITEMNAME)	(O)		(NM) (ITEMNAME)		
(FILEKOD4,S0)	(1)			(I)		(Y)	(N) (NM) (ITEMPRICE)		
(FILEKOD4,S0)	(1)			(N,ITEMPRICE)	(O)				
(FILEKOD4,S0)	(1)			(1,QTYEKORD)	(I)	(Y)	(S) (NM) (QTYEKORD)		
(FILEKOD4,S0)	(1)			(1,QANONEND)	(I)	(Y)	(S)		
(FILEKOD4,S0)	(1)			(N,QTYEKORD)	(O)				

Figure A-2 Program Listings for Sample Applications (Page 28 of 84)

```

(FILEKOD4,S0) ( 1) (1,QANONHND) (O) (NM) (QANONHND) ( )
(FILEKOD4,S0) ( 1) (N,CUSTNO ) (B) (Y) (M) (NM) (CUSTNO) ( )
(FILEKOD4,S0) ( 1) (N,CUSTNAME) (O) (NM) (CUSTNAME) ( )
(FILEKOD4,S0) ( 1) (N,ORDERNO ) (B) (Y) (M) (NM) (ORDERNO) ( )
(FILEKOD4,S0) ( 1) (N,SHIPNO ) (I) (I) (NM) (SHIPNO) ( )
(FILEKOD4,S1) (X) (IF) (S,CHOICE ) (NE) (C,Y) ( )
(FILEKOD4,S0) ( 1) (N,SHIPPER ) (I) (R) (NM) (SHIPPER) ( )
(FILEKOD4,S0) ( 1) (N,INVNO ) (I) (R) (NM) (INVNO) ( )
(FILEKOD4,S0) ( 1) (N,PAYDATE ) (I) (R) (NM) (PAYDATE) ( )
(FILEKOD4,S0) ( 1) (A) (F,MSGLINE ) (H) ( ) ( ) (*SYSTEM*)

```

GLOSSARY FOR SCREEN - FILEKOD4
SCREEN FORMAT

DEC 04, 2003
PAGE 1

```

*-----1-----2-----3-----4-----5-----6-----7-----8*
1 | FILEKOD4 DO YOU WANT TO FILL FILL BACK-ORDER='Y' OR 1
2 | THIS BACK-ORDER (Y OR N)? _ 'N' + ENTER 2
3 | RETURN TO MENU=PF10 3
4 | ITEM NUMBER 4
5 | BACK-ORDER NUMBER 5
6 | ITEM NAME 6
7 | PRICE 7
8 | QUANTITY BACK-ORDERED 8
9 | QUANTITY ON HAND 9
10 | CUSTOMER NUMBER 10
11 | CUSTOMER NAME 11
12 | ORDER NUMBER 12
13 | SHIPMENT NUMBER 13
14 | SHIPPER 14
15 | INVOICE NUMBER 15
16 | DUE DATE 16
17 | 17
18 | 18
19 | 19
20 | 20
21 | 21
22 | 22
23 | 23
24 | 24

```

-----1-----2-----3-----4-----5-----6-----7-----8

SCREEN IDENTIFIER.. M510 FIELD DESIGNATOR..... INITIAL CURSOR POSITION
TRANSACTION CODE... HIGHLIGHT DESIGNATOR.. @ LINE: 2 COLUMN: 42
OUTPUT CONTROL..... S

GLOSSARY FOR SCREEN - FILEKOD4
SCREEN CONTENT

DEC 04, 2003
PAGE 1

DATE CATALOGED 12/04/03 20:55:00 DATE LAST USED DATE LAST UPDATED 12/04/03 20:55:00
CATALOGED BY EXPIRATION DATE LAST UPDATED BY

* SEGMENT 1 *

NUMBER OF SEGMENT OCCURRENCES = 1 POSITION OF FIRST FIELD
NUMBER OF FIELDS IN SEGMENT = 15 LINE: 2 COLUMN: 42

SCREEN	I	R	E	P	M	N	A	C	
FIELD	.SOURCE/DEST.	/	E	H	R	R	O	U	
NAME	QLF FLD NAME	O	Q	I	R	T	D	M	

CHOICE I H N N N NM CHOICE
LOC/XATTRS: LINE 2,COL 42 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)

ITEMNO 1,ITEMNO O
1,ITEMNO I H Y N N U NM ITEMNO
LOC/XATTRS: LINE 4,COL 35 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL=)

BKORDRNO 1,BKORDRNO O
1,BKORDRNO I H N N D NM BKORDRNO
X IF S,CHOICE NE C,Y
LOC/XATTRS: LINE 5,COL 35 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL=)

Figure A-2 Program Listings for Sample Applications (Page 29 of 84)

```

ITEMNAME 1,ITEMNAME O
LOC/XATTRS: LINE 6,COL 35 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

ITMPRICE N,ITMPRICE O
I H Y N N N NM ITMPRICE
LOC/XATTRS: LINE 7,COL 35 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

QTYBKORD N,QTYBKORD O
1,QTYBKORD I H Y N N S NM QTYBKORD
1,QANONHND I H Y N N S NM QTYBKORD
LOC/XATTRS: LINE 8,COL 35 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

QANONHND 1,QANONHND O
LOC/XATTRS: LINE 9,COL 35 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

CUSTNO N,CUSTNO O
N,CUSTNO I H Y N N M NM CUSTNO
LOC/XATTRS: LINE 10,COL 35 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

CUSTNAME N,CUSTNAME O
LOC/XATTRS: LINE 11,COL 35 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

ORDERNO N,ORDERNO O

```

GLOSSARY FOR SCREEN - FILBKOD4
SCREEN CONTENT
SEGMENT 1 CONTINUED

DEC 04, 2003
PAGE 2

SCREEN FIELD NAME	.SOURCE/DEST. QLF FLD NAME	I O	R Q	E I	P R	M R	N O	A C	C O	E/V N	OP	OPERAND B..... QLF FIELD NAME / CONSTANT	PARTIAL FIELD STRT NUMB OPR

```

ORDERNO N,ORDERNO I H Y N N M NM ORDERNO
LOC/XATTRS: LINE 12,COL 35 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )

SHIPNO N,SHIPNO I H N N I NM SHIPNO
X IF S,CHOICE NE C,Y
LOC/XATTRS: LINE 13,COL 35 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

SHIPPER N,SHIPPER I H N N R NM SHIPPER
LOC/XATTRS: LINE 14,COL 35 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

INVNO N,INVNO I H N N R NM INVNO
LOC/XATTRS: LINE 15,COL 35 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

PAYDATE N,PAYDATE I H N N R NM PAYDATE
LOC/XATTRS: LINE 16,COL 35 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )

*NONAME* F,MSGLINE O H
LOC/XATTRS: LINE 24,COL 2 =(COLOR=N,HIGHLIGHT= ,PGM SYMBOL= )

```

** MK5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** MK5WC03 TYPE 0 SCREEN DEFINITION ADDEKOD4 CATALOGED PRIOR TO VALIDATION.

SCREEN NAME	STMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80
(ADDEKOD4)	(SD)	(R)		()	(@)		(M509)					

SCREEN NAME	STMT TYPE	LINE NO.	CONT- LINE	EXTN ATTR	1.....2.....3.....4.....5..... ..6.....7.....8.....9.....0.....1.....	<-TEXT (1-57) <-TEXT (58-114)	SEQUENCE 73 TO 80

Figure A-2 Program Listings for Sample Applications (Page 30 of 84)

```

-----
(ADDEKOD4,SL) ( 1) (0) ( ADDEKOD4 )
(ADDEKOD4,SL) ( 5) (0) ( )
(ADDEKOD4,SL) ( 7) (0) ( )
(ADDEKOD4,SL) ( 9) (0) ( )
(ADDEKOD4,SL) (11) (0) ( )
(ADDEKOD4,SL) (13) (0) ( )
(ADDEKOD4,SL) (17) (0) ( )
(ADDEKOD4,SL) (18) (0) ( )
(ADDEKOD4,SL) (19) (0) ( )
(ADDEKOD4,SL) (20) (0) ( )
** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----

```

SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

```

-----
(ADDEKOD4,SS) ( 1) ( 1)
-----

```

SCREEN NAME	STMT TYPE	SEG NO.	V A	SOURCE/DEST QLF FLD NAME	I R C E P M N / Q H U R R O U O D I R R T D M	A L C C L O E / V T V N O P	OPERAND B. QLF FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80
(ADDEKOD4,S0)	(1)	(1,CUSTNO)	(I)			(P)	(NM) (CUSTNO)		
(ADDEKOD4,S1)							(NE) (C)		
(ADDEKOD4,S1)							(MS) (C,CUSTOMER NUMBER REQUIRED)		
(ADDEKOD4,S0)	(1)	(S) (N,CUSTNO)	(I)			(X)			
(ADDEKOD4,S1)						(M)	(IF) (F,ATTNID) (EQ) (C,ENTER)		
(ADDEKOD4,S0)	(1)	(1,ORDERNO)	(I)			(P)	(NM) (ORDERNO)		
(ADDEKOD4,S1)							(NE) (C)		
(ADDEKOD4,S1)							(MS) (C,ORDER NUMBER REQUIRED)		
(ADDEKOD4,S0)	(1)	(S) (N,ORDERNO)	(I)			(X)			
(ADDEKOD4,S1)						(M)	(IF) (F,ATTNID) (EQ) (C,ENTER)		
(ADDEKOD4,S0)	(1)	(1,ITEMNO)	(B)		(Y)	(X)	(NM) (ITEMNO)		
(ADDEKOD4,S1)						(U)	(IF) (F,ATTNID) (EQ) (C,ENTER)		
(ADDEKOD4,S0)	(1)	(S) (N,ITEMORD)	(I)		(Y)	(X)			
(ADDEKOD4,S1)						(M)	(IF) (F,ATTNID) (EQ) (C,ENTER)		
(ADDEKOD4,S0)	(1)	(1,BKORDRNO)	(I)			(X)	(NM) (BKORDRNO)		
(ADDEKOD4,S1)						(I)	(IF) (F,ATTNID) (EQ) (C,ENTER)		
(ADDEKOD4,S0)	(1)	(N,QTYBKORD)	(I)			(A)	(NM) (QTYBKORD)		
(ADDEKOD4,S1)							(NE) (C)		
(ADDEKOD4,S1)							(A,BQ) (QTYBKORD)		
(ADDEKOD4,S1)							(MS) (C,QUANTITY BACK-ORDERED IN)		
(ADDEKOD4,S1)							(MS) (C,CORRECT - REENTER)		
(ADDEKOD4,S0)	(1)	(A) (F,MSGLINE)	(H)						(*SYSTEM*)

GLOSSARY FOR SCREEN - ADDEKOD4
SCREEN FORMAT

DEC 04, 2003
PAGE 1

```

*-----1-----2-----3-----4-----5-----6-----7-----8*
1 | ADDEKOD4 | 1
2 | | 2
3 | | 3
4 | | 4
5 | | 5
6 | | 6
7 | | 7
8 | | 8
9 | | 9
10 | | 10
11 | | 11
12 | | 12
13 | | 13
14 | | 14
15 | | 15
16 | | 16
17 | | 17
18 | | 18
19 | | 19
20 | | 20
21 | | 21
22 | | 22
23 | | 23
24 | | 24
*-----1-----2-----3-----4-----5-----6-----7-----8*

```

SCREEN IDENTIFIER.. M510
TRANSACTION CODE...

FIELD DESIGNATOR.....
HIGHLIGHT DESIGNATOR.. @

INITIAL CURSOR POSITION
LINE: 5 COLUMN: 45

Figure A-2 Program Listings for Sample Applications (Page 31 of 84)

```

                                OUTPUT CONTROL..... S

                                GLOSSARY FOR SCREEN - ADDBKOD4
                                SCREEN CONTENT
                                DEC 04, 2003
                                PAGE 1

DATE CATALOGED 12/04/03 20:55:00      DATE LAST USED      DATE LAST UPDATED 12/04/03 20:55:00
CATALOGED BY      EXPIRATION DATE      LAST UPDATED BY

                                *****
                                * SEGMENT 1 *
                                *****

NUMBER OF SEGMENT OCCURRENCES = 1      POSITION OF FIRST FIELD
NUMBER OF FIELDS IN SEGMENT = 6      LINE: 5 COLUMN: 45
-----
SCREEN      I R E P M N      A C
FIELD      .SOURCE/DEST. / E H R R O U      C O E/V .....OPERAND B.....
NAME      QLF FLD NAME O Q I R T D M      T N OP QLF FIELD NAME / CONSTANT      PARTIAL FIELD
-----
CUSTNO 1,CUSTNO I H N N P NM CUSTNO
NE C,
MS C,CUSTOMER NUMBER REQUIRED
N,CUSTNO I H N N X M IF F,ATTNID EQ C,ENTER
LOC/XATTRS: LINE 5,COL 45 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )
ORDERNO 1,ORDERNO I H N N P NM ORDERNO
NE C,
MS C,ORDER NUMBER REQUIRED
N,ORDERNO I H N N X M IF F,ATTNID EQ C,ENTER
LOC/XATTRS: LINE 7,COL 45 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )
ITEMNO 1,ITEMNO O H Y N N X NM ITEMNO
U IF F,ATTNID EQ C,ENTER
N,ITEMORD I H Y N N X M IF F,ATTNID EQ C,ENTER
LOC/XATTRS: LINE 9,COL 45 =(COLOR=B,HIGHLIGHT= ,PGM SYMBOL= )
BKORDRNO 1,BKORDRNO I H N N X NM BKORDRNO
I IF F,ATTNID EQ C,ENTER
LOC/XATTRS: LINE 11,COL 45 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )
QTYBKORD N,QTYBKORD I H N N A NM QTYBKORD
NE C,
A EQ QTYBKORD
MS C,QUANTITY BACK-ORDERED IN
MS C,CORRECT - REENTER
LOC/XATTRS: LINE 13,COL 45 =(COLOR=G,HIGHLIGHT= ,PGM SYMBOL= )
*NONAME* F,MSGLINE O H
LOC/XATTRS: LINE 24,COL 2 =(COLOR=N,HIGHLIGHT= ,PGM SYMBOL= )

** MK5WZ01 TYPE 0 END OF SCREEN DEFINITION GLOSSARY.

** MK5WZ33 TYPE 0 THE FOLLOWING CICS TS SYSTEM DEFINITION UTILITY INPUT STATEMENTS ARE REQUIRED FOR THIS RMS MAPSET.
DEFINE GROUP(???)MAPSET(M510)
** MK5WZ04 TYPE 0 NUMBER OF MESSAGES PRINTED IS 90.
MSGINPUT - 411 RECORDS INPUT
MSGPUNCH - 1288 RECORDS OUTPUT
MSLIST - 1746 RECORDS OUTPUT
5 TRACKS ASSIGNED TO M5LIB - 3 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 272.
** MK5WZ09 TYPE 0 17928 BYTES OF TYPE A STORAGE USED DURING DICTIONARY MAINTENANCE
DEC 05, 2003 13.18.08
*****
* Advantage VISION:Transact for z/OS - Version 7.8 *

```

Figure A-2 Program Listings for Sample Applications (Page 32 of 84)

Advantage VISION:Transact APARs and PTFs Applied

NONE.

Common Library Subsystem Information

COMLIB Version C4.5 LIBRARY ACCESS METHOD = B/V RESERVE = NO ITEM TRACKING = 0

COMLIB System Modifications Applied

201,202,203,204,205,206,207,208,209,210,212,213,214,215,216,218

```
=====
=====
M5PARM INSTALLATION PARAMETERS (FIVEPARM)
=====
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
```

```

; WHEN APPROPRIATE.
;
LET S.PFKEY = F.ATTNID
IF F.ATTNID EQ 'PF002'
    LET T.SYNCFLD1 = S.ITEMNO
    LET PF(T.SYNCFLD2,1,1) = S.ITEMNO
    LET PF(T.SYNCFLD2,2,6) = '999999'
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST MENUPFCK.

***** BEGIN ASL PROCEDURE SCRNCNT .
***** Using ASL Translator Version 4.50 *****
; THIS REQUEST CALCULATES THE NUMBER OF SCREENS OF BACK-
; ORDERS THAT ARE POSSIBLE, SHOULD THE TERMINAL OPERATOR
; PRESS PF3.
;
IF F.ATTNID EQ 'PF003'
    LET T.SCREENNO = 1
    LET T.NOSCREEN = 1
    IF 1.BKORDRNO EQ 1.BKORDRNO ;BEGIN LOOP FOR ALL BACKORDERS
        LET T.COUNTOCC = T.COUNTOCC + 1
        IF T.COUNTOCC GT 8
            LET T.NOSCREEN = T.NOSCREEN + 1
            LET T.COUNTOCC = 1
        END
    END
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST SCRNCNT .

***** BEGIN ASL PROCEDURE TEMPS .
***** Using ASL Translator Version 4.50 *****
; TEMP FIELD DEFINITIONS
SYNCFD1: FIELD C 7
SYNCFD2: FIELD C 7
SCREENNO: FIELD Z 2
NOSCREEN: FIELD Z 2
MSG: FIELD C 74
SCRNSLCT: FIELD C 1
UPDATE: FIELD C 1 INIT 'U'
BACKORD: FIELD C 1 INIT 'B'
MENU: FIELD C 1 INIT 'M'
ERROR: FIELD C 1 INIT 'E'
FIRSTOCC: FIELD P 2
LASTOCC: FIELD P 2
COUNTX: FIELD P 2
QANONHND: FIELD C 7
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST TEMPS .

```

DEC 05, 2003 13.18.08

```

*****
* REQUEST NAME - TEMPS *
* INPUT STREAM REQUEST *
*****

```

STMT CAT	REQ	SCREEN	BACK	BR	RNT	LINE
TYPE ACT REQUEST DESCRIPTION	TYP	NAME	CONTROL	TMP		NUMBER
(ER)						(N)
COMMENT ***** ; TEMP FIELD DEFINITIONS *****						*****
COMMENT ***** SYNCFD1: FIELD C 7 *****						*****
COMMENT ***** SYNCFD2: FIELD C 7 *****						*****
COMMENT ***** SCREENNO: FIELD Z 2 *****						*****
COMMENT ***** NOSCREEN: FIELD Z 2 *****						*****
COMMENT ***** MSG: FIELD C 74 *****						*****
COMMENT ***** SCRNSLCT: FIELD C 1 *****						*****
COMMENT ***** UPDATE: FIELD C 1 INIT 'U' *****						*****
COMMENT ***** BACKORD: FIELD C 1 INIT 'B' *****						*****
COMMENT ***** MENU: FIELD C 1 INIT 'M' *****						*****
COMMENT ***** ERROR: FIELD C 1 INIT 'E' *****						*****
COMMENT ***** FIRSTOCC: FIELD P 2 *****						*****
COMMENT ***** LASTOCC: FIELD P 2 *****						*****
COMMENT ***** COUNTX: FIELD P 2 *****						*****
COMMENT ***** QANONHND: FIELD C 7 *****						*****
COMMENT *****						*****

STMT	FIELD	FIELD	FLD	DEC	OUTPT	EDIT	INITIAL	SEQUENCE
TYPE	NAME	LNTH	TYP	PLC	EDIT	LGTH	VALUE	NUMBER
							COMMENTS	

Figure A-2 Program Listings for Sample Applications (Page 34 of 84)

```

(TF) (SYNCFD1) ( 7) (C)
(TF) (SYNCFD2) ( 7) (C)
(TF) (SCREENNO) ( 2) (Z)
(TF) (NOSCREEN) ( 2) (Z)
(TF) (MSG ) ( 74) (C)
(TF) (SCRNSLCT) ( 1) (C)
(TF) (UPDATE ) ( 1) (C)
(TF) (BACKORD ) ( 1) (C)
(TF) (MENU ) ( 1) (C)
(TF) (ERROR ) ( 1) (C)
(TF) (FIRSTOCC) ( 2) (P)
(TF) (LASTOCC ) ( 2) (P)
(TF) (COUNTX ) ( 2) (P)
(TF) (QANONEND) ( 7) (C)

```

```

(U )
(B )
(M )
(E )

```

DEC 05, 2003 13.18.08

```

*****
* SCREEN CONTROL *
*****

```

INPUT SCREEN	STMT TYPE	SEQ NO.	LOG LVL	CON CTR	OPERAND A.. QLF FIELD	OPER ATTN	OPERAND B..... QLF FIELD NAME/CONSTANT	OUTPUT SCREEN	.OPERAND C.. QLF FLD/CONST	PARTL FLD STR NM OP	SEQUENCE 73 TO 80

(ORDMENU4) (SC)						(XR) (1,MENUPFCK	)				
(ORDMENU4) (SC)						(XR) (2,SCRNCNT	)				
(ORDMENU4) (SC)					(F,ATTNID)	(EQ) (C,PF001	)				
(ORDMENU4) (SC)						(OS)			(CUSTUPD4)		
(ORDMENU4) (SC)				(E)	(F,ATTNID)	(EQ) (C,PF002	)				
(ORDMENU4) (SC)						(OS)			(ITMERWS4)		
(ORDMENU4) (SC)				(E)	(F,ATTNID)	(EQ) (C,PF003	)				
(ORDMENU4) (SC)						(OS)			(ITMEROD4)		
(ORDMENU4) (SC)				(E)	(F,ATTNID)	(EQ) (C,PF004	)				
(ORDMENU4) (SC)						(OS)			(ITMUPDT4)		
(ORDMENU4) (SC)				(E)	(F,ATTNID)	(EQ) (C,PF005	)				
(ORDMENU4) (SC)						(OS)			(ENDSCRN4)		

SCREEN NAME	STMT TYPE	CAT ACIN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

(ORDMENU4) (SD)	(R)			()	(@)			(M510)	(M510)			

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80

(ORDMENU4,SL) (1)	(0)			(ORDMENU4	)		
(ORDMENU4,SL) (3)	(0)			(CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS:)			
(ORDMENU4,SL) (5)	(0)			(PF1---CUSTOMER ADD/UPDATE	)		
(ORDMENU4,SL) (7)	(0)			(PF2---ITEM BROWSE BY PRODUCT KEY	)		
(ORDMENU4,SL) (9)	(0)			(PF3---BACK-ORDERED ITEM	)		
(ORDMENU4,SL) (11)	(0)			(PF4---ITEM ADD/UPDATE	)		
(ORDMENU4,SL) (13)	(0)			(PF5---END TRANSACTION	)		
(ORDMENU4,SL) (16)	(0)			(ENTER CUSTOMER OR ITEM NUMBER	)		
(ORDMENU4,SL) (18)	(0)			((ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2)	)		
(ORDMENU4,SL) (18)	(1)			(IS USED)	)		

** MSGW14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

(ORDMENU4,SS)	(1)		(1)	

SCREEN NAME	STMT TYPE	SEG NO.	S QLF FLD NAME	I R C E P M N / Q H U R R O U O D I R R T D M	A L C C L O E / V T V N O P	OPERAND B..... QLF FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80

(ORDMENU4,S0) (1)			(N,CUSTNO)	(I)	(X)	(NM) (ITEMNO	)	
(ORDMENU4,S1)					(V)	(IF) (S,PFKEY)	(EQ) (C,PF001	)
(ORDMENU4,S0) (1) (S)			(1,ITEMNO)	(I)	(X)			
(ORDMENU4,S1)					(M)	(IF) (S,PFKEY)	(EQ) (C,PF003	)

Figure A-2 Program Listings for Sample Applications (Page 35 of 84)


```

(CUSTUPD4,S0) ( 2)          (I)          (N) (NM) (ACTION          )
(CUSTUPD4,S0) ( 2) (N,ORDERNO) (B)      (X) (NM) (ORDERNO      )
(CUSTUPD4,S1)          (U)          (U) (IF) (S,ACTION ) (EQ) (C,U      )
(CUSTUPD4,S1)          (M)          (M) (IF) (S,ACTION ) (EQ) (C,X      )
(CUSTUPD4,S0) ( 2) (N,ORDRDATE) (B)      (P) (NM) (ORDRDATE      )
(CUSTUPD4,S0) ( 2) (N,ORPERSON) (B)      (P) (NM) (ORPERSON      )
(CUSTUPD4,S0) ( 2) (N,ORDPONUM) (B)      (P) (NM) (ORDPONUM      )
(CUSTUPD4,S0) ( 2) (N,ORDUDAT) (B)      (P) (NM) (ORDUDAT       )
(CUSTUPD4,S0) ( 2) (N,ORDINWGN) (B)      (P) (NM) (ORDINWGN      )
(CUSTUPD4,S0) ( 2) (N,ORDCMPLT) (B)      (P) (NM) (ORDCMPLT      )

```

SCREEN NAME	STMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

```

(ITMERWS4) (SD) (R)          ( ) (E)          (M502 ) (M502MAP)

```

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80

```

(ITMERWS4,SL) ( 1) (0)          ( ITMERWS4          NEXT SCREEN OF ITEMS=PA1 )
(ITMERWS4,SL) ( 2) (0)          ( 'U' ACTION=ITEM UPDATE )
(ITMERWS4,SL) ( 3) (0)          ( 'B' ACTION=BACK-ORDERS )
(ITMERWS4,SL) ( 4) (0)          ( 'M' ACTION=MENU )
(ITMERWS4,SL) ( 5) (0)          ( PRODUCT RANGE: _____ TO _____ )
(ITMERWS4,SL) ( 5) (1)          ( QUANTITY )
(ITMERWS4,SL) ( 6) (0)          ( BACK ACT ITEM ITEM QUANTITY SHIPPED ORDERED )
(ITMERWS4,SL) ( 6) (1)          ( )
(ITMERWS4,SL) ( 7) (0)          ( ION NUMBER PRICE ON-HAND YTD YTD )
(ITMERWS4,SL) ( 7) (1)          ( ORDERED )
(ITMERWS4,SL) ( 9) (0)          ( )
(ITMERWS4,SL) ( 9) (1)          ( )
(ITMERWS4,SL) ( 10) (0)         ( )
(ITMERWS4,SL) ( 10) (1)         ( )
(ITMERWS4,SL) ( 11) (0)         ( )
(ITMERWS4,SL) ( 11) (1)         ( )
(ITMERWS4,SL) ( 12) (0)         ( )
(ITMERWS4,SL) ( 12) (1)         ( )
(ITMERWS4,SL) ( 13) (0)         ( )
(ITMERWS4,SL) ( 13) (1)         ( )
(ITMERWS4,SL) ( 14) (0)         ( )
(ITMERWS4,SL) ( 14) (1)         ( )
(ITMERWS4,SL) ( 15) (0)         ( )
(ITMERWS4,SL) ( 15) (1)         ( )
(ITMERWS4,SL) ( 16) (0)         ( )
(ITMERWS4,SL) ( 16) (1)         ( )
(ITMERWS4,SL) ( 17) (0)         ( )
(ITMERWS4,SL) ( 17) (1)         ( )
(ITMERWS4,SL) ( 18) (0)         ( )
(ITMERWS4,SL) ( 18) (1)         ( )
(ITMERWS4,SL) ( 19) (0)         ( )
(ITMERWS4,SL) ( 19) (1)         ( )
(ITMERWS4,SL) ( 20) (0)         ( )
(ITMERWS4,SL) ( 20) (1)         ( )
(ITMERWS4,SL) ( 21) (0)         ( )
(ITMERWS4,SL) ( 21) (1)         ( )
(ITMERWS4,SL) ( 22) (0)         ( )
(ITMERWS4,SL) ( 22) (1)         ( )
(ITMERWS4,SL) ( 23) (0)         ( )
(ITMERWS4,SL) ( 23) (1)         ( )

```

** M502L14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

```

(ITMERWS4,SS) ( 1)          ( 1)

```

SCREEN NAME	STMT TYPE	SEG NO.	S A	V QLF	SOURCE/DEST FLD NAME	I O	R D	C I	E R	P R	M T	N D	A T	L V	C N	OPERAND B.....	PARTIAL FIELD ST NO	SEQUENCE 73 TO 80

```

(ITMERWS4,S0) ( 1) (T,SINCF1D1) (B)          (P) (NM) (SINCF1D1      )
(ITMERWS4,S0) ( 1) (T,SINCF1D2) (B)          (P) (NM) (SINCF1D2      )
(ITMERWS4,S0) ( 1) (A) (F,MSGLINE ) (H)          (*SYSTEM*)

```

Figure A-2 Program Listings for Sample Applications (Page 37 of 84)

SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES												SEQUENCE 73 TO 80
(ITMERWS4,SS) (2) (15)															
SCREEN NAME	STMT TYPE	SEG S	V NO. A	IR SOURCE/DEST QLF FLD NAME	CE / Q HURROU O D I R R T D M	PMN A L C C L O E/V	OPERAND B.....				PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80			
(ITMERWS4,S0)	(2)			(I)	(Y)	(N)	(NM)	(ACTION					)		
(ITMERWS4,S0)	(2)		(1,ITEMNO	) (I)		(X)	(NM)	(ITEMNO					)		
(ITMERWS4,S1)						(M)	(IF)	(S,ACTION	) (EQ)	(C,U			)		
(ITMERWS4,S1)						(M)	(IF)	(S,ACTION	) (EQ)	(C,M			)		
(ITMERWS4,S0)	(2)	(S)	(2,ITEMNO	) (O)											
(ITMERWS4,S0)	(2)		(2,ITMPRICE)	(O)			(NM)	(ITMPRICE					)		
(ITMERWS4,S0)	(2)		(2,QANONHND)	(O)			(NM)	(QANONHND					)		
(ITMERWS4,S0)	(2)		(2,SHIPYTD	) (O)			(NM)	(SHIPYTD					)		
(ITMERWS4,S0)	(2)		(2,ORDYTD	) (O)			(NM)	(ORDYTD					)		
(ITMERWS4,S0)	(2)		(2,QTYBKORD)	(O)			(NM)	(QTYBKORD					)		
SCREEN NAME	STMT TYPE	CAT ACTN TYPE	GLOSS DESIG	FIELD DESIG	HILITE	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80			
(ITMEKOD4) (SD) (R) () (0) (M503) (ITMEKOD)															
SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....+.. <-TEXT (1-57)							SEQUENCE 73 TO 80			
.....6.....7.....8.....9.....0.....+.....1..... <-TEXT (58-114)															
(ITMEKOD4,SL)	(1)	(0)			(ITMEKOD4								)		
(ITMEKOD4,SL)	(3)	(0)			(	ITEM NUMBER		PRICE		QUANTITY			)		
(ITMEKOD4,SL)	(3)	(1)			(/BOX								)		
(ITMEKOD4,SL)	(4)	(0)			(	QUANTITY ON HAND		SHIPPED YTD					)		
(ITMEKOD4,SL)	(4)	(1)			(								)		
(ITMEKOD4,SL)	(5)	(0)			(	ORDERED YTD		QUANTITY BACK-ORDERED					)		
(ITMEKOD4,SL)	(5)	(1)			(								)		
(ITMEKOD4,SL)	(6)	(0)			(	REORDER POINT		PRODUCT CODE					)		
(ITMEKOD4,SL)	(7)	(0)			(	*****							)		
(ITMEKOD4,SL)	(7)	(1)			(*****								)		
(ITMEKOD4,SL)	(8)	(0)			(	* SPECIFIED SCREEN OF BACK-ORDERS=SCREEN#	+)						)		
(ITMEKOD4,SL)	(8)	(1)			(ENTER *								)		
(ITMEKOD4,SL)	(9)	(0)			(	* NEXT SCREEN OF BACK-ORDERS=ENTER							)		
(ITMEKOD4,SL)	(9)	(1)			(*								)		
(ITMEKOD4,SL)	(10)	(0)			(	* FILL A BACK-ORDER='X' + PF8							)		
(ITMEKOD4,SL)	(10)	(1)			(*								)		
(ITMEKOD4,SL)	(11)	(0)			(	* ADD BACK-ORDER=PF9 RETURN TO MENU=PF10							)		
(ITMEKOD4,SL)	(11)	(1)			(*								)		
(ITMEKOD4,SL)	(12)	(0)			(	*****							)		
(ITMEKOD4,SL)	(12)	(1)			(*****								)		
(ITMEKOD4,SL)	(13)	(0)			(	FILL BACK CUSTOMER ORDER							)		
(ITMEKOD4,SL)	(13)	(1)			(	SCREEN							)		
(ITMEKOD4,SL)	(14)	(0)			(	BACK ORDER NUMBER NUMBER							)		
(ITMEKOD4,SL)	(14)	(1)			(	OF							)		
(ITMEKOD4,SL)	(15)	(0)			(	ORDERS NUMBER							)		
(ITMEKOD4,SL)	(16)	(0)			(								)		
(ITMEKOD4,SL)	(17)	(0)			(								)		
(ITMEKOD4,SL)	(18)	(0)			(								)		
(ITMEKOD4,SL)	(19)	(0)			(								)		
(ITMEKOD4,SL)	(20)	(0)			(								)		
(ITMEKOD4,SL)	(21)	(0)			(								)		
(ITMEKOD4,SL)	(22)	(0)			(								)		
(ITMEKOD4,SL)	(23)	(0)			(								)		
** M5W14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.															
SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES												SEQUENCE 73 TO 80
(ITMEKOD4,SS) (1) (1)															
SCREEN NAME	STMT TYPE	SEG S	V NO. A	IR SOURCE/DEST QLF FLD NAME	CE / Q HURROU O D I R R T D M	PMN A L C C L O E/V	OPERAND B.....				PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80			

Figure A-2 Program Listings for Sample Applications (Page 38 of 84)

NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D	I	R	R	T	D	M	T	V	N	O	P	QLF	FIELD	NAME	OR	CONSTANT	ST	NO	OP	73	TO	80
(ITMEKOD4,S0)	(	1)		(1,ITEMNO	)	(B)								(M)		(NM)	(ITEMNO				(IF	(F,ATTNID	)	(EQ)	(C,PF010					
(ITMEKOD4,S1)														(X)			(IF	(F,ATTNID	)	(EQ)	(C,PF010									
(ITMEKOD4,S0)	(	1)		(1,ITEMPRICE	)	(O)											(NM)	(ITEMPRICE												
(ITMEKOD4,S0)	(	1)		(1,PACKBOX	)	(O)											(NM)	(PACKBOX												
(ITMEKOD4,S0)	(	1)		(1,QANONHND	)	(O)											(NM)	(QANONHND												
(ITMEKOD4,S0)	(	1)		(1,SHIPYTD	)	(O)											(NM)	(SHIPYTD												
(ITMEKOD4,S0)	(	1)		(1,ORDYTD	)	(O)											(NM)	(ORDYTD												
(ITMEKOD4,S0)	(	1)		(1,QTYEKORD	)	(O)											(NM)	(QTYEKORD												
(ITMEKOD4,S0)	(	1)		(1,REORDPT	)	(O)											(NM)	(REORDPT												
(ITMEKOD4,S0)	(	1)		(1,PRODCODE	)	(O)											(NM)	(PRODCODE												
(ITMEKOD4,S0)	(	1)				(I)									(N)		(NM)	(REQSCRN												
(ITMEKOD4,S0)	(	1)		(T,SCREENNO	)	(B)								(Y)		(P)	(NM)	(SCREENNO												
(ITMEKOD4,S0)	(	1)		(T,NOSCREEN	)	(B)								(Y)		(P)	(NM)	(NOSCREEN												
(ITMEKOD4,S0)	(	1)	(A)	(F,MSGLINE	)									(H)															(*SYSTEM*)	
SCREEN	SIMT	SEGMENT	NO.	OF																									SEQUENCE	
NAME	TYPE			OCCURRENCES																									73 TO 80	
(ITMEKOD4,SS)	(	2)		(	8)																									
SCREEN	SIMT	SEG	S	SOURCE/DEST				I	R	C	E	P	M	N	A	L	C												PARTIAL	
NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D	I	R	R	T	D	M	T	V	N	O	P	QLF	FIELD	NAME	OR	CONSTANT					FIELD	SEQUENCE
(ITMEKOD4,S0)	(	2)				(I)								(N)		(NM)	(XFIELD													
(ITMEKOD4,S0)	(	2)		(1,EKORDRNO	)	(B)								(X)		(NM)	(EKORDRNO													
(ITMEKOD4,S1)														(M)		(IF	(S,XFIELD	)	(EQ)	(C,X										
(ITMEKOD4,S0)	(	2)		(N,CUSTNO	)	(I)								(X)		(NM)	(CUSTNO													
(ITMEKOD4,S1)														(M)		(IF	(S,XFIELD	)	(EQ)	(C,X										
(ITMEKOD4,S0)	(	2)	(S)	(1,CUSTNO	)	(O)																								
(ITMEKOD4,S0)	(	2)		(N,ORDERNO	)	(I)								(X)		(NM)	(ORDERNO													
(ITMEKOD4,S1)														(M)		(IF	(S,XFIELD	)	(EQ)	(C,X										
(ITMEKOD4,S0)	(	2)	(S)	(1,ORDERNO	)	(O)																								
(ITMEKOD4,S0)	(	2)	(V)	(N,ITEMORD	)	(I)								(X)		(NM)	(ITEMORD													
(ITMEKOD4,S1)														(M)		(IF	(S,XFIELD	)	(EQ)	(C,X										
SCREEN	SIMT	CAT	GLOSS	FIELD	HILITE	AUTO	OUTPUT	TRANSACT	SCREEN	PROC	TERMINAL																		SEQUENCE	
NAME	TYPE	ACTN	TYPE	DESIG	DESIG	FORMAT	CONTROL	CODE	ID	NULL	DEVICE																		73 TO 80	
(ITMUPDT4)	(SD)	(R)		(	)	(@)										(M504	)	(ITMUPDT)												
SCREEN	SIMT	LINE	CONT-	EXTN	+...1....+...2....+...3....+...4....+...5....+..																								<-TEXT (1-57)	SEQUENCE
NAME	TYPE	NO.	INUE	ATTR	..6....+...7....+...8....+...9....+...0....+...1....																								<-TEXT (58-114)	73 TO 80
(ITMUPDT4,SL)	(	1)	(0)		(	ITMUPDT4																								
(ITMUPDT4,SL)	(	1)	(1)		(*****																									
(ITMUPDT4,SL)	(	2)	(0)		(																									
(ITMUPDT4,SL)	(	2)	(1)		(* MODIFY RECORD=ENTER *																									
(ITMUPDT4,SL)	(	3)	(0)		(	ITEM NUMBER																								
(ITMUPDT4,SL)	(	3)	(1)		(* RECORD DELETE=PF11 *																									
(ITMUPDT4,SL)	(	4)	(0)		(	PRICE																								
(ITMUPDT4,SL)	(	4)	(1)		(* RETURN TO MENU=PF10 *																									
(ITMUPDT4,SL)	(	5)	(0)		(	QUANTITY/BOX																								
(ITMUPDT4,SL)	(	5)	(1)		(*****																									
(ITMUPDT4,SL)	(	6)	(0)		(	QUANTITY ON HAND																								
(ITMUPDT4,SL)	(	7)	(0)		(	SHIPPED YTD																								
(ITMUPDT4,SL)	(	8)	(0)		(	ORDERED YTD																								
(ITMUPDT4,SL)	(	9)	(0)		(	QUANTITY BACK-ORDERED																								
(ITMUPDT4,SL)	(	10)	(0)		(	RE-ORDER POINT																								
(ITMUPDT4,SL)	(	11)	(0)		(	PRODUCT CODE																								
** MSGW14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.																														
SCREEN	SIMT	SEGMENT	NO.	OF																									SEQUENCE	
NAME	TYPE			OCCURRENCES																									73 TO 80	
(ITMUPDT4,SS)	(	1)		(	1)																									

Figure A-2 Program Listings for Sample Applications (Page 39 of 84)

```

      V      I R C E P M N      A L C      PARTIAL
SCREEN SIMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ITMUPDT4,S0) ( 1) (T,DELETE ) (B) (D) (Y,Y) (P) (NM) (DELETE )
(ITMUPDT4,S0) ( 1) (1,ITEMNO ) (B) (Y) (U) (NM) (ITEMNO )
(ITMUPDT4,S1) (D) (IF (F,ATINID ) (EQ) (C,PF011 )
(ITMUPDT4,S1) (X) (IF (F,ATINID ) (EQ) (C,PF010 )
(ITMUPDT4,S0) ( 1) (1,ITMPRICE) (B) (P) (NM) (ITMPRICE )
(ITMUPDT4,S0) ( 1) (1,PACKBOX ) (B) (P) (NM) (PACKBOX )
(ITMUPDT4,S0) ( 1) (1,QANONHND) (B) (P) (NM) (QANONHND )
(ITMUPDT4,S0) ( 1) (1,SHIPYTD ) (B) (P) (NM) (SHIPYTD )
(ITMUPDT4,S0) ( 1) (1,ORDYTD ) (B) (P) (NM) (ORDYTD )
(ITMUPDT4,S0) ( 1) (1,QTYBKORD) (O) (NM) (QTYBKORD )
(ITMUPDT4,S0) ( 1) (1,REORDPT ) (B) (P) (NM) (REORDPT )
(ITMUPDT4,S0) ( 1) (1,PRODCODE) (B) (P) (NM) (PRODCODE )
(ITMUPDT4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)
** M5X901 TYPE 0 TEMPORARY FIELD DELETE DEFINED - FORMAT C, LENGTH 1, DECIMALS 0, EDIT LENGTH 1.

SCREEN SIMT CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN PROC TERMINAL SEQUENCE
NAME TYPE ACTN TYPE DESIG DESIG FORMAT CONTROL CODE ID NULL DEVICE 73 TO 80

(ENDSCRN4) (SD) (R) ( ) (@)

SCREEN SIMT LINE CONT- EXTN .....1....2....3....4....5.... <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6....7....8....9....0....+....1.... <-TEXT (58-114) 73 TO 80

(ENDSCRN4,SL) ( 1) (0) ( ENDSRN4 )
(ENDSCRN4,SL) ( 4) (0) ( YOU HAVE ENDED )
(ENDSCRN4,SL) ( 6) (0) ( YOUR APPLICATION. )
(ENDSCRN4,SL) ( 8) (0) ( PLEASE CLEAR THE SCREEN )
(ENDSCRN4,SL) (10) (0) ( TO CONTINUE. )
(ENDSCRN4,SL) (13) (0) (***** )
(ENDSCRN4,SL) (13) (1) (***** )
(ENDSCRN4,SL) (14) (0) ( )
(ENDSCRN4,SL) (15) (0) ( )
** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN SIMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES 73 TO 80

(ENDSCRN4,SS) ( 1) ( 1)

      V      I R C E P M N      A L C      PARTIAL
SCREEN SIMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ENDSCRN4,S0) ( 1) (F,DATE ) (O) (NM) (DATE )
(ENDSCRN4,S0) ( 1) (F,TIME ) (O) (NM) (TIME )
(ENDSCRN4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)

DEC 05, 2003 13.18.08 *****
* REQUEST NAME - MENUFPCK *
* INPUT STREAM REQUEST *
*****

SIMT CAT REQ SCREEN BACK BR RNT LINE
TYPE ACT REQUEST DESCRIPTION TYP NAME CONTROL TMP NUMBER

(ER) (1)
COMMENT ***** ; THIS REQUEST SAVES THE FUNCTION KEY PRESSED FOR LATER *****
COMMENT ***** ; VALIDATION BY SCREEN DEF, AND SETS UP THE ITEM BROWSE *****
COMMENT ***** ; WHEN APPROPRIATE. *****
COMMENT ***** ; *****
COMMENT ***** LET S.PFKEY = F.ATINID *****
COMMENT ***** IF F.ATINID EQ 'PF002' *****
COMMENT ***** LET T.SYNCFD1 = S.ITEMNO *****
COMMENT ***** LET PF(T.SYNCFD2,1,1) = S.ITEMNO *****
COMMENT ***** LET PF(T.SYNCFD2,2,6) = '999999' *****

```

Figure A-2 Program Listings for Sample Applications (Page 40 of 84)

```

COMMENT ***** END *****
COMMENT *****

-----

STMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD SEQ
NUMB NO. LEV CTR QLF FIELD NAME ATN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OPR NUMBER

( 1) (100) (R ) (F,ATTNID ) (S,PFKEY )
( 2) (101) (F,ATTNID ) (EQ) (C,PF002 )
( 3) (102) (NS) (END )
( 4) (103) (R ) (S,ITEMNO ) (T,SYNCFD1)
( 5) (104) (R ) (S,ITEMNO ) (T,SYNCFD2) ( 1) ( 1) (C)
( 6) (105) (R ) (C,999999 ) (T,SYNCFD2) ( 2) ( 6) (C)
** MK5BX01 TYPE 0 DECODING REQUEST MENUPFCK FOR SCREEN ORDMENU4 .

*****
DEC 05, 2003 13.18.08 * REQUEST NAME - SCRNCNT *
* INPUT STREAM REQUEST *
*****

-----

STMT CAT REQ SCREEN BACK BR RNT LINE
TYPE ACT REQUEST DESCRIPTION TYP NAME CONTROL TMP NUMBER

(ER) (2)
COMMENT ***** ; THIS REQUEST CALCULATES THE NUMBER OF SCREENS OF BACK- *****
COMMENT ***** ; ORDERS THAT ARE POSSIBLE, SHOULD THE TERMINAL OPERATOR *****
COMMENT ***** ; PRESS PF3. *****
COMMENT ***** ; *****
COMMENT ***** IF F.ATTNID EQ 'PF003' *****
COMMENT ***** LET T.SCREENNO = 1 *****
COMMENT ***** LET T.NOSCREEN = 1 *****
COMMENT ***** IF 1.BKORDRNO EQ 1.BKORDRNO ;BEGIN LOOP FOR ALL BA *****
COMMENT ***** LET T.COUNTOCC = T.COUNTOCC + 1 *****
COMMENT ***** IF T.COUNTOCC GT 8 *****
COMMENT ***** LET T.NOSCREEN = T.NOSCREEN + 1 *****
COMMENT ***** LET T.COUNTOCC = 1 *****
COMMENT ***** END *****
COMMENT ***** END *****
COMMENT ***** END *****
COMMENT ***** END *****

-----

STMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD SEQ
NUMB NO. LEV CTR QLF FIELD NAME ATN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OPR NUMBER

( 1) (100) (F,ATTNID ) (EQ) (C,PF003 )
( 2) (101) (NS) (END )
( 3) (102) (R ) (D,1 ) (T,SCREENNO)
( 4) (103) (R ) (D,1 ) (T,NOSCREEN)
( 5) (104) (1,BKORDRNO) (EQ) (1,BKORDRNO )
( 6) (105) (NS) (END )
( 7) (106) (T,COUNTOCC) (+) (D,1 ) (T,COUNTOCC)
( 8) (107) (T,COUNTOCC) (GT) (D,8 )
( 9) (108) (NS) (END )
(10) (109) (T,NOSCREEN) (+) (D,1 ) (T,NOSCREEN)
(11) (110) (R ) (D,1 ) (T,COUNTOCC)
** MK5BX01 TYPE 0 DECODING REQUEST SCRNCNT FOR SCREEN ORDMENU4 .
** MK5DI02 TYPE 0 (S# 7) DEFINITION OF TEMPORARY FIELD COUNTOCC - FORMAT P LENGTH 8 DECIMALS 0 EDIT LENGTH 20 .

*****
** MK5TJ05 TYPE 0 THE FOLLOWING CICS TS SYSTEM DEFINITION UTILITY INPUT STATEMENTS ARE REQUIRED FOR THIS APPLICATION.
DEFINE GROUP(???),DUMP(NO),TWASIZE(00000),TRANSACTION(M510),
INDOUBT(BACKOUT),PROGRAM(M510)
DEFINE GROUP(???),LANGUAGE(ASSEMBLER),PROGRAM(M510)
** MK5TJ03 TYPE 0 THE IMS PSB GENERATION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -
PCB TYPE=DB,POS=M,KEYLEN=0017,PROCOPT=AP,DBDNAME=CUSTOMDB
SENSESEG NAME=CUSTOMER
SENSESEG NAME=ORDER,PARENT=CUSTOMER
SENSESEG NAME=ITEMORD,PARENT=ORDER
PCB TYPE=DB,POS=M,KEYLEN=0014,PROCOPT=AP,DBDNAME=ITEMDB
SENSESEG NAME=ITEMROOT
SENSESEG NAME=BKORDERS,PARENT=ITEMROOT
PCB TYPE=DB,POS=M,KEYLEN=0007,PROCOPT=GP,DBDNAME=ITEMDB
SENSESEG NAME=ITEMROOT
PSBGEN LANG=ASSEM,MAXQ=002,PSENAME=M14CSIT
END
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - QANONHND COUNTX LASTOCC FIRSTOCC ERROR MENU .
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - BACKORD UPDATE SCRNSLCT MSG .

```

Figure A-2 Program Listings for Sample Applications (Page 41 of 84)

```

** MK5M204 TYPE 0 NUMBER OF MESSAGES PRINTED IS 17.
      M5INPUT - 66 RECORDS INPUT
      M5PUNCH - 667 RECORDS OUTPUT
      M5MAP - 0 RECORDS OUTPUT
      M5TPIEL - 14 RECORDS OUTPUT
      M5PREP - 0 RECORDS OUTPUT
      M5COMMAR - 135 RECORDS OUTPUT
      M5LIST - 639 RECORDS OUTPUT
      75 TRACKS ASSIGNED TO M5LIB - 23 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 800.
** MK5M209 TYPE 0 16392 BYTES OF TYPE A STORAGE USED DURING APPLICATION GENERATION
-----
RUN      SIMT LIB  SYS  DEBUG SRC LKED  SYS  CICS MSG  POS RUN  TRG TRG PSB      PROGRAM      SEQUENCE
NAME     TYPE OPEN DELIM LEVEL LST PQMS ROLLB KEYS PROC PCB TYPE MON SYS NAME      NAME              73 TO 80
-----
(M501    ,RC)      (#)                      (C)          (G) (C)    (M14CSIT )

RUN      SIMT  FILE      FILE      SYNCHRONIZING      SYNC  SYNC  DL/I  RECORD  STOR  SQL      SEQUENCE
NAME     TYPE  NAME      TYPE      FIELD 1  FIELD 2  QUAL  TYPE  PATH  LENGTH FLD  TYPE  AUTH-ID  73 TO 80
-----
(M501    ,RF) (CUSTOMER) (FILE )                      (U)
(M501    ,RF) (ITEMX ) (FILE1 )                      (U)
(M501    ,RF) (ITEMX ) (FILE2 ) (SYNCFD1) (SYNCFD2) (T)

***** BEGIN ASL PROCEDURE VALDCUPD.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED
;
IF F.ATTNID NE 'ENTER', 'PF006', 'PF007', 'PF010'
MSG 'INVALID KEY PRESSED - RETRY'
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST VALDCUPD.

***** BEGIN ASL PROCEDURE FINDX .
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE MAKES SURE THAT ONLY ONE 'X' IS ENTERED
; WHEN PF6 IS PRESSED ON SCREEN CUSTUPD4
;
IF F.ATTNID EQ 'PF006' AND,
S.ACTION EQ 'X'
LET T.COUNTX = T.COUNTX + 1
IF T.COUNTX > 1
MSG 'ONLY ONE 'X' PER PF6'
END
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST FINDX .

***** BEGIN ASL PROCEDURE ONEXCUPD.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE MAKES SURE THAT AT LEAST ONE 'X'
; WAS ENTERED IF PF6 WAS PRESSED.
;
IF F.ATTNID EQ 'PF006' AND,
T.COUNTX NE 1
MSG 'AT LEASET ONE 'X' PER PF6'
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST ONEXCUPD.

***** BEGIN ASL PROCEDURE SYNCLOAD.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE SETS UP THE SYNCHRONIZATION FIELDS FOR
; THE ITEM BROWSE WHEN APPROPRIATE.
;
IF F.ATTNID EQ 'PF006'
LET T.SYNCFD1 = ITEMORD
LET T.SYNCFD2 = ITEMORD
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST SYNCLOAD.

```

Figure A-2 Program Listings for Sample Applications (Page 42 of 84)

```

***** BEGIN ASL PROCEDURE OUR1N1 .
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED
; BY THE OPERATOR FOR SCREEN ORUPT4
;
IF F.ATTNID EQ 'ENTER', 'PF001', 'PF010'
; IF PF1 OR PF10 HAVE BEEN PRESSED, MAKE SURE ACTION IS BLANK
IF F.ATTNID NE 'ENTER'
LET S.ACTION = ' '
END
ELSE
MSG 'INVALID KEY PRESSED - RETRY'
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OUR1N1 .

***** BEGIN ASL PROCEDURE OUR2N1 .
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE MAKES SURE THAT AN ITEM (WHOSE QUANTITY
; ORDERED HAS BEEN CHANGED) HASN'T ALREADY BEEN SHIPPED.
;
IF F.ATTNID EQ 'ENTER' AND S.ACTION EQ 'U'
; DOES THIS ITEM ALREADY EXIST ON THIS ORDER
IF ITEMORD EQ ITEMORD
; IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED
IF S.ITEMQTYOR EQ ITEMQTYOR
CONTINUE
ELSE
; IF NOT, HAS THE ITEM BEEN SHIPPED?
IF ITEMORD EQ ITEMSHIP
LET T.MSG = 'CANNOT CHANGE QUANTITY ORDERED '
LET PF(T.MSG,32,31) = '- ITEM ALREADY SHIPPED'
SET S.ITEMORD
MSG T.MSG
END
END
IF S.ITEMPRICE EQ ' '
LET S.ITEMPRICE = 1.ITEMPRICE
END
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OUR2N1 .

***** BEGIN ASL PROCEDURE OUR2N2 .
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE CONTROLS THE UPDATE OF THE QUANTITY ON
; HAND FIELD IN THE ITEM DATA BASE
;
IF F.ATTNID EQ 'ENTER' AND S.ACTION EQ 'U'
; IS ITEM ALREADY ON THIS ORDER?
IF ITEMORD EQ ITEMORD
; IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED
IF S.ITEMQTYOR EQ ITEMQTYOR
CONTINUE
ELSE
LET 1.QANONHND = 1.QANONHND + ITEMQTYOR
END
END
LET T.QANONHND = 1.QANONHND - S.ITEMQTYOR
IF T.QANONHND LT 0
IF ITEMORD EQ ITEMORD
LET 1.QANONHND = 1.QANONHND - ITEMQTYOR
END
END
LET 1.QANONHND = T.QANONHND * -1
LET T.MSG = 'QUANTITY ORDERED IS '
LET PF(T.MSG,21,7) = T.QANONHND
LET PF(T.MSG,31,31) = 'MORE THAN QUANTITY ON HAND'
SET S.ITEMQTYOR
MSG T.MSG
END
LET 1.QANONHND = T.QANONHND
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OUR2N2 .

***** BEGIN ASL PROCEDURE VLDORDAD.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED
; FOR THE ORDADDR SCREEN
;
IF F.ATTNID NE 'ENTER', 'PF001', 'PF006', 'PF010'

```

Figure A-2 Program Listings for Sample Applications (Page 43 of 84)


```

        MSG 'FUNCTION KEY INVALID - RETRY'
    END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST VLDORDAD.

***** BEGIN ASL PROCEDURE OAR2N1 .
***** Using ASL Translator Version 4.50 *****
; IF PRICE HAS NOT BEEN ENTERED ON THIS NEW ITEM, GET
; CURRENT PRICE FROM THE DATA BASE.
;
IF S.ITMPRICE EQ ' '
    LET S.ITMPRICE = 1.ITMPRICE
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OAR2N1 .

***** BEGIN ASL PROCEDURE OUTYPE2 .
***** Using ASL Translator Version 4.50 *****
; CALL TYPE2 PROCEDURES FOR ORDUPDT4
;
CALL OUR2N1
CALL OUR2N2
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OUTYPE2 .

***** BEGIN ASL PROCEDURE CUTYPE1 .
***** Using ASL Translator Version 4.50 *****
; CALL TYPE 1 PROCEDURES FOR CUSTUPD4
;
CALL VALDCUPD
CALL FINDX
CALL ONEXCUPD
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST CUTYPE1 .

***** BEGIN ASL PROCEDURE TEMPS .
***** Using ASL Translator Version 4.50 *****
; TEMP FIELD DEFINITIONS
SYNCFLD1: FIELD C 7
SYNCFLD2: FIELD C 7
SCREENNO: FIELD Z 2
NOSCREEN: FIELD Z 2
MSG: FIELD C 74
SCRNSLCT: FIELD C 1
UPDATE: FIELD C 1 INIT 'U'
BACKORD: FIELD C 1 INIT 'B'
MENU: FIELD C 1 INIT 'M'
ERROR: FIELD C 1 INIT 'E'
FIRSTOCC: FIELD P 2
LASTOCC: FIELD P 2
COUNTX: FIELD P 2
QANONHND: FIELD C 7
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST TEMPS .

*****
DEC 05, 2003 13.26.45          * REQUEST NAME - TEMPS *
                              * INPUT STREAM REQUEST *
                              *****

```

STMT	CAT		REQ	SCREEN	BACK	BR	RNT	LINE
TYPE	ACT	REQUEST DESCRIPTION	TYP	NAME	CONTROL	TMP		NUMBER
(ER)				(N)				
COMMENT	*****	; TEMP FIELD DEFINITIONS		*****				
COMMENT	*****	SYNCFLD1: FIELD C 7		*****				
COMMENT	*****	SYNCFLD2: FIELD C 7		*****				
COMMENT	*****	SCREENNO: FIELD Z 2		*****				
COMMENT	*****	NOSCREEN: FIELD Z 2		*****				
COMMENT	*****	MSG: FIELD C 74		*****				
COMMENT	*****	SCRNSLCT: FIELD C 1		*****				
COMMENT	*****	UPDATE: FIELD C 1 INIT 'U'		*****				
COMMENT	*****	BACKORD: FIELD C 1 INIT 'B'		*****				
COMMENT	*****	MENU: FIELD C 1 INIT 'M'		*****				
COMMENT	*****	ERROR: FIELD C 1 INIT 'E'		*****				
COMMENT	*****	FIRSTOCC: FIELD P 2		*****				
COMMENT	*****	LASTOCC: FIELD P 2		*****				
COMMENT	*****	COUNTX: FIELD P 2		*****				
COMMENT	*****	QANONHND: FIELD C 7		*****				

Figure A-2 Program Listings for Sample Applications (Page 44 of 84)

```

COMMENT *****

-----
STMT  FIELD  FIELD FLD DEC OUTPT EDIT  INITIAL  SEQUENCE
TYPE  NAME   LENGTH TYP PLC EDIT  LGTH   VALUE   COMMENTS  NUMBER
-----
(TF) (SYNCFD1) ( 7) (C)
(TF) (SYNCFD2) ( 7) (C)
(TF) (SCREENNO) ( 2) (Z)
(TF) (NOSCREEN) ( 2) (Z)
(TF) (MSG      ) (74) (C)
(TF) (SCRNSLCT) ( 1) (C)
(TF) (UPDATE   ) ( 1) (C)          (U      )
(TF) (BACKORD  ) ( 1) (C)          (B      )
(TF) (MENU     ) ( 1) (C)          (M      )
(TF) (ERROR    ) ( 1) (C)          (E      )
(TF) (FIRSTOCC ) ( 2) (P)
(TF) (LASTOCC  ) ( 2) (P)
(TF) (COUNTX   ) ( 2) (P)
(TF) (QANONHND) ( 7) (C)

-----
DEC 05, 2003 13.26.45          *****
*          SCREEN CONTROL      *
*****

-----
INPUT  STMT  SEQ LOG CON .OPERAND A.. OPER .....OPERAND B..... OUTPUT  .OPERAND C.. PARTL FLD SEQUENCE
SCREEN TYPE NO. LVL CTR QLF FIELD  ATN QLF FIELD NAME/CONSTANT  SCREEN  QLF FLD/CNST STR NM OP  73 TO 80
-----
(CUSTUPD4) (SC)
(CUSTUPD4) (SC)          (F,ATTNID ) (XR) (1,CUTYPE1 )
(CUSTUPD4) (SC)          (EQ) (C,ENTER )
(CUSTUPD4) (SC)          (OS)
(CUSTUPD4) (SC)          (E) (F,ATTNID ) (EQ) (C,PF006 ) (CUSTUPD4)
(CUSTUPD4) (SC)          (OS) (O,SYNCLoad ) (ORDUPDT4)
(CUSTUPD4) (SC)          (E) (F,ATTNID ) (EQ) (C,PF007 )
(CUSTUPD4) (SC)          (OS) (ORDADD4 )
(CUSTUPD4) (SC)          (E) (F,ATTNID ) (EQ) (C,PF010 )
(CUSTUPD4) (SC)          (OS) (ORDMENU4)
(ORDUPDT4) (SC)          (XR) (1,OURIN1 )
(ORDUPDT4) (SC)          (XR) (2,OUTYPE2 )
(ORDUPDT4) (SC)          (F,ATTNID ) (EQ) (C,PF001 )
(ORDUPDT4) (SC)          (OS) (CUSTUPD4)
(ORDUPDT4) (SC)          (E) (F,ATTNID ) (EQ) (C,PF010 )
(ORDUPDT4) (SC)          (OS) (ORDMENU4)
(ORDUPDT4) (SC)          (E) (F,ATTNID ) (EQ) (C,ENTER )
(ORDUPDT4) (SC)          (OS) (ORDUPDT4)
(ORDADD4 ) (SC)          (XR) (1,VLDORDAD )
(ORDADD4 ) (SC)          (XR) (2,QAR2N1 )
(ORDADD4 ) (SC)          (F,ATTNID ) (EQ) (C,PF001 )
(ORDADD4 ) (SC)          (OS) (CUSTUPD4)
(ORDADD4 ) (SC)          (E) (F,ATTNID ) (EQ) (C,PF006 )
(ORDADD4 ) (SC)          (OS) (ORDUPDT4)
(ORDADD4 ) (SC)          (E) (F,ATTNID ) (EQ) (C,PF010 )
(ORDADD4 ) (SC)          (OS) (ORDMENU4)
(ORDADD4 ) (SC)          (E) (F,ATTNID ) (EQ) (C,ENTER )
(ORDADD4 ) (SC)          (OS) (ORDADD4 )

-----
SCREEN  STMT  CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN  PROC TERMINAL SEQUENCE
NAME    TYPE  ACTN TYPE DESIG DESIG  FORMAT CONTROL CODE   ID    NULL  DEVICE  73 TO 80
-----
(CUSTUPD4) (SD) (R)          ( ) (E)          (M501 ) (CUSTSET)

-----
SCREEN  STMT  LINE CONT- EXIN .....1....2....3....4....5.... <-TEXT (1-57) SEQUENCE
NAME    TYPE  NO.  INUE ATTR  ..6....7....8....9....0....+...1.... <-TEXT (58-114) 73 TO 80
-----
(CUSTUPD4,SL) ( 1) (0)          ( CUSTUPD4 )
(CUSTUPD4,SL) ( 3) (0)          ( CUSTOMER NUMBER )
(CUSTUPD4,SL) ( 4) (0)          ( CUSTOMER NAME )
(CUSTUPD4,SL) ( 5) (0)          ( CUSTOMER PHONE )
(CUSTUPD4,SL) ( 7) (0)          ( ACT ORDER ) (F.O. DUE INVOICE )
(CUSTUPD4,SL) ( 7) (1)          ( ORDER )
(CUSTUPD4,SL) ( 8) (0)          ( ION NUMBER DATE SELLER NUMBER DATE GENERATED)
(CUSTUPD4,SL) ( 8) (1)          ( COMPL )
(CUSTUPD4,SL) ( 9) (0)          ( )

```

Figure A-2 Program Listings for Sample Applications (Page 45 of 84)

```

(CUSTUPD4,SL) ( 9) (1) ( - )
(CUSTUPD4,SL) (11) (0) ( - )
(CUSTUPD4,SL) (11) (1) ( - )
(CUSTUPD4,SL) (13) (0) ( - )
(CUSTUPD4,SL) (13) (1) ( - )
(CUSTUPD4,SL) (15) (0) ( - )
(CUSTUPD4,SL) (15) (1) ( - )
(CUSTUPD4,SL) (17) (0) ( - )
(CUSTUPD4,SL) (17) (1) ( - )
(CUSTUPD4,SL) (19) (0) ( - )
(CUSTUPD4,SL) (19) (1) ( - )
(CUSTUPD4,SL) (21) (0) (NEXT SCREEN OF ORDERS=PA1 UPDATE=U ACT + ENTER )
(CUSTUPD4,SL) (22) (0) (ORDER DETAILS=X ACT + PF6 ADD ORDER=PF7 )
(CUSTUPD4,SL) (22) (1) ( RETURN TO MENU=PF10 )
** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----
SCREEN SIMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES
73 TO 80
-----
(CUSTUPD4,SS) ( 1) ( 1)
-----
V I R C E P M N A L C PARTIAL
SCREEN SIMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80
-----
(CUSTUPD4,S0) ( 1) (N,CUSTNO ) (B) (X) (NM) (CUSTNO )
(CUSTUPD4,S1) (M) (IF) (F,ATTNID ) (EQ) (C,PF006 )
(CUSTUPD4,S1) (M) (IF) (F,ATTNID ) (EQ) (C,PF007 )
(CUSTUPD4,S1) (U) (IF) (F,ATTNID ) (EQ) (C,ENTER )
(CUSTUPD4,S0) ( 1) (N,CUSTNAME) (B) (P) (NM) (CUSTNAME )
(CUSTUPD4,S0) ( 1) (N,CUSTPH ) (B) (P) (NM) (CUSTPH )
(CUSTUPD4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)
-----
SCREEN SIMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES
73 TO 80
-----
(CUSTUPD4,SS) ( 2) ( 5)
-----
V I R C E P M N A L C PARTIAL
SCREEN SIMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80
-----
(CUSTUPD4,S0) ( 2) (I) (N) (NM) (ACTION )
(CUSTUPD4,S0) ( 2) (N,ORDERNO ) (B) (X) (NM) (ORDERNO )
(CUSTUPD4,S1) (U) (IF) (S,ACTION ) (EQ) (C,U )
(CUSTUPD4,S1) (M) (IF) (S,ACTION ) (EQ) (C,X )
(CUSTUPD4,S0) ( 2) (N,ORDDATE) (B) (P) (NM) (ORDDATE )
(CUSTUPD4,S0) ( 2) (N,ORPERSON) (B) (P) (NM) (ORPERSON )
(CUSTUPD4,S0) ( 2) (N,ORDPONUM) (B) (P) (NM) (ORDPONUM )
(CUSTUPD4,S0) ( 2) (N,ORDUDAT) (B) (P) (NM) (ORDUDAT )
(CUSTUPD4,S0) ( 2) (N,ORDINVCN) (B) (P) (NM) (ORDINVCN )
(CUSTUPD4,S0) ( 2) (N,ORDCMPLT) (B) (P) (NM) (ORDCMPLT )
-----
SCREEN SIMT CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN PROC TERMINAL SEQUENCE
NAME TYPE ACTN TYPE DESIG DESIG FORMAT CONTROL CODE ID NULL DEVICE
73 TO 80
-----
(ORDUPDT4) (SD) (R) ( ) (0)
-----
SCREEN SIMT LINE CONT- EXTN .....1.....2.....3.....4.....5..... <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6.....7.....8.....9.....0.....1..... <-TEXT (58-114) 73 TO 80
-----
(ORDUPDT4,SL) ( 1) (0) ( ORDUPDT4 )
(ORDUPDT4,SL) ( 1) (1) (UPDATE=U ACT + ENTER )
(ORDUPDT4,SL) ( 2) (0) ( CUSTOMER NUMBER )
(ORDUPDT4,SL) ( 2) (1) (CUST UPDT SCR=PF1 )
(ORDUPDT4,SL) ( 3) (0) ( ORDER NUMBER )
(ORDUPDT4,SL) ( 3) (1) (RETURN TO MENU=PF10 )
(ORDUPDT4,SL) ( 4) (0) ( CUSTOMER NAME )
(ORDUPDT4,SL) ( 5) (1) ( QTY )

```

Figure A-2 Program Listings for Sample Applications (Page 46 of 84)

```

(ORDUPDT4,SL) ( 6) (0) ( ACT ITEM      ITEM      PRICE  QUANTITY )
(ORDUPDT4,SL) ( 6) (1) ( BACK      ON )
(ORDUPDT4,SL) ( 7) (0) ( ION NUMBER      NAME      ORDERED )
(ORDUPDT4,SL) ( 7) (1) ( ORDERED HAND )
(ORDUPDT4,SL) ( 8) (0) ( - )
(ORDUPDT4,SL) ( 8) (1) ( - )
(ORDUPDT4,SL) ( 9) (0) ( - )
(ORDUPDT4,SL) ( 9) (1) ( - )
(ORDUPDT4,SL) (10) (0) ( - )
(ORDUPDT4,SL) (10) (1) ( - )
(ORDUPDT4,SL) (11) (0) ( - )
(ORDUPDT4,SL) (11) (1) ( - )
(ORDUPDT4,SL) (12) (0) ( - )
(ORDUPDT4,SL) (12) (1) ( - )
(ORDUPDT4,SL) (13) (0) ( - )
(ORDUPDT4,SL) (13) (1) ( - )
(ORDUPDT4,SL) (14) (0) ( - )
(ORDUPDT4,SL) (14) (1) ( - )
(ORDUPDT4,SL) (15) (0) ( - )
(ORDUPDT4,SL) (15) (1) ( - )
(ORDUPDT4,SL) (16) (0) ( - )
(ORDUPDT4,SL) (16) (1) ( - )
(ORDUPDT4,SL) (17) (0) ( - )
(ORDUPDT4,SL) (17) (1) ( - )
(ORDUPDT4,SL) (18) (0) ( - )
(ORDUPDT4,SL) (18) (1) ( - )
(ORDUPDT4,SL) (19) (0) ( - )
(ORDUPDT4,SL) (19) (1) ( - )
(ORDUPDT4,SL) (20) (0) ( - )
(ORDUPDT4,SL) (20) (1) ( - )
(ORDUPDT4,SL) (21) (0) ( - )
(ORDUPDT4,SL) (21) (1) ( - )
(ORDUPDT4,SL) (22) (0) ( - )
(ORDUPDT4,SL) (22) (1) ( - )
(ORDUPDT4,SL) (23) (0) ( - )
(ORDUPDT4,SL) (23) (1) ( - )
** MSGW14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

```

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE
				73 TO 80

```

(ORDUPDT4,SS) ( 1) ( 1)

```

SCREEN NAME	SIMT TYPE	SEG NO.	SEG S	V	SOURCE/DEST	IR	CE	PMN	A	LC	C	LO	E/V	OPERAND B.	PARTIAL FIELD	SEQUENCE
					Q LF FLD NAME	OD	IR	RT	DM	T	V	N	OP	Q LF FIELD NAME OR CONSTANT	ST NO	OP 73 TO 80
(ORDUPDT4,S0)	(1)				(N,CUSTNO)	(B)		(H,Y)	(X)	(NM)	(CUSTNO			)		
(ORDUPDT4,S1)									(M)	(IF)	(F,ATINID)	(EQ)	(C,PF001	)		
(ORDUPDT4,S1)									(M)	(IF)	(F,ATINID)	(EQ)	(C,ENTER	)		
(ORDUPDT4,S0)	(1)				(N,ORDERNO)	(B)		(Y)	(X)	(NM)	(ORDERNO			)		
(ORDUPDT4,S1)									(Q)	(IF)	(F,ATINID)	(EQ)	(C,ENTER	)		
(ORDUPDT4,S0)	(1)				(N,CUSTNAME)	(O)			(NM)	(CUSTNAME				)		
(ORDUPDT4,S0)	(1)	(A)			(F,MSGLINE)		(H)									(*SYSTEM*)

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE
				73 TO 80

```

(ORDUPDT4,SS) ( 2) ( 15)

```

SCREEN NAME	SIMT TYPE	SEG NO.	SEG S	V	SOURCE/DEST	IR	CE	PMN	A	LC	C	LO	E/V	OPERAND B.	PARTIAL FIELD	SEQUENCE
					Q LF FLD NAME	OD	IR	RT	DM	T	V	N	OP	Q LF FIELD NAME OR CONSTANT	ST NO	OP 73 TO 80
(ORDUPDT4,S0)	(2)						(I)		(H)	(N)	(NM)	(ACTION		)		
(ORDUPDT4,S0)	(2)				(N,ITEMORD)	(B)				(X)	(NM)	(ITEMORD		)		
(ORDUPDT4,S1)										(V)	(IF)	(S,ACTION)	(EQ)	(C,U	)	
(ORDUPDT4,S0)	(2)	(S)			(1,ITEMNO)	(I)				(X)						
(ORDUPDT4,S1)										(U)	(IF)	(S,ACTION)	(EQ)	(C,U	)	
(ORDUPDT4,S0)	(2)				(2,ITEMNAME)	(O)				(NM)	(ITEMNAME			)		
(ORDUPDT4,S0)	(2)				(N,ITEMPRICE)	(B)				(P)	(NM)	(ITEMPRICE		)		
(ORDUPDT4,S0)	(2)				(N,ITEMQTYOR)	(B)				(P)	(NM)	(ITEMQTYOR		)		
(ORDUPDT4,S0)	(2)				(N,ITEMQTYORD)	(B)				(P)	(NM)	(ITEMQTYORD		)		
(ORDUPDT4,S0)	(2)				(2,QANONHND)	(O)				(NM)	(QANONHND			)		

Figure A-2 Program Listings for Sample Applications (Page 47 of 84)

SCREEN NAME	SIMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACT CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80							
(ORDADD4) (SD) (R) () (0)																			
SCREEN NAME	SIMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5..... ..6.....7.....8.....9.....0.....1.....						<-TEXT (1-57) <-TEXT (58-114)	SEQUENCE 73 TO 80							
(ORDADD4 ,SL)	(1)	(0)			(ORDADD4						)								
(ORDADD4 ,SL)	(1)	(1)			(ADD NEW DATA=ENTER						)								
(ORDADD4 ,SL)	(2)	(1)			(CUST UPDT SCRN=PF1						)								
(ORDADD4 ,SL)	(3)	(0)			(CUSTOMER NUMBER						)								
(ORDADD4 ,SL)	(3)	(1)			(VIEW ADDED ITEMS=PF6						)								
(ORDADD4 ,SL)	(4)	(0)			(CUSTOMER NAME						)								
(ORDADD4 ,SL)	(4)	(1)			(RETURN TO MENU=PF10						)								
(ORDADD4 ,SL)	(5)	(0)			(CUSTOMER PHONE						)								
(ORDADD4 ,SL)	(7)	(0)			(ORDER NUMBER	SELLER					)								
(ORDADD4 ,SL)	(8)	(0)			(P.O. NUMBER	DUE DATE	ORDER DATE				)								
(ORDADD4 ,SL)	(10)	(0)			(ITEM	PRICE	QUANTITY	BACK			)								
(ORDADD4 ,SL)	(11)	(0)			(NUMBER		ORDERED	ORDERED			)								
(ORDADD4 ,SL)	(12)	(0)			(						)								
(ORDADD4 ,SL)	(13)	(0)			(						)								
(ORDADD4 ,SL)	(14)	(0)			(						)								
(ORDADD4 ,SL)	(15)	(0)			(						)								
(ORDADD4 ,SL)	(16)	(0)			(						)								
(ORDADD4 ,SL)	(17)	(0)			(						)								
(ORDADD4 ,SL)	(18)	(0)			(						)								
(ORDADD4 ,SL)	(19)	(0)			(						)								
(ORDADD4 ,SL)	(20)	(0)			(						)								
(ORDADD4 ,SL)	(21)	(0)			(						)								
(ORDADD4 ,SL)	(22)	(0)			(						)								
(ORDADD4 ,SL)	(23)	(0)			(						)								
** M\$W14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.																			
SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES									SEQUENCE 73 TO 80							
(ORDADD4 ,SS) (1) (1)																			
SCREEN NAME	SIMT TYPE	SEG NO.	V A	S QLF	DEST FLD NAME	I O	R D	C I	E R	P R	M T	N D	A T	L C	C L	O E/V	OPERAND B.....	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80
(ORDADD4 ,S0)	(1)				(N,CUSTNO)	(B)			(Y)		(X)	(NM)	(CUSTNO		)				
(ORDADD4 ,S1)											(M)	(IF)	(F,ATTNID)	(EQ)	(C,PF001		)		
(ORDADD4 ,S1)												(M)	(IF)	(F,ATTNID)	(EQ)	(C,PF006		)	
(ORDADD4 ,S1)												(M)	(IF)	(F,ATTNID)	(EQ)	(C,ENTER		)	
(ORDADD4 ,S0)	(1)				(N,CUSTNAME)	(O)						(NM)	(CUSTNAME		)				
(ORDADD4 ,S0)	(1)				(N,CUSTPH)	(O)						(NM)	(CUSTPH		)				
(ORDADD4 ,S0)	(1)				(N,ORDERNO)	(B)					(X)	(NM)	(ORDERNO		)				
(ORDADD4 ,S1)												(M)	(IF)	(F,ATTNID)	(EQ)	(C,PF006		)	
(ORDADD4 ,S1)												(V)	(IF)	(F,ATTNID)	(EQ)	(C,ENTER		)	
(ORDADD4 ,S0)	(1)				(N,ORPERSON)	(B)						(P)	(NM)	(ORPERSON		)			
(ORDADD4 ,S0)	(1)				(N,ORDPONUM)	(B)						(P)	(NM)	(ORDPONUM		)			
(ORDADD4 ,S0)	(1)				(N,ORDUDAT)	(B)						(P)	(NM)	(ORDUDAT		)			
(ORDADD4 ,S0)	(1)				(N,ORDDATE)	(B)						(P)	(NM)	(ORDDATE		)			
(ORDADD4 ,S0)	(1)	(A)			(F,MSGLINE)				(H)										(*SYSTEM*)

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES															SEQUENCE 73 TO 80	
(ORDADD4 ,SS) (2) (11)																			
SCREEN NAME	SIMT TYPE	SEG NO.	V A	S QLF	DEST FLD NAME	I O	R D	C I	E R	P R	M T	N D	A T	L C	C L	O E/V	OPERAND B.....	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80
QLF FIELD NAME ODI RRT DM T V N OP QLF FIELD NAME OR CONSTANT																			
(ORDADD4 ,S0) (2) (N,ITEMORD) (I) (X) (NM) (ITEMORD)																			

Figure A-2 Program Listings for Sample Applications (Page 48 of 84)

```

(ORDADD4 ,S1) (I) (IF (F,ATTNID ) (EQ) (C,ENTER )
(ORDADD4 ,S0) ( 2) (S) (1,ITEMNO ) (I) (X)
(ORDADD4 ,S1) (U) (IF (F,ATTNID ) (EQ) (C,ENTER )
(ORDADD4 ,S0) ( 2) (N,ITMPRICE) (I) (R) (NM) (ITMPRICE )
(ORDADD4 ,S0) ( 2) (N,ITMQTYCR) (I) (R) (NM) (ITMQTYCR )
(ORDADD4 ,S0) ( 2) (S) (1,QANONHND) (I) (S)
(ORDADD4 ,S1) (LE) (1,QANONHND )
(ORDADD4 ,S1) (MS) (C,NOT ENOUGH IN STOCK )
(ORDADD4 ,S0) ( 2) (N,QTYBKORD) (I) (R) (NM) (QTYBKORD )

```

SCREEN NAME	STMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

```

(ORDMENU4) (SD) (R) ( ) (8) (M510 ) (M510 )

```

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80

```

(ORDMENU4,SL) ( 1) (0) ( ORDMENU4 )
(ORDMENU4,SL) ( 3) (0) ( CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS:)
(ORDMENU4,SL) ( 5) (0) ( PF1---CUSTOMER ADD/UPDATE )
(ORDMENU4,SL) ( 7) (0) ( PF2---ITEM BROWSE BY PRODUCT KEY )
(ORDMENU4,SL) ( 9) (0) ( PF3---BACK-ORDERED ITEM )
(ORDMENU4,SL) (11) (0) ( PF4---ITEM ADD/UPDATE )
(ORDMENU4,SL) (13) (0) ( PF5---END TRANSACTION )
(ORDMENU4,SL) (16) (0) ( ENTER CUSTOMER OR ITEM NUMBER )
(ORDMENU4,SL) (18) (0) ( ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2)
(ORDMENU4,SL) (18) (1) ( IS USED )
** MSGW14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

```

SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

```

(ORDMENU4,SS) ( 1) ( 1)

```

SCREEN NAME	STMT TYPE	SEG NO.	V A	SOURCE/DEST QLF FLD NAME	I R C E P M N / Q H U R R O U O D I R R T D M	A L C C L O E / V T V N O P	OPERAND B..... QLF FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80

```

(ORDMENU4,S0) ( 1) (N,CUSTNO ) (I) (X) (NM) (ITEMNO )
(ORDMENU4,S1) (V) (IF (S,PEKEY ) (EQ) (C,PF001 )
(ORDMENU4,S0) ( 1) (S) (1,ITEMNO ) (I) (X)
(ORDMENU4,S1) (M) (IF (S,PEKEY ) (EQ) (C,PF003 )
(ORDMENU4,S1) (V) (IF (S,PEKEY ) (EQ) (C,PF004 )
(ORDMENU4,S0) ( 1) (V) (I) (N) (NM) (PFKEY )
(ORDMENU4,S1) (IN) (5 )
(ORDMENU4,S1) (EQ) (C,PF001 )
(ORDMENU4,S1) (1,0,EQ) (C,PF002 )
(ORDMENU4,S1) (1,0,EQ) (C,PF003 )
(ORDMENU4,S1) (1,0,EQ) (C,PF004 )
(ORDMENU4,S1) (1,0,EQ) (C,PF005 )
(ORDMENU4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)

```

```

DEC 05, 2003 13.26.45 *****
* REQUEST NAME - VALDCUPD *
* INPUT STREAM REQUEST *
*****

```

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER

```

(ER) (S)
COMMENT ***** ; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED *****
COMMENT ***** ; *****
COMMENT ***** IF F.ATTNID NE 'ENTER', 'PF006', 'PF007', 'PF010' *****
COMMENT ***** MSG 'INVALID KEY PRESSED - RETRY' *****
COMMENT ***** END *****
COMMENT *****

```

Figure A-2 Program Listings for Sample Applications (Page 49 of 84)

STMT NUMB	SEQ NO.	LOG LEV	CON CIR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER
(1)	(100)			(F,ATTNID)	(NE)	(C,ENTER			)	
(2)	(101)				(NS)	(END			)	
(3)	(102)			(F,ATTNID)	(NE)	(C,PF006			)	
(4)	(103)				(NS)	(END			)	
(5)	(104)			(F,ATTNID)	(NE)	(C,PF007			)	
(6)	(105)				(NS)	(END			)	
(7)	(106)			(F,ATTNID)	(NE)	(C,PF010			)	
(8)	(107)				(NS)	(END			)	
(9)	(108)				(MS)	(C,INVALID KEY PRESSED - RETRY			)	

** MK5BX01 TYPE 0 DECODING REQUEST VALDCUPD FOR SCREEN CUSTUPD4 .

DEC 05, 2003 13.26.45

 * REQUEST NAME - FINDX *
 * INPUT STREAM REQUEST *

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(S)
COMMENT	*****	; THIS PROCEDURE MAKES SURE THAT ONLY ONE 'X' IS ENTERED	*****				*****
COMMENT	*****	; WHEN PF6 IS PRESSED ON SCREEN CUSTUPD4	*****				*****
COMMENT	*****	;	*****				*****
COMMENT	*****	IF F.ATTNID EQ 'PF006' AND,	*****				*****
COMMENT	*****	S.ACTION EQ 'X'	*****				*****
COMMENT	*****	LET T.COUNTX = T.COUNTX + 1	*****				*****
COMMENT	*****	IF T.COUNTX > 1	*****				*****
COMMENT	*****	MSG 'ONLY ONE 'X'' PER PF6'	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****		*****				*****

STMT NUMB	SEQ NO.	LOG LEV	CON CIR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER
(1)	(100)			(F,ATTNID)	(EQ)	(C,PF006			)	
(2)	(101)				(NS)	(END			)	
(3)	(102)			(S,ACTION)	(EQ)	(C,X			)	
(4)	(103)				(NS)	(END			)	
(5)	(104)			(T,COUNTX)	(+)	(D,1		(T,COUNTX)	)	
(6)	(105)			(T,COUNTX)	(GT)	(D,1			)	
(7)	(106)				(NS)	(END			)	
(8)	(107)				(MS)	(C,ONLY ONE 'X' PER PF6			)	

** MK5BX01 TYPE 0 DECODING REQUEST FINDX FOR SCREEN CUSTUPD4 .

DEC 05, 2003 13.26.45

 * REQUEST NAME - ONEXCUPD *
 * INPUT STREAM REQUEST *

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(S)
COMMENT	*****	; THIS PROCEDURE MAKES SURE THAT AT LEAST ONE 'X'	*****				*****
COMMENT	*****	; WAS ENTERED IF PF6 WAS PRESSED.	*****				*****
COMMENT	*****	;	*****				*****
COMMENT	*****	IF F.ATTNID EQ 'PF006' AND,	*****				*****
COMMENT	*****	T.COUNTX NE 1	*****				*****
COMMENT	*****	MSG 'AT LEASET ONE 'X'' PER PF6'	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****		*****				*****

STMT NUMB	SEQ NO.	LOG LEV	CON CIR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER
(1)	(100)			(F,ATTNID)	(EQ)	(C,PF006			)	
(2)	(101)				(NS)	(END			)	
(3)	(102)			(T,COUNTX)	(NE)	(D,1			)	

Figure A-2 Program Listings for Sample Applications (Page 50 of 84)

```

( 4) (103)                (NS)      (END                                )
( 5) (104)                (MS)      (C,AT LEASET ONE 'X' PER PF6      )
** MK5BX01 TYPE 0 DECODING REQUEST ONEXCUPD FOR SCREEN CUSTUPT4 .
*****
DEC 05, 2003 13.26.45      * REQUEST NAME - SYNCLOAD *
                          * INPUT STREAM REQUEST *
                          *****

-----
SIMT CAT                      REQ SCREEN BACK BR RNT      LINE
TYPE ACT REQUEST DESCRIPTION  TYP  NAME  CONTROL TMP     NUMBER

(ER)                                                                    (O)
COMMENT ***** ; THIS PROCEDURE SETS UP THE SYNCHRONIZATION FIELDS FOR *****
COMMENT ***** ; THE ITEM BROWSE WHEN APPROPRIATE. *****
COMMENT ***** ; *****
COMMENT ***** IF F.ATTNID EQ 'PF006' *****
COMMENT ***** LET T.SYNCFD1 = ITEMORD *****
COMMENT ***** LET T.SYNCFD2 = ITEMORD *****
COMMENT ***** END *****
COMMENT ***** *****

-----
SIMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD  SEQ
NUMB NO.  LEV CTR QLF FIELD NAME ATN  QLF  FIELD NAME OR CONSTANT  QLF FIELD NAME STRT NUMB OPR  NUMBER

( 1) (100)          (F,ATTNID ) (EQ) (C,PF006                                )
( 2) (101)          (NS)      (END                                )
( 3) (102)          (R )      (ITEMORD                                ) (T,SYNCFD1)
( 4) (103)          (R )      (ITEMORD                                ) (T,SYNCFD2)
** MK5BX01 TYPE 0 DECODING REQUEST SYNCLOAD FOR SCREEN ORUPT4 .

*****
DEC 05, 2003 13.26.45      * REQUEST NAME - OUR1N1 *
                          * INPUT STREAM REQUEST *
                          *****

-----
SIMT CAT                      REQ SCREEN BACK BR RNT      LINE
TYPE ACT REQUEST DESCRIPTION  TYP  NAME  CONTROL TMP     NUMBER

(ER)                                                                    (1)
COMMENT ***** ; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED *****
COMMENT ***** ; BY THE OPERATOR FOR SCREEN ORUPT4 *****
COMMENT ***** ; *****
COMMENT ***** IF F.ATTNID EQ 'ENTER', 'PF001', 'PF010' *****
COMMENT ***** ; IF PF1 OR PF10 HAVE BEEN PRESSED, MAKE SURE ACTION IS BL *****
COMMENT ***** IF F.ATTNID NE 'ENTER' *****
COMMENT ***** LET S.ACTION = ' ' *****
COMMENT ***** END *****
COMMENT ***** ELSE *****
COMMENT ***** MSG 'INVALID KEY PRESSED - RETRY' *****
COMMENT ***** END *****
COMMENT ***** *****

-----
SIMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD  SEQ
NUMB NO.  LEV CTR QLF FIELD NAME ATN  QLF  FIELD NAME OR CONSTANT  QLF FIELD NAME STRT NUMB OPR  NUMBER

( 1) (100)          (F,ATTNID ) (EQ) (C,ENTER                                )
( 2) (101)          (GS)      (106                                )
( 3) (102)          (F,ATTNID ) (EQ) (C,PF001                                )
( 4) (103)          (GS)      (106                                )
( 5) (104)          (F,ATTNID ) (EQ) (C,PF010                                )
( 6) (105)          (NS)      (110                                )
( 7) (106)          (F,ATTNID ) (NE) (C,ENTER                                )
( 8) (107)          (NS)      (END                                )
( 9) (108)          (R )      (C)                                ) (S,ACTION )
(10) (109)          (GO)      (END                                )
(11) (110)          (MS)      (C,INVALID KEY PRESSED - RETRY      )
** MK5BX01 TYPE 0 DECODING REQUEST OUR1N1 FOR SCREEN ORUPT4 .

*****
DEC 05, 2003 13.26.45      * REQUEST NAME - OUR2N1 *
                          * INPUT STREAM REQUEST *
                          *****

```

Figure A-2 Program Listings for Sample Applications (Page 51 of 84)

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK CONTROL	BR RNT	LINE NUMBER
(ER)							(S)
COMMENT	*****	; THIS PROCEDURE MAKES SURE THAT AN ITEM (WHOSE QUANTITY	*****				*****
COMMENT	*****	; ORDERED HAS BEEN CHANGED) HASN'T ALREADY BEEN SHIPPED.	*****				*****
COMMENT	*****	;	*****				*****
COMMENT	*****	IF F.ATTNID EQ 'ENTER' AND S.ACTION EQ 'U'	*****				*****
COMMENT	*****	; DOES THIS ITEM ALREADY EXIST ON THIS ORDER	*****				*****
COMMENT	*****	IF ITEMORD EQ ITEMORD	*****				*****
COMMENT	*****	; IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED	*****				*****
COMMENT	*****	IF S.ITMOTYOR EQ ITMOTYOR	*****				*****
COMMENT	*****	CONTINUE	*****				*****
COMMENT	*****	ELSE	*****				*****
COMMENT	*****	; IF NOT, HAS THE ITEM BEEN SHIPPED?	*****				*****
COMMENT	*****	IF ITEMORD EQ ITEMSHIP	*****				*****
COMMENT	*****	LET T.MSG = 'CANNOT CHANGE QUANTITY ORDERED '	*****				*****
COMMENT	*****	LET PF(T.MSG,32,31) = '- ITEM ALREADY SHIPPED'	*****				*****
COMMENT	*****	SET S.ITEMORD	*****				*****
COMMENT	*****	MSG T.MSG	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****	IF S.ITMPRICE EQ ' '	*****				*****
COMMENT	*****	LET S.ITMPRICE = 1.ITMPRICE	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****	END	*****				*****

STMT NUMB	SEQ NO.	LOG LEV	CON CIR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD NUMB	OPR	SEQ NUMBER
(1)	(100)			(F,ATTNID)	(EQ)	(C,ENTER				)	
(2)	(101)				(NS)	(END				)	
(3)	(102)			(S,ACTION)	(EQ)	(C,U				)	
(4)	(103)				(NS)	(END				)	
(5)	(104)			(ITEMORD)	(EQ)	(ITEMORD				)	
(6)	(105)				(NS)	(115				)	
(7)	(106)			(S,ITMOTYOR)	(EQ)	(ITMOTYOR				)	
(8)	(107)				(NS)	(109				)	
(9)	(108)				(GO)	(END				)	
(10)	(109)			(ITEMORD)	(EQ)	(ITEMSHIP				)	
(11)	(110)				(NS)	(115				)	
(12)	(111)				(R)	(C,CANNOT CHANGE QUANTITY ORDERED)	(T,MSG			)	
(13)	(112)				(R)	(C,- ITEM ALREADY SHIPPED)	(T,MSG		(32)	(31) (C)	
(14)	(113)			(S,ITEMORD)	(SE)						
(15)	(114)				(MS)	(T,MSG				)	
(16)	(115)			(S,ITMPRICE)	(EQ)	(C)				)	
(17)	(116)				(NS)	(END				)	
(18)	(117)				(R)	(1,ITMPRICE				) (S,ITMPRICE)	
** MK5BX01 TYPE 0 DECODING REQUEST OUR2N1 FOR SCREEN ORUPT4 .											

DEC 05, 2003 13.26.45

 * REQUEST NAME - OUR2N2 *
 * INPUT STREAM REQUEST *

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK CONTROL	BR RNT	LINE NUMBER
(ER)							(S)
COMMENT	*****	; THIS PROCEDURE CONTROLS THE UPDATE OF THE QUANTITY ON	*****				*****
COMMENT	*****	; HAND FIELD IN THE ITEM DATA BASE	*****				*****
COMMENT	*****	;	*****				*****
COMMENT	*****	IF F.ATTNID EQ 'ENTER' AND S.ACTION EQ 'U'	*****				*****
COMMENT	*****	; IS ITEM ALREADY ON THIS ORDER?	*****				*****
COMMENT	*****	IF ITEMORD EQ ITEMORD	*****				*****
COMMENT	*****	;IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED	*****				*****
COMMENT	*****	IF S.ITMOTYOR EQ ITMOTYOR	*****				*****
COMMENT	*****	CONTINUE	*****				*****
COMMENT	*****	ELSE	*****				*****
COMMENT	*****	LET 1.QANONND = 1.QANONND + ITMOTYOR	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****	END	*****				*****
COMMENT	*****	LET T.QANONND = 1.QANONND - S.ITMOTYOR	*****				*****
COMMENT	*****	IF T.QANONND LT 0	*****				*****

Figure A-2 Program Listings for Sample Applications (Page 52 of 84)

```

COMMENT ***** IF ITEMORD EQ ITEMORD *****
COMMENT ***** LET 1.QANONHND = 1.QANONHND - ITMQTYOR *****
COMMENT ***** END *****
COMMENT ***** LET 1.QANONHND = T.QANONHND * -1 *****
COMMENT ***** LET T.MSG = 'QUANTITY ORDERED IS ' *****
COMMENT ***** LET PF(T.MSG,21,7) = T.QANONHND *****
COMMENT ***** LET PF(T.MSG,31,31) = 'MORE THAN QUANTITY ON HAND' *****
COMMENT ***** SET S.ITMQTYOR *****
COMMENT ***** MSG T.MSG *****
COMMENT ***** END *****
COMMENT ***** LET 1.QANONHND = T.QANONHND *****
COMMENT ***** END *****
COMMENT *****

```

STMT NUMB	SEQ NO.	LOG LEV	CON CIR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	...RESULT... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER

(1)	(100)			(F,ATTNID)	(EQ)	(C,ENTER				
(2)	(101)				(NS)	(END				
(3)	(102)			(S,ACTION)	(EQ)	(C,U				
(4)	(103)				(NS)	(END				
(5)	(104)			(ITEMORD)	(EQ)	(ITEMORD				
(6)	(105)				(NS)	(110				
(7)	(106)			(S,ITMQTYOR)	(EQ)	(ITMQTYOR				
(8)	(107)				(NS)	(109				
(9)	(108)				(GO)	(END				
(10)	(109)			(1,QANONHND)	(+)	(ITMQTYOR		(1,QANONHND)		
(11)	(110)			(1,QANONHND)	(-)	(S,ITMQTYOR		(T,QANONHND)		
(12)	(111)			(T,QANONHND)	(LT)	(D,0				
(13)	(112)				(NS)	(122				
(14)	(113)			(ITEMORD)	(EQ)	(ITEMORD				
(15)	(114)				(NS)	(116				
(16)	(115)			(1,QANONHND)	(-)	(ITMQTYOR		(1,QANONHND)		
(17)	(116)			(T,QANONHND)	(*)	(D,-1		(1,QANONHND)		
(18)	(117)				(R)	(C,QUANTITY ORDERED IS		(T,MSG)		
(19)	(118)				(R)	(T,QANONHND		(T,MSG)	(21) (7) (C)	
(20)	(119)				(R)	(C,MORE THAN QUANTITY ON HAND		(T,MSG)	(31) (31) (C)	
(21)	(120)			(S,ITMQTYOR)	(SE)					
(22)	(121)				(MS)	(T,MSG				
(23)	(122)				(R)	(T,QANONHND		(1,QANONHND)		

** MK5BX01 TYPE 0 DECODING REQUEST OUR2N2 FOR SCREEN ORCUPDT4 .

DEC 05, 2003 13.26.45

* REQUEST NAME - VLDORDAD *
* INPUT STREAM REQUEST *

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER

(ER)							(1)
COMMENT *****		; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED					*****
COMMENT *****		; FOR THE ORDADOR SCREEN					*****
COMMENT *****		;					*****
COMMENT *****		IF F.ATTNID NE 'ENTER', 'PF001', 'PF006', 'PF010'					*****
COMMENT *****		MSG 'FUNCTION KEY INVALID - RETRY'					*****
COMMENT *****		END					*****

STMT NUMB	SEQ NO.	LOG LEV	CON CIR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	...RESULT... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER

(1)	(100)			(F,ATTNID)	(NE)	(C,ENTER				
(2)	(101)				(NS)	(END				
(3)	(102)			(F,ATTNID)	(NE)	(C,PF001				
(4)	(103)				(NS)	(END				
(5)	(104)			(F,ATTNID)	(NE)	(C,PF006				
(6)	(105)				(NS)	(END				
(7)	(106)			(F,ATTNID)	(NE)	(C,PF010				
(8)	(107)				(NS)	(END				
(9)	(108)				(MS)	(C,FUNCTION KEY INVALID - RETRY				

** MK5BX01 TYPE 0 DECODING REQUEST VLDORDAD FOR SCREEN ORDADDA .

DEC 05, 2003 13.26.45

* REQUEST NAME - QAR2N1 *

Figure A-2 Program Listings for Sample Applications (Page 53 of 84)

```

* INPUT STREAM REQUEST *
*****

-----
STMT  CAT                                REQ  SCREEN  BACK BR RNT  LINE
TYPE ACT REQUEST DESCRIPTION            TYP  NAME    CONTROL TMP  NUMBER
-----
(ER)
COMMENT ***** ; IF PRICE HAS NOT BEEN ENTERED ON THIS NEW ITEM, GET      ***** (2)
COMMENT ***** ; CURRENT PRICE FROM THE DATA BASE.                      *****
COMMENT ***** ;                                                                *****
COMMENT ***** IF S.ITMPRICE EQ ' '                                       *****
COMMENT ***** LET S.ITMPRICE = 1.ITMPRICE                               *****
COMMENT ***** END                                                         *****
COMMENT *****                                                             *****

-----
STMT  SEQ  LOG CON  ..OPERAND A... OPER  .....OPERAND B.....  ....RESULT.... PARTIAL FIELD  SEQ
NUMB NO.  LEV CTR QLF FIELD NAME  AIN  QLF  FIELD NAME OR CONSTANT  QLF FIELD NAME STRT NUMB OPR  NUMBER
-----
( 1) (100)          (S,ITMPRICE)  (EQ)  (C)
( 2) (101)          (NS)          (END)
( 3) (102)          (R )          (1,ITMPRICE)
** MK5BX01 TYPE 0 DECODING REQUEST OUR2N1 FOR SCREEN ORDADD4 .

DEC 05, 2003 13.26.45
*****
* REQUEST NAME - OUTYPE2 *
* INPUT STREAM REQUEST *
*****

-----
STMT  CAT                                REQ  SCREEN  BACK BR RNT  LINE
TYPE ACT REQUEST DESCRIPTION            TYP  NAME    CONTROL TMP  NUMBER
-----
(ER)
COMMENT ***** ; CALL TYPE2 PROCEDURES FOR ORCUPDT4                      ***** (2)
COMMENT ***** ;                                                                *****
COMMENT ***** CALL OUR2N1                                                *****
COMMENT ***** CALL OUR2N2                                                *****
COMMENT *****                                                             *****

-----
STMT  SEQ  LOG CON  ..OPERAND A... OPER  .....OPERAND B.....  ....RESULT.... PARTIAL FIELD  SEQ
NUMB NO.  LEV CTR QLF FIELD NAME  AIN  QLF  FIELD NAME OR CONSTANT  QLF FIELD NAME STRT NUMB OPR  NUMBER
-----
( 1) (100)          (GO)          (SUB OUR2N1)
( 2) (101)          (GO)          (SUB OUR2N2)
** MK5BX01 TYPE 0 DECODING REQUEST OUTYPE2 FOR SCREEN ORCUPDT4 .

DEC 05, 2003 13.26.45
*****
* REQUEST NAME - CUTYPE1 *
* INPUT STREAM REQUEST *
*****

-----
STMT  CAT                                REQ  SCREEN  BACK BR RNT  LINE
TYPE ACT REQUEST DESCRIPTION            TYP  NAME    CONTROL TMP  NUMBER
-----
(ER)
COMMENT ***** ; CALL TYPE 1 PROCEDURES FOR CUSTUPD4                      ***** (1)
COMMENT ***** ;                                                                *****
COMMENT ***** CALL VALDCUPD                                              *****
COMMENT ***** CALL FINDX                                                 *****
COMMENT ***** CALL ONEXCUPD                                              *****
COMMENT *****                                                             *****

-----
STMT  SEQ  LOG CON  ..OPERAND A... OPER  .....OPERAND B.....  ....RESULT.... PARTIAL FIELD  SEQ
NUMB NO.  LEV CTR QLF FIELD NAME  AIN  QLF  FIELD NAME OR CONSTANT  QLF FIELD NAME STRT NUMB OPR  NUMBER
-----
( 1) (100)          (GO)          (SUB VALDCUPD)
( 2) (101)          (GO)          (SUB FINDX)
( 3) (102)          (GO)          (SUB ONEXCUPD)
** MK5BX01 TYPE 0 DECODING REQUEST CUTYPE1 FOR SCREEN CUSTUPD4 .

```

Figure A-2 Program Listings for Sample Applications (Page 54 of 84)

```

** MK5TJ05 TYPE 0 THE FOLLOWING CICS TS SYSTEM DEFINITION UTILITY INPUT STATEMENTS ARE REQUIRED FOR THIS APPLICATION.
    DEFINE GROUP(???),DUMP(NO),TWASIZE(00000),TRANSACTION(M501),
    INDOUBT(BACKOUT),PROGRAM(M501)
    DEFINE GROUP(???),LANGUAGE(ASSEMBLER),PROGRAM(M501)
** MK5TJ03 TYPE 0 THE IMS PSB GENERATION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -
    PCB TYPE=DB,POS=M,KEYLEN=0021,PROCOPT=AP,DBDNAME=CUSTOMDB
    SENSEG NAME=CUSTOMER
    SENSEG NAME=ORDER,PARENT=CUSTOMER
    SENSEG NAME=SHIPINV,PARENT=ORDER
    SENSEG NAME=ITEMSHIP,PARENT=SHIPINV
    SENSEG NAME=ITEMORD,PARENT=ORDER
    PCB TYPE=DB,POS=M,KEYLEN=0007,PROCOPT=AP,DBDNAME=ITEMDB
    SENSEG NAME=ITEMROOT
    PCB TYPE=DB,POS=M,KEYLEN=0007,PROCOPT=GP,DBDNAME=ITEMDB
    SENSEG NAME=ITEMROOT
    PSBGEN LANG=ASSEM,MAXQ=016,PSENAME=M14CSIT
    END
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - LASTOCC FIRSTOCC ERROR MENU BACKORD UPDATE .
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - SCRNSLCT NOSCREEN SCREENNO .

** MK5W204 TYPE 0 NUMBER OF MESSAGES PRINTED IS 31.
    M5INPUT - 194 RECORDS INPUT
    M5FUNCH - 1048 RECORDS OUTPUT
    M5MAP - 0 RECORDS OUTPUT
    M5TPTEL - 15 RECORDS OUTPUT
    M5PREP - 0 RECORDS OUTPUT
    M5COMMAR - 97 RECORDS OUTPUT
    M5LIST - 981 RECORDS OUTPUT
    75 TRACKS ASSIGNED TO M5LIB - 23 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 800.
** MK5W209 TYPE 0 17928 BYTES OF TYPE A STORAGE USED DURING APPLICATION GENERATION

```

RUN NAME	STMT TYPE	LIB OPEN	SYS DELIM	DEBUG LEVEL	SRC LST	LKED PGMS	SYS ROLLB	CICS KEYS	MSG PROC	POS PCB	RUN TYPE	TRG MON	TRG SYS	PSB NAME	PROGRAM NAME	SEQUENCE 73 TO 80
(M502	,RC)		(#)					(C)			(G)	(C)		(M14CSIT)		

RUN NAME	STMT TYPE	FILE NAME	FILE TYPE	SYNCHRONIZING FIELD 1	FIELD 2	SYNC QUAL	SYNC TYPE	DL/I PATH	RECORD LENGTH	FLD	STOR TYPE	SQL AUTH-ID	SEQUENCE 73 TO 80
(M502	,RF)	(CUSTOMER	(FILE)									(U)	
(M502	,RF)	(ITEMX)	(FILE1)									(U)	
(M502	,RF)	(ITEMX)	(FILE2)	(SYNCFLD1)	(SYNCFLD2)		(T)						

```

***** BEGIN ASL PROCEDURE SCRNEKOD.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE CALCULATES THE MAXIMUM NUMBER OF SCREENS OF
; BACK-ORDERS POSSIBLE SHOULD THE TERMINAL OPERATOR CHOOSE
; AN ACTION OF 'B' WHICH INVOKES THE ITEMKOD SCREEN.
;
IF T.SCRNSLCT EQ 'B'
  LET T.NOSCREEN = 0
  LET T.SCREENNO = 0
  IF 1.BKORDRNO EQ 1.BKORDRNO ;BEGIN LOOP FOR ALL BACKORDERS
    LET T.COUNTOCC = T.COUNTOCC + 1
    IF T.COUNTOCC > 8
      LET T.NOSCREEN = T.NOSCREEN + 1
      LET T.COUNTOCC = 0
    END
  END
END
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST SCRNEKOD.

***** BEGIN ASL PROCEDURE IBTYPE1 .
***** Using ASL Translator Version 4.50 *****
; TYPE 1 CONTROL PROCEDURE FOR ITIMERWS4 SCREEN
;
CALL SAVEACTN
CALL VALDSLCT
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST IBTYPE1 .

```

Figure A-2 Program Listings for Sample Applications (Page 55 of 84)

```

***** BEGIN ASL PROCEDURE SAVEACTN.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE SAVES THE ACTION ENTERED BY THE
; TERMINAL OPERATOR INTO T.SCRNSLCT
;
IF S.ACTION NE ' '
LET T.SCRNSLCT = S.ACTION
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST SAVEACTN.

```

```

***** BEGIN ASL PROCEDURE VALDSLCT.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE VALIDATES THE ACTION ENTERED BY THE
; TERMINAL OPERATOR BY CHECKING THE VALUE SAVED IN
; T.SCRNSLCT BY THE PREVIOUS PROCEDURE.
;
IF T.SCRNSLCT NE ' ', 'U', 'B', 'M'
LET T.SCRNSLCT = T.ERROR
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST VALDSLCT.

```

```

***** BEGIN ASL PROCEDURE TEMPS .
***** Using ASL Translator Version 4.50 *****
; TEMP FIELD DEFINITIONS
SYNCFD1: FIELD C 7
SYNCFD2: FIELD C 7
SCREENNO: FIELD Z 2
NOSCREEN: FIELD Z 2
MSG: FIELD C 74
SCRNSLCT: FIELD C 1
UPDATE: FIELD C 1 INIT 'U'
BACKORD: FIELD C 1 INIT 'B'
MENU: FIELD C 1 INIT 'M'
ERROR: FIELD C 1 INIT 'E'
FIRSTOCC: FIELD P 2
LASTOCC: FIELD P 2
COUNTX: FIELD P 2
QANONHND: FIELD C 7
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST TEMPS .

```

```

DEC 05, 2003 13.34.11
*****
* REQUEST NAME - TEMPS *
* INPUT STREAM REQUEST *
*****

```

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(N)
COMMENT	*****	; TEMP FIELD DEFINITIONS					*****
COMMENT	*****	SYNCFD1: FIELD C 7					*****
COMMENT	*****	SYNCFD2: FIELD C 7					*****
COMMENT	*****	SCREENNO: FIELD Z 2					*****
COMMENT	*****	NOSCREEN: FIELD Z 2					*****
COMMENT	*****	MSG: FIELD C 74					*****
COMMENT	*****	SCRNSLCT: FIELD C 1					*****
COMMENT	*****	UPDATE: FIELD C 1 INIT 'U'					*****
COMMENT	*****	BACKORD: FIELD C 1 INIT 'B'					*****
COMMENT	*****	MENU: FIELD C 1 INIT 'M'					*****
COMMENT	*****	ERROR: FIELD C 1 INIT 'E'					*****
COMMENT	*****	FIRSTOCC: FIELD P 2					*****
COMMENT	*****	LASTOCC: FIELD P 2					*****
COMMENT	*****	COUNTX: FIELD P 2					*****
COMMENT	*****	QANONHND: FIELD C 7					*****
COMMENT	*****						*****

STMT TYPE	FIELD NAME	FIELD FLD LNTH	DEC TYP	OUTPT PLC	EDIT LGTH	INITIAL VALUE	COMMENTS	SEQUENCE NUMBER
(TF)	(SYNCFD1)	(7)	(C)					
(TF)	(SYNCFD2)	(7)	(C)					
(TF)	(SCREENNO)	(2)	(Z)					
(TF)	(NOSCREEN)	(2)	(Z)					
(TF)	(MSG)	(74)	(C)					
(TF)	(SCRNSLCT)	(1)	(C)					

Figure A-2 Program Listings for Sample Applications (Page 56 of 84)

```

(TF) (UPDATE ) ( 1) (C)          (U          )
(TF) (BACKORD ) ( 1) (C)          (B          )
(TF) (MENU ) ( 1) (C)            (M          )
(TF) (ERROR ) ( 1) (C)           (E          )
(TF) (FIRSTOCC ) ( 2) (P)
(TF) (LASTOCC ) ( 2) (P)
(TF) (COUNIX ) ( 2) (P)
(TF) (QANONHND ) ( 7) (C)

```

DEC 05, 2003 13.34.11

```

*****
* SCREEN CONTROL *
*****

```

INPUT SCREEN	STMT TYPE	SEQ NO.	LOG LVL	CON CTR	OPERAND A.. QLF FIELD	OPER ATTN	OPERAND B..... QLF FIELD NAME/CONSTANT	OUTPUT SCREEN	.OPERAND C.. QLF FLD/CNST	PARTL FLD STR NM OP	SEQUENCE 73 TO 80
(ITMWRWS4)	(SC)						(XR) (1,IBTYPE1)				
(ITMWRWS4)	(SC)						(XR) (2,SCRNEKOD)				
(ITMWRWS4)	(SC)				(T,SCRNSLCT)		(EQ) (T,UPDATE)				
(ITMWRWS4)	(SC)						(OS))			(ITMUPDT4)	
(ITMWRWS4)	(SC)			(E)	(T,SCRNSLCT)		(EQ) (T,BACKORD)				
(ITMWRWS4)	(SC)						(OS))			(ITMBKOD4)	
(ITMWRWS4)	(SC)			(E)	(T,SCRNSLCT)		(EQ) (T,MENU)				
(ITMWRWS4)	(SC)						(OS))			(ORDMENU4)	
(ITMWRWS4)	(SC)			(E)	(T,SCRNSLCT)		(EQ) (T,ERROR)				
(ITMWRWS4)	(SC)						(OS))			(ENDSCRN4)	
(ITMWRWS4)	(SC)			(E)	(T,SCRNSLCT)		(EQ) (C,ERROR)				
(ITMWRWS4)	(SC)						(OS))			(ITMWRWS4) (C,L000)	
(ITMWRWS4)	(SC)			(E)							
(ITMWRWS4)	(SC)						(OS))			(ITMWRWS4)	

SCREEN NAME	STMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80
(ITMWRWS4)	(SD)	(R)		()	(@)			(M502)	(M502MAP)			

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80
					..6.....7.....8.....9.....0.....1.....	<-TEXT (58-114)	

```

(ITMWRWS4,SL) ( 1) (0) ( ITMWRWS4 NEXT SCREEN OF ITEMS=PA1 )
(ITMWRWS4,SL) ( 2) (0) ( 'U' ACTION=ITEM UPDATE )
(ITMWRWS4,SL) ( 3) (0) ( 'B' ACTION=BACK-ORDERS )
(ITMWRWS4,SL) ( 4) (0) ( 'M' ACTION=MENU )
(ITMWRWS4,SL) ( 5) (0) ( PRODUCT RANGE: _____ TO _____ )
(ITMWRWS4,SL) ( 5) (1) ( QUANTITY )
(ITMWRWS4,SL) ( 6) (0) ( ACT ITEM ITEM QUANTITY SHIPPED ORDERED )
(ITMWRWS4,SL) ( 6) (1) ( BACK )
(ITMWRWS4,SL) ( 7) (0) ( ION NUMBER PRICE ON-HAND YTD YTD )
(ITMWRWS4,SL) ( 7) (1) ( ORDERED )
(ITMWRWS4,SL) ( 9) (0) ( )
(ITMWRWS4,SL) ( 9) (1) ( )
(ITMWRWS4,SL) (10) (0) ( )
(ITMWRWS4,SL) (10) (1) ( )
(ITMWRWS4,SL) (11) (0) ( )
(ITMWRWS4,SL) (11) (1) ( )
(ITMWRWS4,SL) (12) (0) ( )
(ITMWRWS4,SL) (12) (1) ( )
(ITMWRWS4,SL) (13) (0) ( )
(ITMWRWS4,SL) (13) (1) ( )
(ITMWRWS4,SL) (14) (0) ( )
(ITMWRWS4,SL) (14) (1) ( )
(ITMWRWS4,SL) (15) (0) ( )
(ITMWRWS4,SL) (15) (1) ( )
(ITMWRWS4,SL) (16) (0) ( )
(ITMWRWS4,SL) (16) (1) ( )
(ITMWRWS4,SL) (17) (0) ( )
(ITMWRWS4,SL) (17) (1) ( )
(ITMWRWS4,SL) (18) (0) ( )
(ITMWRWS4,SL) (18) (1) ( )
(ITMWRWS4,SL) (19) (0) ( )
(ITMWRWS4,SL) (19) (1) ( )
(ITMWRWS4,SL) (20) (0) ( )
(ITMWRWS4,SL) (20) (1) ( )
(ITMWRWS4,SL) (21) (0) ( )

```

Figure A-2 Program Listings for Sample Applications (Page 57 of 84)

```

(ITMWRWS4,SL) ( 21) (1) ( _____ )
(ITMWRWS4,SL) ( 22) (0) ( _____ )
(ITMWRWS4,SL) ( 22) (1) ( _____ )
(ITMWRWS4,SL) ( 23) (0) ( _____ )
(ITMWRWS4,SL) ( 23) (1) ( _____ )
** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES
73 TO 80

(ITMWRWS4,SS) ( 1) ( 1)
-----
V I R C E P M N A L C PARTIAL
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ITMWRWS4,S0) ( 1) (T,SYNCFD1) (B) (P) (NM) (SYNCFD1) )
(ITMWRWS4,S0) ( 1) (T,SYNCFD2) (B) (P) (NM) (SYNCFD2) )
(ITMWRWS4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES
73 TO 80

(ITMWRWS4,SS) ( 2) ( 15)
-----
V I R C E P M N A L C PARTIAL
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ITMWRWS4,S0) ( 2) (I) (Y) (N) (NM) (ACTION )
(ITMWRWS4,S0) ( 2) (1,ITEMNO ) (I) (X) (NM) (ITEMNO )
(ITMWRWS4,S1) (M) (IF) (S,ACTION ) (EQ) (C,U )
(ITMWRWS4,S1) (M) (IF) (S,ACTION ) (EQ) (C,M )
(ITMWRWS4,S0) ( 2) (S) (2,ITEMNO ) (O)
(ITMWRWS4,S0) ( 2) (2,ITEMPRICE) (O) (NM) (ITEMPRICE )
(ITMWRWS4,S0) ( 2) (2,QANONHAND) (O) (NM) (QANONHAND )
(ITMWRWS4,S0) ( 2) (2,SHIPYTD ) (O) (NM) (SHIPYTD )
(ITMWRWS4,S0) ( 2) (2,ORDYTD ) (O) (NM) (ORDYTD )
(ITMWRWS4,S0) ( 2) (2,QTYBKORD) (O) (NM) (QTYBKORD )
-----
SCREEN STMT CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN PROC TERMINAL SEQUENCE
NAME TYPE ACTN TYPE DESIG DESIG FORMAT CONTROL CODE ID NULL DEVICE
73 TO 80

(ITMUPDT4) (SD) (R) ( _ ) ( @ ) (M504 ) (ITMUPDT)
-----
SCREEN STMT LINE CONT- EXIN .....1....2....3....4....5.... <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6....7....8....9....0....+....1.... <-TEXT (58-114) 73 TO 80

(ITMUPDT4,SL) ( 1) (0) ( ITMUPDT4 )
(ITMUPDT4,SL) ( 1) (1) (***** )
(ITMUPDT4,SL) ( 2) (0) ( )
(ITMUPDT4,SL) ( 2) (1) (* MODIFY RECORD=ENTER * )
(ITMUPDT4,SL) ( 3) (0) ( ITEM NUMBER )
(ITMUPDT4,SL) ( 3) (1) (* RECORD DELETE=PF11 * )
(ITMUPDT4,SL) ( 4) (0) ( PRICE )
(ITMUPDT4,SL) ( 4) (1) (* RETURN TO MENU=PF10 * )
(ITMUPDT4,SL) ( 5) (0) ( QUANTITY/BOX )
(ITMUPDT4,SL) ( 5) (1) (***** )
(ITMUPDT4,SL) ( 6) (0) ( QUANTITY ON HAND )
(ITMUPDT4,SL) ( 7) (0) ( SHIPPED YTD )
(ITMUPDT4,SL) ( 8) (0) ( ORDERED YTD )
(ITMUPDT4,SL) ( 9) (0) ( QUANTITY BACK-ORDERED )
(ITMUPDT4,SL) ( 10) (0) ( RE-ORDER POINT )
(ITMUPDT4,SL) ( 11) (0) ( PRODUCT CODE )
** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES
73 TO 80

```

Figure A-2 Program Listings for Sample Applications (Page 58 of 84)

```

-----
(ITMUPDT4,SS) (1) (1)
-----
V I R C E P M N A L C PARTIAL
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80
-----
(ITMUPDT4,S0) (1) (T,DELETE ) (B) (D) (Y,Y) (P) (NM) (DELETE )
(ITMUPDT4,S0) (1) (1,ITEMNO ) (B) (Y) (U) (NM) (ITEMNO )
(ITMUPDT4,S1) (D) (IF (F,ATINID ) (EQ) (C,PF011 )
(ITMUPDT4,S1) (X) (IF (F,ATINID ) (EQ) (C,PF010 )
(ITMUPDT4,S0) (1) (1,ITMPRICE) (B) (P) (NM) (ITMPRICE )
(ITMUPDT4,S0) (1) (1,PACKBOX ) (B) (P) (NM) (PACKBOX )
(ITMUPDT4,S0) (1) (1,QANONHND) (B) (P) (NM) (QANONHND )
(ITMUPDT4,S0) (1) (1,SHIPYTD ) (B) (P) (NM) (SHIPYTD )
(ITMUPDT4,S0) (1) (1,ORDYTD ) (B) (P) (NM) (ORDYTD )
(ITMUPDT4,S0) (1) (1,QTYBKORD) (O) (NM) (QTYBKORD )
(ITMUPDT4,S0) (1) (1,REORDEPT ) (B) (P) (NM) (REORDEPT )
(ITMUPDT4,S0) (1) (1,PRODCODE) (B) (P) (NM) (PRODCODE )
(ITMUPDT4,S0) (1) (A) (F,MSGLINE ) (H) (*SYSTEM*)
** M5X901 TYPE 0 TEMPORARY FIELD DELETE DEFINED - FORMAT C, LENGTH 1, DECIMALS 0, EDIT LENGTH 1.

```

```

-----
SCREEN STMT CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN PROC TERMINAL SEQUENCE
NAME TYPE ACIN TYPE DESIG DESIG FORMAT CONTROL CODE ID NULL DEVICE 73 TO 80
-----
(ITMEKOD4) (SD) (R) ( ) (E) (M503 ) (ITMEKOD)

```

```

SCREEN STMT LINE CONT- EXIN .....1.....2.....3.....4.....5..... <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6.....7.....8.....9.....0.....1..... <-TEXT (58-114) 73 TO 80
-----
(ITMEKOD4,SL) (1) (0) ( ITMEKOD4 )
(ITMEKOD4,SL) (3) (0) ( ITEM NUMBER _ PRICE _ QUANTITY )
(ITMEKOD4,SL) (3) (1) (/BOX _ QUANTITY ON HAND _ SHIPPED YTD _ )
(ITMEKOD4,SL) (4) (0) ( )
(ITMEKOD4,SL) (4) (1) ( ORDERED YTD _ QUANTITY BACK-ORDERED _ )
(ITMEKOD4,SL) (5) (0) ( )
(ITMEKOD4,SL) (5) (1) ( REORDER POINT _ PRODUCT CODE _ )
(ITMEKOD4,SL) (6) (0) ( ***** )
(ITMEKOD4,SL) (7) (0) ( )
(ITMEKOD4,SL) (7) (1) ( ***** )
(ITMEKOD4,SL) (8) (0) ( * SPECIFIED SCREEN OF BACK-ORDERS=SCREEN# + )
(ITMEKOD4,SL) (8) (1) ( ENTER * )
(ITMEKOD4,SL) (9) (0) ( * NEXT SCREEN OF BACK-ORDERS=ENTER )
(ITMEKOD4,SL) (9) (1) ( * )
(ITMEKOD4,SL) (10) (0) ( * FILL A BACK-ORDER='X' + PF8 )
(ITMEKOD4,SL) (10) (1) ( * )
(ITMEKOD4,SL) (11) (0) ( * ADD BACK-ORDER=PF9 RETURN TO MENU=PF10 )
(ITMEKOD4,SL) (11) (1) ( * )
(ITMEKOD4,SL) (12) (0) ( ***** )
(ITMEKOD4,SL) (12) (1) ( ***** )
(ITMEKOD4,SL) (13) (0) ( FILL BACK CUSTOMER ORDER )
(ITMEKOD4,SL) (13) (1) ( SCREEN _ BACK ORDER NUMBER NUMBER )
(ITMEKOD4,SL) (14) (0) ( _ OF _ )
(ITMEKOD4,SL) (14) (1) ( ORDERS NUMBER )
(ITMEKOD4,SL) (15) (0) ( )
(ITMEKOD4,SL) (16) (0) ( )
(ITMEKOD4,SL) (17) (0) ( )
(ITMEKOD4,SL) (18) (0) ( )
(ITMEKOD4,SL) (19) (0) ( )
(ITMEKOD4,SL) (20) (0) ( )
(ITMEKOD4,SL) (21) (0) ( )
(ITMEKOD4,SL) (22) (0) ( )
(ITMEKOD4,SL) (23) (0) ( )
** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

```

```

-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES 73 TO 80
-----
(ITMEKOD4,SS) (1) (1)

```

```

V I R C E P M N A L C PARTIAL
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE

```

Figure A-2 Program Listings for Sample Applications (Page 59 of 84)

NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D	I	R	T	D	M	T	V	N	O	P	QLF	FIELD	NAME	OR	CONSTANT	ST	NO	OP	73	TO	80															
(ITMEKOD4,S0)	(	1)		(1,ITEMNO	)	(B)		(Y)		(M)		(NM)	(ITEMNO	)						(IF	(F,ATINID	)	(EQ)	(C,PF010	)																			
(ITMEKOD4,S1)										(X)																																		
(ITMEKOD4,S0)	(	1)		(1,ITMPRICE	)	(O)							(NM)	(ITMPRICE	)																													
(ITMEKOD4,S0)	(	1)		(1,PACKBOX	)	(O)							(NM)	(PACKBOX	)																													
(ITMEKOD4,S0)	(	1)		(1,QANONHND	)	(O)							(NM)	(QANONHND	)																													
(ITMEKOD4,S0)	(	1)		(1,SHIPYTD	)	(O)							(NM)	(SHIPYTD	)																													
SCREEN	SIMT	SEG	S	V	SOURCE/DEST	/	Q	H	U	R	R	O	U	A	L	C	C	L	O	E	/V		OPERAND	B		PARTIAL	FIELD	SEQUENCE																
NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D	I	R	T	D	M	T	V	N	O	P	QLF	FIELD	NAME	OR	CONSTANT	ST	NO	OP	73	TO	80															
(ITMEKOD4,S0)	(	1)		(1,ORDYTD	)	(O)								(NM)	(ORDYTD	)																												
(ITMEKOD4,S0)	(	1)		(1,QTYEKORD	)	(O)								(NM)	(QTYEKORD	)																												
(ITMEKOD4,S0)	(	1)		(1,REORDPT	)	(O)								(NM)	(REORDPT	)																												
(ITMEKOD4,S0)	(	1)		(1,PRODCODE	)	(O)								(NM)	(PRODCODE	)																												
(ITMEKOD4,S0)	(	1)		(1,REQSCRN	)	(O)							(N)	(NM)	(REQSCRN	)																												
(ITMEKOD4,S0)	(	1)		(T,SCREENNO	)	(B)		(Y)		(P)			(NM)	(SCREENNO	)																													
(ITMEKOD4,S0)	(	1)		(T,NOSCREEN	)	(B)		(Y)		(P)			(NM)	(NOSCREEN	)																													
(ITMEKOD4,S0)	(	1)	(A)	(F,MSGLINE	)			(H)																				(*SYSTEM*)																
SCREEN	SIMT	SEGMENT	NO.	OF																								SEQUENCE																
NAME	TYPE	NO.	NO.	OCCURRENCES																								73	TO	80														
(ITMEKOD4,SS)	(	2)		(	8)																																							
SCREEN	SIMT	SEG	S	V	SOURCE/DEST	/	Q	H	U	R	R	O	U	A	L	C	C	L	O	E	/V		OPERAND	B		PARTIAL	FIELD	SEQUENCE																
NAME	TYPE	NO.	A	QLF	FLD	NAME	O	D	I	R	R	T	D	M	T	V	N	O	P	QLF	FIELD	NAME	OR	CONSTANT	ST	NO	OP	73	TO	80														
(ITMEKOD4,S0)	(	2)		(I)	(Y)								(N)	(NM)	(XFIELD	)																												
(ITMEKOD4,S0)	(	2)		(1,BKORDRNO	)	(B)							(X)	(NM)	(BKORDRNO	)																												
(ITMEKOD4,S1)													(M)	(IF	(S,XFIELD	)	(EQ)	(C,X	)																									
(ITMEKOD4,S0)	(	2)		(N,CUSTNO	)	(I)							(X)	(NM)	(CUSTNO	)																												
(ITMEKOD4,S1)													(M)	(IF	(S,XFIELD	)	(EQ)	(C,X	)																									
(ITMEKOD4,S0)	(	2)	(S)	(1,CUSTNO	)	(O)																																						
(ITMEKOD4,S0)	(	2)		(N,ORDERNO	)	(I)							(X)	(NM)	(ORDERNO	)																												
(ITMEKOD4,S1)													(M)	(IF	(S,XFIELD	)	(EQ)	(C,X	)																									
(ITMEKOD4,S0)	(	2)	(S)	(1,ORDERNO	)	(O)																																						
(ITMEKOD4,S0)	(	2)	(V)	(N,ITEMORD	)	(I)							(X)	(NM)	(ITEMORD	)																												
(ITMEKOD4,S1)													(M)	(IF	(S,XFIELD	)	(EQ)	(C,X	)																									
SCREEN	SIMT	CAT	GLOSS	FIELD	HILITE	AUTO	OUTPUT	TRANSACT	SCREEN	PROC	TERMINAL																	SEQUENCE																
NAME	TYPE	ACTN	TYPE	DESIG	DESIG	FORMAT	CONTROL	CODE	ID	NULL	DEVICE																	73	TO	80														
(ORDMENU4)	(SD)	(R)		(	)	(@)		(M510	)	(M510	)																																	
SCREEN	SIMT	LINE	CONT-	EXTN	+...1....+...2....+...3....+...4....+...5....+.	<-TEXT (1-57)																						SEQUENCE																
NAME	TYPE	NO.	INUE	ATTR	..6....+...7....+...8....+...9....+...0....+...1....	<-TEXT (58-114)																						73	TO	80														
(ORDMENU4,SL)	(	1)	(0)	(	ORDMENU4	)																																						
(ORDMENU4,SL)	(	3)	(0)	(	CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS:	)																																						
(ORDMENU4,SL)	(	5)	(0)	(	PF1---CUSTOMER ADD/UPDATE	)																																						
(ORDMENU4,SL)	(	7)	(0)	(	PF2---ITEM BROWSE BY PRODUCT KEY	)																																						
(ORDMENU4,SL)	(	9)	(0)	(	PF3---BACK-ORDERED ITEM	)																																						
(ORDMENU4,SL)	(	11)	(0)	(	PF4---ITEM ADD/UPDATE	)																																						
(ORDMENU4,SL)	(	13)	(0)	(	PF5---END TRANSACTION	)																																						
(ORDMENU4,SL)	(	16)	(0)	(	ENTER CUSTOMER OR ITEM NUMBER	)																																						
(ORDMENU4,SL)	(	18)	(0)	(	(ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2)	)																																						
(ORDMENU4,SL)	(	18)	(1)	(	IS USED)	)																																						
** M5W14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.																																												
SCREEN	SIMT	SEGMENT	NO.	OF																								SEQUENCE																
NAME	TYPE	NO.	NO.	OCCURRENCES																								73	TO	80														

Figure A-2 Program Listings for Sample Applications (Page 60 of 84)


```

COMMENT ***** IF T.SCRNSLCT EQ 'B' *****
COMMENT ***** LET T.NOSCREEN = 0 *****
COMMENT ***** LET T.SCREENNO = 0 *****
COMMENT ***** IF 1.BKORDRNO EQ 1.BKORDRNO ;BEGIN LOOP FOR ALL BA *****
COMMENT ***** LET T.COUNTTOCC = T.COUNTTOCC + 1 *****
COMMENT ***** IF T.COUNTTOCC > 8 *****
COMMENT ***** LET T.NOSCREEN = T.NOSCREEN + 1 *****
COMMENT ***** LET T.COUNTTOCC = 0 *****
COMMENT ***** END *****
COMMENT ***** END *****
COMMENT ***** END *****
COMMENT ***** *****

```

STMT	SEQ	LOG CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD	SEQ						
NUMB	NO.	LEV	CTR	QLF	FIELD NAME	ATN	QLF	FIELD NAME OR CONSTANT	QLF	FIELD NAME	STRT	NUMB	OPR	NUMBER
(	1	(100)		(T,SCRNSLCT)	(EQ)	(C,B		)						
(	2	(101)			(NS)	(END		)						
(	3	(102)			(R)	(D,0		)		(T,NOSCREEN)				
(	4	(103)			(R)	(D,0		)		(T,SCREENNO)				
(	5	(104)		(1,BKORDRNO)	(EQ)	(1,BKORDRNO		)						
(	6	(105)			(NS)	(END		)						
(	7	(106)		(T,COUNTTOCC)	(+)	(D,1		)		(T,COUNTTOCC)				
(	8	(107)		(T,COUNTTOCC)	(GT)	(D,8		)						
(	9	(108)			(NS)	(END		)						
(	10	(109)		(T,NOSCREEN)	(+)	(D,1		)		(T,NOSCREEN)				
(	11	(110)			(R)	(D,0		)		(T,COUNTTOCC)				

** MK5BX01 TYPE 0 DECODING REQUEST SCRNKOD FOR SCREEN ITMRWS4 .
** MK5DI02 TYPE 0 (S# 7) DEFINITION OF TEMPORARY FIELD COUNTTOCC - FORMAT P LENGTH 8 DECIMALS 0 EDIT LENGTH 20 .

DEC 05, 2003 13.34.11

* REQUEST NAME - IBTYPE1 *
* INPUT STREAM REQUEST *

STMT	CAT	REQ	SCREEN	BACK	BR	RNT	LINE
TYPE	ACT	REQUEST DESCRIPTION	TYP	NAME	CONTROL	TMP	NUMBER
(ER)							(1)
COMMENT *****		; TYPE 1 CONTROL PROCEDURE FOR ITMRWS4 SCREEN					*****
COMMENT *****		;					*****
COMMENT *****		CALL SAVEACTN					*****
COMMENT *****		CALL VALDSLCT					*****
COMMENT *****							*****

STMT	SEQ	LOG CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD	SEQ						
NUMB	NO.	LEV	CTR	QLF	FIELD NAME	ATN	QLF	FIELD NAME OR CONSTANT	QLF	FIELD NAME	STRT	NUMB	OPR	NUMBER
(	1	(100)			(GO)	(SUB SAVEACTN		)						
(	2	(101)			(GO)	(SUB VALDSLCT		)						

** MK5BX01 TYPE 0 DECODING REQUEST IBTYPE1 FOR SCREEN ITMRWS4 .

DEC 05, 2003 13.34.11

* REQUEST NAME - SAVEACTN *
* INPUT STREAM REQUEST *

STMT	CAT	REQ	SCREEN	BACK	BR	RNT	LINE
TYPE	ACT	REQUEST DESCRIPTION	TYP	NAME	CONTROL	TMP	NUMBER
(ER)							(S)
COMMENT *****		; THIS PROCEDURE SAVES THE ACTION ENTERED BY THE					*****
COMMENT *****		; TERMINAL OPERATOR INTO T.SCRNSLCT					*****
COMMENT *****		;					*****
COMMENT *****		IF S.ACTION NE ' '					*****
COMMENT *****		LET T.SCRNSLCT = S.ACTION					*****
COMMENT *****		END					*****
COMMENT *****							*****

STMT	SEQ	LOG CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD	SEQ						
NUMB	NO.	LEV	CTR	QLF	FIELD NAME	ATN	QLF	FIELD NAME OR CONSTANT	QLF	FIELD NAME	STRT	NUMB	OPR	NUMBER

Figure A-2 Program Listings for Sample Applications (Page 62 of 84)

```

( 1) (100)      (S,ACTION ) (NE) (C)
( 2) (101)      (NS) (END )
( 3) (102)      (R ) (S,ACTION ) (T,SCRNSLCT)
** MK5EX01 TYPE 0 DECODING REQUEST SAVEACTN FOR SCREEN ITIMERWS4 .

```

DEC 05, 2003 13.34.11

```

*****
* REQUEST NAME - VALDSLCT *
* INPUT STREAM REQUEST *
*****

```

STMT	CAT	REQ	SCREEN	BACK	BR	RNT	LINE
TYPE	ACT	TYP	NAME	CONTROL	TMP		NUMBER

```

(ER) ***** (S)
COMMENT ***** ; THIS PROCEDURE VALIDATES THE ACTION ENTERED BY THE *****
COMMENT ***** ; TERMINAL OPERATOR BY CHECKING THE VALUE SAVED IN *****
COMMENT ***** ; T.SCRNSLCT BY THE PREVIOUS PROCEDURE. *****
COMMENT ***** ; *****
COMMENT ***** IF T.SCRNSLCT NE ' ', 'U', 'B', 'M' *****
COMMENT ***** LET T.SCRNSLCT = T.ERROR *****
COMMENT ***** END *****
COMMENT ***** *****

```

STMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD	SEQ
NUMB	NO.	LEV	CIR	QLF FIELD NAME	ATN	QLF FIELD NAME OR CONSTANT	QLF FIELD NAME	STMT NUMB	OPER

```

( 1) (100)      (T,SCRNSLCT) (NE) (C)
( 2) (101)      (NS) (END )
( 3) (102)      (T,SCRNSLCT) (NE) (C,U )
( 4) (103)      (NS) (END )
( 5) (104)      (T,SCRNSLCT) (NE) (C,B )
( 6) (105)      (NS) (END )
( 7) (106)      (T,SCRNSLCT) (NE) (C,M )
( 8) (107)      (NS) (END )
( 9) (108)      (R ) (T,ERROR ) (T,SCRNSLCT)
** MK5EX01 TYPE 0 DECODING REQUEST VALDSLCT FOR SCREEN ITIMERWS4 .

```

```

** MK5TJ05 TYPE 0 THE FOLLOWING CICS TS SYSTEM DEFINITION UTILITY INPUT STATEMENTS ARE REQUIRED FOR THIS APPLICATION.
      DEFINE GROUP(???) ,DUMP(NO),TWASIZE(00000),TRANSACTION(M502),
      INDOUBT(BACKOUT),PROGRAM(M502)

```

```

** MK5TJ03 TYPE 0 THE IMS PSB GENERATION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -
      PCB TYPE=DB, POS=M, KEYLEN=0017, PROCOPT=AP, DBDNAME=CUSTOMDB
      SENSEG NAME=CUSTOMER
      SENSEG NAME=ORDER, PARENT=CUSTOMER
      SENSEG NAME=ITEMORD, PARENT=ORDER
      PCB TYPE=DB, POS=M, KEYLEN=0014, PROCOPT=AP, DBDNAME=ITEMDB
      SENSEG NAME=ITEMROOT
      SENSEG NAME=BKORDERS, PARENT=ITEMROOT
      PCB TYPE=DB, POS=M, KEYLEN=0007, PROCOPT=GP, DBDNAME=ITEMDB
      SENSEG NAME=ITEMROOT
      PSBGEN LANG=ASSEM, MAXQ=015, PSBNAME=M14CSIT
      END

```

```

** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - QANONHND COUNIX LASTOCC FIRSTOCC MSG

```

```

** MK5M204 TYPE 0 NUMBER OF MESSAGES PRINTED IS 19.

```

```

M5INPUT - 81 RECORDS INPUT
M5PUNCH - 632 RECORDS OUTPUT
M5MAP - 0 RECORDS OUTPUT
M5TPTEL - 14 RECORDS OUTPUT
M5PREP - 0 RECORDS OUTPUT
M5COMMAR - 107 RECORDS OUTPUT
M5LIST - 628 RECORDS OUTPUT

```

```

** MK5M209 TYPE 0 75 TRACKS ASSIGNED TO M5LIB - 23 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 800.
16392 BYTES OF TYPE A STORAGE USED DURING APPLICATION GENERATION

```

RUN	STMT	LIB	SYS	DEBUG	SRC	LKED	SYS	CICS	MSG	POS	RUN	TRG	TRG	PSB	PROGRAM	SEQUENCE
NAME	TYPE	OPEN	DELIM	LEVEL	LST	PGMS	ROLLB	KEYS	PROC	PCB	TYPE	MON	SYS	NAME	NAME	73 TO 80

```

(M503 ,RC) (#) (C) (G) (C) (M14CSIT )

```

Figure A-2 Program Listings for Sample Applications (Page 63 of 84)

RUN NAME	STMT TYPE	FILE NAME	FILE TYPE	SYNCHRONIZING		SYNC QUAL	SYNC TYPE	DL/I PATH	RECORD LENGTH	FLD	STOR TYPE	SQL AUTH-ID	SEQUENCE 73 TO 80
				FIELD 1	FIELD 2								
(M503	,RF)	(CUSTOMER	(FILE)				(U)						
(M503	,RF)	(ITEMX)	(FILE1)				(U)						

```

***** BEGIN ASL PROCEDURE SCRNSSEL .
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE SELECTS THE DESIRED OCCURRENCES OF
; BACKORDERS FOR SCREEN DISPLAY, BASED ON T.SCREENNO
;
LET T.COUNTOCC = 0
LET T.LASTOCC = T.SCREENNO * 8
LET T.FIRSTOCC = T.LASTOCC - 7
IF 1.BKORDRNO EQ 1.BKORDRNO ;BEGIN LOOP FOR ALL BACKORDERS
LET T.COUNTOCC = T.COUNTOCC + 1
IF T.COUNTOCC >= T.FIRSTOCC
TRANSFER TO OUTPUT
END
ELSE
TRANSFER TO OUTPUT
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST SCRNSSEL .

***** BEGIN ASL PROCEDURE IBKODVLD.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED
; BY THE TERMINAL OPERATOR
;
IF F.ATTNID NE 'ENTER', 'PF008', 'PF009', 'PF010'
MSG 'INVALID KEY PRESSED - RETRY'
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST IBKODVLD.

***** BEGIN ASL PROCEDURE IBKODITM.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE MOVES THE ITEM NUMBER INTO THE VIRTUAL
; FIELD USED TO MATCH AGAINST THE CUSTOMER DATA BASE
; ITEMORD SEGMENT FOR OUTPUT TO THE FILLKOD SCREEN.
;
IF F.ATTNID EQ 'PF008' AND S.XFIELD EQ 'X'
LET S.ITEMORD = S.ITEMNO
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST IBKODITM.

***** BEGIN ASL PROCEDURE SCRNNOCK.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE CHECKS THE SCREEN NUMBER REQUESTED
; IF THE TERMINAL OPERATOR HAS PRESSED ENTER, WHETHER
; OR NOT HE HAS REQUESTED A SPECIFIC SCREEN NUMBER.
;
IF F.ATTNID EQ 'ENTER'
IF S.REQSCRN EQ ' '
LET T.SCREENNO = T.SCREENNO + 1
ELSE
LET T.SCREENNO = S.REQSCRN + 0
END
IF S.SCREENNO > S.NOSCREEN
MSG 'REQUESTED SCREEN NOT AVAILABLE'
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST SCRNNOCK.

***** BEGIN ASL PROCEDURE IBOTYPE1.
***** Using ASL Translator Version 4.50 *****
; TYPE 1 CONTROL PROCEDURE FOR ITIMEKOD4
;
CALL IBKODVLD
CALL IBKODITM
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST IBOTYPE1.

```

Figure A-2 Program Listings for Sample Applications (Page 64 of 84)

```

***** BEGIN ASL PROCEDURE TEMPS .
***** Using ASL Translator Version 4.50 *****
; TEMP FIELD DEFINITIONS
SYNCFD1: FIELD C 7
SYNCFD2: FIELD C 7
SCREENNO: FIELD Z 2
NOSCREEN: FIELD Z 2
MSG: FIELD C 74
SCRNSLCT: FIELD C 1
UPDATE: FIELD C 1 INIT 'U'
BACKORD: FIELD C 1 INIT 'B'
MENU: FIELD C 1 INIT 'M'
ERROR: FIELD C 1 INIT 'E'
FIRSTOCC: FIELD P 2
LASTOCC: FIELD P 2
COUNTX: FIELD P 2
QANONHND: FIELD C 7
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST TEMPS .

```

DEC 05, 2003 13.40.43

```

*****
* REQUEST NAME - TEMPS *
* INPUT STREAM REQUEST *
*****

```

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(N)
COMMENT	*****	; TEMP FIELD DEFINITIONS					*****
COMMENT	*****	SYNCFD1: FIELD C 7					*****
COMMENT	*****	SYNCFD2: FIELD C 7					*****
COMMENT	*****	SCREENNO: FIELD Z 2					*****
COMMENT	*****	NOSCREEN: FIELD Z 2					*****
COMMENT	*****	MSG: FIELD C 74					*****
COMMENT	*****	SCRNSLCT: FIELD C 1					*****
COMMENT	*****	UPDATE: FIELD C 1 INIT 'U'					*****
COMMENT	*****	BACKORD: FIELD C 1 INIT 'B'					*****
COMMENT	*****	MENU: FIELD C 1 INIT 'M'					*****
COMMENT	*****	ERROR: FIELD C 1 INIT 'E'					*****
COMMENT	*****	FIRSTOCC: FIELD P 2					*****
COMMENT	*****	LASTOCC: FIELD P 2					*****
COMMENT	*****	COUNTX: FIELD P 2					*****
COMMENT	*****	QANONHND: FIELD C 7					*****
COMMENT	*****						*****

STMT TYPE	FIELD NAME	FIELD LENGTH	FLD TYP	DEC PLC	OUTPT EDIT	EDIT LGTH	INITIAL VALUE	COMMENTS	SEQUENCE NUMBER
(TF)	(SYNCFD1)	(7)	(C)						
(TF)	(SYNCFD2)	(7)	(C)						
(TF)	(SCREENNO)	(2)	(Z)						
(TF)	(NOSCREEN)	(2)	(Z)						
(TF)	(MSG)	(74)	(C)						
(TF)	(SCRNSLCT)	(1)	(C)						
(TF)	(UPDATE)	(1)	(C)			(U)			
(TF)	(BACKORD)	(1)	(C)			(B)			
(TF)	(MENU)	(1)	(C)			(M)			
(TF)	(ERROR)	(1)	(C)			(E)			
(TF)	(FIRSTOCC)	(2)	(P)						
(TF)	(LASTOCC)	(2)	(P)						
(TF)	(COUNTX)	(2)	(P)						
(TF)	(QANONHND)	(7)	(C)						

DEC 05, 2003 13.40.43

```

*****
* SCREEN CONTROL *
*****

```

INPUT SCREEN	STMT TYPE	SEQ NO.	LOG LVL	CON CTR	.OPERAND A.. QLF FIELD	OPER ATTN	 QLF FIELD	.OPERAND B..... NAME/CONSTANT	OUTPUT SCREEN	.OPERAND C.. QLF FLD/CNST	PARTL FLD STR NM OP	SEQUENCE 73 TO 80
(ITMBKOD4)	(SC)					(XR)	(1,IBOTYPE1)					
(ITMBKOD4)	(SC)					(XR)	(2,SCRNDCK)					
(ITMBKOD4)	(SC)			(F,ATTNID)		(EQ)	(C,ENTER)					
(ITMBKOD4)	(SC)					(OS)	(O,SCRNSEL)				(ITMBKOD4)	

Figure A-2 Program Listings for Sample Applications (Page 65 of 84)

```

(ITMEKOD4) (SC)          (E) (F,ATTNID ) (EQ) (C,PF008          )
(ITMEKOD4) (SC)          (OS)          ) (FILBKOD4)
(ITMEKOD4) (SC)          (E) (F,ATTNID ) (EQ) (C,PF009          )
(ITMEKOD4) (SC)          (OS)          ) (ADDERKOD4)
(ITMEKOD4) (SC)          (E) (F,ATTNID ) (EQ) (C,PF010          )
(ITMEKOD4) (SC)          (OS)          ) (ORDMENU4)

```

SCREEN NAME	SMT TYPE	CAT ACIN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACT CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

```

(ITMEKOD4) (SD) (R)          ( ) (E)          (M503 ) (ITMEKOD)

```

SCREEN NAME	SMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80

```

(ITMEKOD4,SL) ( 1) (0)      ( ITMEKOD4                                     )
(ITMEKOD4,SL) ( 3) (0)      (      ITEM NUMBER _      PRICE _      QUANTITY )
(ITMEKOD4,SL) ( 3) (1)      (/BOX _      )
(ITMEKOD4,SL) ( 4) (0)      (      QUANTITY ON HAND _      SHIPPED YTD _      )
(ITMEKOD4,SL) ( 4) (1)      (      )
(ITMEKOD4,SL) ( 5) (0)      (      ORDERED YTD _      QUANTITY BACK-ORDERED _      )
(ITMEKOD4,SL) ( 5) (1)      (      )
(ITMEKOD4,SL) ( 6) (0)      (      REORDER POINT _      PRODUCT CODE _      )
(ITMEKOD4,SL) ( 7) (0)      (      ***** )
(ITMEKOD4,SL) ( 7) (1)      (      ***** )
(ITMEKOD4,SL) ( 8) (0)      (      * SPECIFIED SCREEN OF BACK-ORDERS=SCREEN# + )
(ITMEKOD4,SL) ( 8) (1)      ( ENTER * )
(ITMEKOD4,SL) ( 9) (0)      (      * NEXT SCREEN OF BACK-ORDERS=ENTER )
(ITMEKOD4,SL) ( 9) (1)      (      * )
(ITMEKOD4,SL) (10) (0)      (      * FILL A BACK-ORDER='X' + PF8 )
(ITMEKOD4,SL) (10) (1)      (      * )
(ITMEKOD4,SL) (11) (0)      (      * ADD BACK-ORDER=PF9 RETURN TO MENU=PF10 )
(ITMEKOD4,SL) (11) (1)      (      * )
(ITMEKOD4,SL) (12) (0)      (      ***** )
(ITMEKOD4,SL) (12) (1)      (      ***** )
(ITMEKOD4,SL) (13) (0)      (      FILL BACK CUSTOMER ORDER )
(ITMEKOD4,SL) (13) (1)      (      SCREEN _      )
(ITMEKOD4,SL) (14) (0)      (      BACK ORDER NUMBER NUMBER )
(ITMEKOD4,SL) (14) (1)      (      OF _      )
(ITMEKOD4,SL) (15) (0)      (      ORDERS NUMBER )
(ITMEKOD4,SL) (16) (0)      (      )
(ITMEKOD4,SL) (17) (0)      (      )
(ITMEKOD4,SL) (18) (0)      (      )
(ITMEKOD4,SL) (19) (0)      (      )
(ITMEKOD4,SL) (20) (0)      (      )
(ITMEKOD4,SL) (21) (0)      (      )
(ITMEKOD4,SL) (22) (0)      (      )
(ITMEKOD4,SL) (23) (0)      (      )
** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

```

SCREEN NAME	SMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

```

(ITMEKOD4,SS) ( 1)      ( 1)

```

SCREEN NAME	SMT TYPE	NO. A	V SEG S	SOURCE/DEST QLF FLD NAME	I R C E P M N / Q H U R R O U O D I R R T D M	A L C C L O E/V	OPERAND B.....	PARTIAL FIELD	SEQUENCE ST NO OP 73 TO 80

```

(ITMEKOD4,S0) ( 1)      (1,ITEMNO ) (B)      (Y)      (M)      (NM) (ITEMNO )
(ITMEKOD4,S1)          (X)      (IF) (F,ATTNID ) (EQ) (C,PF010 ) )
(ITMEKOD4,S0) ( 1)      (1,ITMPRICE) (O)          (NM) (ITMPRICE )
(ITMEKOD4,S0) ( 1)      (1,PACKBOX ) (O)          (NM) (PACKBOX )
(ITMEKOD4,S0) ( 1)      (1,QANONHND) (O)          (NM) (QANONHND )
(ITMEKOD4,S0) ( 1)      (1,SHIPYTD ) (O)          (NM) (SHIPYTD )
(ITMEKOD4,S0) ( 1)      (1,ORDYTD ) (O)          (NM) (ORDYTD )
(ITMEKOD4,S0) ( 1)      (1,QTYBKORD) (O)          (NM) (QTYBKORD )
(ITMEKOD4,S0) ( 1)      (1,REORDPT ) (O)          (NM) (REORDPT )
(ITMEKOD4,S0) ( 1)      (1,PRODCODE) (O)          (NM) (PRODCODE )
(ITMEKOD4,S0) ( 1)      (I)          (N)      (NM) (REQSCRN )
(ITMEKOD4,S0) ( 1)      (T,SCREENNO) (B)          (Y)      (P)      (NM) (SCREENNO )
(ITMEKOD4,S0) ( 1)      (T,NOSCREEN) (B)          (Y)      (P)      (NM) (NOSCREEN )
(ITMEKOD4,S0) ( 1) (A)      (F,MSGLINE ) (H)          (N)      (NM) (NOSCREEN )

```

(*SYSTEM*)

Figure A-2 Program Listings for Sample Applications (Page 66 of 84)

SCREEN NAME	SMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES											SEQUENCE 73 TO 80
(ITMEKOD4,SS) (2) (8)														
SCREEN NAME	SMT TYPE	SEG NO.	V A	SOURCE/DEST QLF FLD NAME	I R O D I R R T D M	C E P M N / Q H U R R O U O D I R R T D M	A L C C L O E/V T V N O P	OPERAND B..... QLF FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80				
(ITMEKOD4,S0)	(2)			(I)	(Y)		(N)	(NM) (XFIELD)						
(ITMEKOD4,S0)	(2)			(1,BKORDNO)	(B)		(X)	(NM) (BKORDNO)						
(ITMEKOD4,S1)							(M)	(IF) (S,XFIELD) (EQ) (C,X)						
(ITMEKOD4,S0)	(2)			(N,CUSTNO)	(I)		(X)	(NM) (CUSTNO)						
(ITMEKOD4,S1)							(M)	(IF) (S,XFIELD) (EQ) (C,X)						
(ITMEKOD4,S0)	(2)	(S)		(1,CUSTNO)	(O)									
(ITMEKOD4,S0)	(2)			(N,ORDERNO)	(I)		(X)	(NM) (ORDERNO)						
(ITMEKOD4,S1)							(M)	(IF) (S,XFIELD) (EQ) (C,X)						
(ITMEKOD4,S0)	(2)	(S)		(1,ORDERNO)	(O)									
(ITMEKOD4,S0)	(2)	(V)		(N,ITEMORD)	(I)		(X)	(NM) (ITEMORD)						
(ITMEKOD4,S1)							(M)	(IF) (S,XFIELD) (EQ) (C,X)						
SCREEN NAME	SMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80		
(FILEKOD4) (SD) (R) () (8)														
SCREEN NAME	SMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5..... <-TEXT (1-57)							SEQUENCE 73 TO 80		
.....6.....7.....8.....9.....0.....1..... <-TEXT (58-114)														
(FILEKOD4,SL)	(1)	(0)			(FILEKOD4 DO YOU WANT TO FILL)									
(FILEKOD4,SL)	(1)	(1)			(FILL BACK-ORDER='Y' OR)									
(FILEKOD4,SL)	(2)	(0)			(THIS BACK-ORDER (Y OR N)? _)									
(FILEKOD4,SL)	(2)	(1)			('N' + ENTER)									
(FILEKOD4,SL)	(3)	(1)			(RETURN TO MENU=PF10)									
(FILEKOD4,SL)	(4)	(0)			(ITEM NUMBER)									
(FILEKOD4,SL)	(5)	(0)			(BACK-ORDER NUMBER)									
(FILEKOD4,SL)	(6)	(0)			(ITEM NAME)									
(FILEKOD4,SL)	(7)	(0)			(PRICE)									
(FILEKOD4,SL)	(8)	(0)			(QUANTITY BACK-ORDERED)									
(FILEKOD4,SL)	(9)	(0)			(QUANTITY ON HAND)									
(FILEKOD4,SL)	(10)	(0)			(CUSTOMER NUMBER)									
(FILEKOD4,SL)	(11)	(0)			(CUSTOMER NAME)									
(FILEKOD4,SL)	(11)	(1)			()									
(FILEKOD4,SL)	(12)	(0)			(ORDER NUMBER)									
(FILEKOD4,SL)	(13)	(0)			(SHIPMENT NUMBER)									
(FILEKOD4,SL)	(14)	(0)			(SHIPPER)									
(FILEKOD4,SL)	(15)	(0)			(INVOICE NUMBER)									
(FILEKOD4,SL)	(16)	(0)			(DUE DATE)									
** M\$W14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.														
SCREEN NAME	SMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES											SEQUENCE 73 TO 80
(FILEKOD4,SS) (1) (1)														
SCREEN NAME	SMT TYPE	SEG NO.	V A	SOURCE/DEST QLF FLD NAME	I R O D I R R T D M	C E P M N / Q H U R R O U O D I R R T D M	A L C C L O E/V T V N O P	OPERAND B..... QLF FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80				
(FILEKOD4,S0)	(1)			(I)			(N)	(NM) (CHOICE)						
(FILEKOD4,S0)	(1)			(1,ITEMNO)	(B)	(Y)	(U)	(NM) (ITEMNO)						
(FILEKOD4,S0)	(1)			(1,BKORDNO)	(B)		(D)	(NM) (BKORDNO)						
(FILEKOD4,S1)							(X)	(IF) (S,CHOICE) (NE) (C,Y)						
(FILEKOD4,S0)	(1)			(1,ITEMNAME)	(O)		(NM)	(ITEMNAME)						
(FILEKOD4,S0)	(1)			(I)	(Y)		(N)	(NM) (ITEMPRICE)						
(FILEKOD4,S0)	(1)	(S)		(N,ITEMPRICE)	(O)									
(FILEKOD4,S0)	(1)			(1,QTYBKORD)	(I)	(Y)	(S)	(NM) (QTYBKORD)						
(FILEKOD4,S0)	(1)	(S)		(1,QANONHND)	(I)	(Y)	(S)							

Figure A-2 Program Listings for Sample Applications (Page 67 of 84)


```

(FILEKOD4,S0) ( 1) (S) (N,QTYBKORD) (O)
(FILEKOD4,S0) ( 1) (1,QANONHND) (O)
(FILEKOD4,S0) ( 1) (N,CUSTNO ) (B) (Y) (M) (NM) (CUSTNO )
(FILEKOD4,S0) ( 1) (N,CUSTNAME) (O) (NM) (CUSTNAME )
(FILEKOD4,S0) ( 1) (N,ORDERNO ) (B) (Y) (M) (NM) (ORDERNO )
(FILEKOD4,S0) ( 1) (N,SHIPNO ) (I) (I) (NM) (SHIPNO )
(FILEKOD4,S1) (X) (IF) (S,CHOICE ) (NE) (C,Y )
(FILEKOD4,S0) ( 1) (N,SHIPPER ) (I) (R) (NM) (SHIPPER )
(FILEKOD4,S0) ( 1) (N,INWNO ) (I) (R) (NM) (INWNO )
(FILEKOD4,S0) ( 1) (N,PAYDATE ) (I) (R) (NM) (PAYDATE )
(FILEKOD4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)

```

SCREEN NAME	SMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

```

(ADDEKOD4) (SD) (R) ( ) (0)

```

SCREEN NAME	SMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80

```

(ADDEKOD4,SL) ( 1) (0) ( ADDEKOD4 )
(ADDEKOD4,SL) ( 5) (0) ( CUSTOMER NUMBER )
(ADDEKOD4,SL) ( 7) (0) ( ORDER NUMBER )
(ADDEKOD4,SL) ( 9) (0) ( ITEM NUMBER )
(ADDEKOD4,SL) (11) (0) ( BACK-ORDER NUMBER )
(ADDEKOD4,SL) (13) (0) ( QUANTITY )
(ADDEKOD4,SL) (17) (0) ( $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ )
(ADDEKOD4,SL) (18) (0) ( $ ADD BACK-ORDER=ENTER $ )
(ADDEKOD4,SL) (19) (0) ( $ RETURN TO MENU=PF10 $ )
(ADDEKOD4,SL) (20) (0) ( $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ )
** M$W14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

```

SCREEN NAME	SMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

```

(ADDEKOD4,SS) ( 1) ( 1)

```

SCREEN NAME	SMT TYPE	SEG NO.	S A	V Q	DEST FLD	SOURCE NAME	I O	R D	C I	E R	P T	M D	N M	A T	L V	C OP	OPERAND FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO	SEQUENCE 73 TO 80

```

(ADDEKOD4,S0) ( 1) (1,CUSTNO ) (I) (P) (NM) (CUSTNO )
(ADDEKOD4,S1) (NE) (C)
(ADDEKOD4,S1) (MS) (C,CUSTOMER NUMBER REQUIRED)
(ADDEKOD4,S0) ( 1) (S) (N,CUSTNO ) (I) (X)
(ADDEKOD4,S1) (M) (IF) (F,ATTNID ) (EQ) (C,ENTER )
(ADDEKOD4,S0) ( 1) (1,ORDERNO ) (I) (P) (NM) (ORDERNO )
(ADDEKOD4,S1) (NE) (C)
(ADDEKOD4,S1) (MS) (C,ORDER NUMBER REQUIRED )
(ADDEKOD4,S0) ( 1) (S) (N,ORDERNO ) (I) (X)
(ADDEKOD4,S1) (M) (IF) (F,ATTNID ) (EQ) (C,ENTER )
(ADDEKOD4,S0) ( 1) (1,ITEMNO ) (B) (Y) (X) (NM) (ITEMNO )
(ADDEKOD4,S1) (U) (IF) (F,ATTNID ) (EQ) (C,ENTER )
(ADDEKOD4,S0) ( 1) (S) (N,ITEMORD ) (I) (Y) (X)
(ADDEKOD4,S1) (M) (IF) (F,ATTNID ) (EQ) (C,ENTER )
(ADDEKOD4,S0) ( 1) (1,BKORDRNO ) (I) (X) (NM) (BKORDRNO )
(ADDEKOD4,S1) (I) (IF) (F,ATTNID ) (EQ) (C,ENTER )
(ADDEKOD4,S0) ( 1) (N,QTYBKORD) (I) (A) (NM) (QTYBKORD )
(ADDEKOD4,S1) (NE) (C)
(ADDEKOD4,S1) (A,BQ) (QTYBKORD )
(ADDEKOD4,S1) (MS) (C,QUANTITY BACK-ORDERED IN)
(ADDEKOD4,S1) (MS) (C,CORRECT - REENTER )
(ADDEKOD4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)

```

SCREEN NAME	SMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

```

(ORDMENU4) (SD) (R) ( ) (0) (M510 ) (M510 )

```

Figure A-2 Program Listings for Sample Applications (Page 68 of 84)

```

-----
SCREEN STMT LINE CONT- EXTN .....1.....2.....3.....4.....5..... <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6.....7.....8.....9.....0.....1.... <-TEXT (58-114) 73 TO 80
-----

(ORDMENU4,SL) ( 1) (0) ( ORDMENU4 )
(ORDMENU4,SL) ( 3) (0) ( CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS:)
(ORDMENU4,SL) ( 5) (0) ( PF1---CUSTOMER ADD/UPDATE )
(ORDMENU4,SL) ( 7) (0) ( PF2---ITEM BROWSE BY PRODUCT KEY )
(ORDMENU4,SL) ( 9) (0) ( PF3---BACK-ORDERED ITEM )
(ORDMENU4,SL) (11) (0) ( PF4---ITEM ADD/UPDATE )
(ORDMENU4,SL) (13) (0) ( PF5---END TRANSACTION )
(ORDMENU4,SL) (16) (0) ( ENTER CUSTOMER OR ITEM NUMBER )
(ORDMENU4,SL) (18) (0) ( ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2)
(ORDMENU4,SL) (18) (1) ( IS USED)
** M$NL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----

SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES 73 TO 80
-----

(ORDMENU4,SS) ( 1) ( 1)
-----

SCREEN STMT SEG S V I R C E P M N A L C PARTIAL
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80 SEQUENCE

(ORDMENU4,S0) ( 1) (N,CUSTNO ) (I) (X) (NM) (ITEMNO )
(ORDMENU4,S1) (V) (IF) (S,PFKEY ) (EQ) (C,PF001 )
(ORDMENU4,S0) ( 1) (S) (1,ITEMNO ) (I) (X)
(ORDMENU4,S1) (M) (IF) (S,PFKEY ) (EQ) (C,PF003 )
(ORDMENU4,S1) (V) (IF) (S,PFKEY ) (EQ) (C,PF004 )
(ORDMENU4,S0) ( 1) (V) (I) (N) (NM) (PFKEY )
(ORDMENU4,S1) (LN) (5 )
(ORDMENU4,S1) (EQ) (C,PF001 )
(ORDMENU4,S1) (1,0,EQ) (C,PF002 )
(ORDMENU4,S1) (1,0,EQ) (C,PF003 )
(ORDMENU4,S1) (1,0,EQ) (C,PF004 )
(ORDMENU4,S1) (1,0,EQ) (C,PF005 )
(ORDMENU4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)

DEC 05, 2003 13.40.43 *****
* REQUEST NAME - SCRNSSEL *
* INPUT STREAM REQUEST *
*****

STMT CAT REQ SCREEN BACK BR RNT LINE
TYPE ACT REQUEST DESCRIPTION TYP NAME CONTROL TMP NUMBER

(ER) (O)
COMMENT ***** ; THIS PROCEDURE SELECTS THE DESIRED OCCURRENCES OF *****
COMMENT ***** ; BACKORDERS FOR SCREEN DISPLAY, BASED ON T.SCREENNO *****
COMMENT ***** ; *****
COMMENT ***** LET T.COUNTOCC = 0 *****
COMMENT ***** LET T.LASTOCC = T.SCREENNO * 8 *****
COMMENT ***** LET T.FIRSTOCC = T.LASTOCC - 7 *****
COMMENT ***** IF 1.BKORDRNO EQ 1.BKORDRNO ;BEGIN LOOP FOR ALL BACK *****
COMMENT ***** LET T.COUNTOCC = T.COUNTOCC + 1 *****
COMMENT ***** IF T.COUNTOCC >= T.FIRSTOCC *****
COMMENT ***** TRANSFER TO OUTPUT *****
COMMENT ***** END *****
COMMENT ***** ELSE *****
COMMENT ***** TRANSFER TO OUTPUT *****
COMMENT ***** END *****
COMMENT ***** *****

STMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD SEQ
NUMB NO. LEV CTR QLF FIELD NAME ATN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OPR NUMBER

( 1) (100) (R ) (D,0 ) (T,COUNTOCC)
( 2) (101) (T,SCREENNO) (*) (D,8 ) (T,LASTOCC )
( 3) (102) (T,LASTOCC ) (-) (D,7 ) (T,FIRSTOCC)
( 4) (103) (1,BKORDRNO) (EQ) (1,BKORDRNO )
( 5) (104) (NS) (OUTPUT )
( 6) (105) (T,COUNTOCC) (+) (D,1 ) (T,COUNTOCC)

```

Figure A-2 Program Listings for Sample Applications (Page 69 of 84)

```

( 7) (106)          (T,COUNT0CC)  (GE) (T,FIRSTOCC          )
( 8) (107)          (NS)          (END          )
( 9) (108)          (GO)          (OUTPUT          )
** MK5BX01 TYPE 0 DECODING REQUEST SCRNSEL FOR SCREEN ITMKOD4 .
** MK5DI02 TYPE 0 (S# 1) DEFINITION OF TEMPORARY FIELD COUNT0CC - FORMAT P LENGTH 8 DECIMALS 0 EDIT LENGTH 20 .

```

```

DEC 05, 2003 13.40.43          *****
                                * REQUEST NAME - IBKODVLD *
                                * INPUT STREAM REQUEST *
                                *****

```

STMT	CAT	REQ	SCREEN	BACK	BR	RNT	LINE
TYPE	ACT	TYP	NAME	CONTROL	TMP		NUMBER
(ER)							(S)
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****

STMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD	SEQ
NUMB	NO.	LEV	CTR	QLF FIELD NAME	ATN	QLF FIELD NAME OR CONSTANT	QLF FIELD NAME	STRT NUMB OPR	NUMBER
(1)	(100)			(F,ATINID)	(NE)	(C,ENMER			
(2)	(101)				(NS)	(END			
(3)	(102)			(F,ATINID)	(NE)	(C,PF008			
(4)	(103)				(NS)	(END			
(5)	(104)			(F,ATINID)	(NE)	(C,PF009			
(6)	(105)				(NS)	(END			
(7)	(106)			(F,ATINID)	(NE)	(C,PF010			
(8)	(107)				(NS)	(END			
(9)	(108)				(MS)	(C,INVALID KEY PRESSED - RETRY			
** MK5BX01	TYPE 0					DECODING REQUEST IBKODVLD FOR SCREEN ITMKOD4 .			

```

DEC 05, 2003 13.40.43          *****
                                * REQUEST NAME - IBKODITM *
                                * INPUT STREAM REQUEST *
                                *****

```

STMT	CAT	REQ	SCREEN	BACK	BR	RNT	LINE
TYPE	ACT	TYP	NAME	CONTROL	TMP		NUMBER
(ER)							(S)
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****
COMMENT	*****						*****

STMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD	SEQ
NUMB	NO.	LEV	CTR	QLF FIELD NAME	ATN	QLF FIELD NAME OR CONSTANT	QLF FIELD NAME	STRT NUMB OPR	NUMBER
(1)	(100)			(F,ATINID)	(EQ)	(C,PF008			
(2)	(101)				(NS)	(END			
(3)	(102)			(S,XFIELD)	(EQ)	(C,X			
(4)	(103)				(NS)	(END			
(5)	(104)				(R)	(S,ITEMNO			(S,ITEMORD)
** MK5BX01	TYPE 0					DECODING REQUEST IBKODITM FOR SCREEN ITMKOD4 .			

```

DEC 05, 2003 13.40.43          *****
                                * REQUEST NAME - SCRNNOCK *
                                * INPUT STREAM REQUEST *
                                *****

```

Figure A-2 Program Listings for Sample Applications (Page 70 of 84)

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(2)
COMMENT *****		; THIS PROCEDURE CHECKS THE SCREEN NUMBER REQUESTED	*****				
COMMENT *****		; IF THE TERMINAL OPERATOR HAS PRESSED ENTER, WHETHER	*****				
COMMENT *****		; OR NOT HE HAS REQUESTED A SPECIFIC SCREEN NUMBER.	*****				
COMMENT *****		;	*****				
COMMENT *****		IF F.ATTNID EQ 'ENTER'	*****				
COMMENT *****		IF S.REQSCRN EQ ' '	*****				
COMMENT *****		LET T.SCREENNO = T.SCREENNO + 1	*****				
COMMENT *****		ELSE	*****				
COMMENT *****		LET T.SCREENNO = S.REQSCRN + 0	*****				
COMMENT *****		END	*****				
COMMENT *****		IF S.SCREENNO > S.NOSCREEN	*****				
COMMENT *****		MSG 'REQUESTED SCREEN NOT AVAILABLE'	*****				
COMMENT *****		END	*****				
COMMENT *****		END	*****				
COMMENT *****		END	*****				

STMT NUMB	SEQ NO.	LOG LEV	CON CTR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER
(1)	(100)			(F,ATTNID)	(EQ)	(C,ENTER				
(2)	(101)				(NS)	(END				
(3)	(102)			(S,REQSCRN)	(EQ)	(C)				
(4)	(103)				(NS)	(106				
(5)	(104)			(T,SCREENNO)	(+)	(D,1				(T,SCREENNO)
(6)	(105)				(GO)	(107				
(7)	(106)			(S,REQSCRN)	(+)	(D,0				(T,SCREENNO)
(8)	(107)			(S,SCREENNO)	(GT)	(S,NOSCREEN				
(9)	(108)				(NS)	(END				
(10)	(109)				(MS)	(C,REQUESTED SCREEN NOT AVAILABLE)				
** MK5BX01 TYPE 0 DECODING REQUEST SCRNOCK FOR SCREEN ITMKOD4 .										

DEC 05, 2003 13.40.43

 * REQUEST NAME - IBOTYPE1 *
 * INPUT STREAM REQUEST *

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(1)
COMMENT *****		; TYPE 1 CONTROL PROCEDURE FOR ITMKOD4	*****				
COMMENT *****		;	*****				
COMMENT *****		CALL IBKODVLD	*****				
COMMENT *****		CALL IBKODITM	*****				
COMMENT *****			*****				

STMT NUMB	SEQ NO.	LOG LEV	CON CTR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER
(1)	(100)				(GO)	(SUB IBKODVLD				
(2)	(101)				(GO)	(SUB IBKODITM				
** MK5BX01 TYPE 0 DECODING REQUEST IBOTYPE1 FOR SCREEN ITMKOD4 .										

** MK5TJ05 TYPE 0 THE FOLLOWING CICS TS SYSTEM DEFINITION UTILITY INPUT STATEMENTS ARE REQUIRED FOR THIS APPLICATION.

DEFINE GROUP(???) , DUMP(NO) , TVASIZE(00000) , TRANSACTION(M503) , *

INDOUBT(BACKOUT) , PROGRAM(M503)

DEFINE GROUP(???) , LANGUAGE(ASSEMBLER) , PROGRAM(M503)

** MK5TJ03 TYPE 0 THE IMS PSB GENERATION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -

PCB TYPE=DB, POS=4, KEYLEN=0017, PROCOPT=AP, DEDNAME=CUSTOMDB

SENSE3 NAME=CUSTOMER

SENSE3 NAME=ORDER, PARENT=CUSTOMER

SENSE3 NAME=ITEMORD, PARENT=ORDER

PCB TYPE=DB, POS=4, KEYLEN=0014, PROCOPT=AP, DEDNAME=ITEMDB

SENSE3 NAME=ITEMROOT

SENSE3 NAME=BKORDERS, PARENT=ITEMROOT

PSBGEN LANG=ASSEM, MAXQ=009, PSBNAME=M14CSIT

END

** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - QANONHND COUNIX ERROR MENU BACKORD UPDATE .

** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - SCRNSLCT MSG SYNCFLD2 SYNCFLD1 .

Figure A-2 Program Listings for Sample Applications (Page 71 of 84)

```
** MK5W204 TYPE 0  NUMBER OF MESSAGES PRINTED IS 20.
                      M5INPUT  - 92 RECORDS INPUT
                      M5PUNCH  - 546 RECORDS OUTPUT
                      M5MAP    - 0 RECORDS OUTPUT
                      M5TPTEL  - 12 RECORDS OUTPUT
                      M5PREP   - 0 RECORDS OUTPUT
                      M5COMPAR  - 97 RECORDS OUTPUT
                      M5LIST   - 622 RECORDS OUTPUT
                      75 TRACKS ASSIGNED TO M5LIB - 23 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 800.
** MK5W209 TYPE 0  15144 BYTES OF TYPE A STORAGE USED DURING APPLICATION GENERATION
```

RUN NAME	STMT TYPE	LIB OPEN	SYS DELIM	DEBUG LEVEL	SRC LST	LKED PGMS	SYS ROLLB	CICS KEYS	MSG PROC	POS PCB	RUN TYPE	TRG MON	TRG SYS	PSB NAME	PROGRAM NAME	SEQUENCE 73 TO 80

(M504	,RC)	(#)					(C)			(G)	(C)			(M14CSIT)		

RUN NAME	STMT TYPE	FILE NAME	FILE TYPE	SYNCHRONIZING FIELD 1	FIELD 2	SYNC QUAL	SYNC TYPE	DL/I PATH	RECORD LENGTH	FLD	STOR TYPE	SQL AUTH-ID	SEQUENCE 73 TO 80

(M504	,RF)	(CUSTOMER)	(FILE)									(U)	
(M504	,RF)	(ITEMX)	(FILE1)									(U)	

```
***** BEGIN ASL PROCEDURE DELTRIG .
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE TRIGGERS THE DELETE OF AN ITEM FROM THE
; ITEM DATA BASE BY MOVING 'PF011' IN S.DELETE WHEN
; THE TERMINAL OPERATOR PASSES PF11.
; IT ALSO VALIDATES THE FUNCTION KEY PRESSED.
;
IF F.ATNID EQ 'ENTER', 'PF011', 'PF010'
ELSE
  MSG 'INVALID FUNCTION KEY PRESSED'
END
;
IF F.ATNID EQ 'PF011' AND S.DELETE EQ ' '
  LET S.DELETE = 'PF011'
  SET S.ITEMNO
  LET T.MSG = 'PRESS PF11 AGAIN TO DELETE ITEM'
  LET PF(T.MSG,32,31) = ';PRESS PF10 TO CANCEL DELETION'
  MSG T.MSG
END
** MK5CP01 TYPE 0  PREPROCESSING INPUT STREAM REQUEST DELTRIG .
```

```
***** BEGIN ASL PROCEDURE TEMPS .
***** Using ASL Translator Version 4.50 *****
; TEMP FIELD DEFINITIONS
SYNCFLD1: FIELD C 7
SYNCFLD2: FIELD C 7
SCREENNO: FIELD Z 2
NOSCREEN: FIELD Z 2
MSG: FIELD C 74
SCRNSLCT: FIELD C 1
UPDATE: FIELD C 1 INIT 'U'
BACKORD: FIELD C 1 INIT 'B'
MENU: FIELD C 1 INIT 'M'
ERROR: FIELD C 1 INIT 'E'
FIRSTOCC: FIELD P 2
LASTOCC: FIELD P 2
COUNTX: FIELD P 2
QANONHND: FIELD C 7
** MK5CP01 TYPE 0  PREPROCESSING INPUT STREAM REQUEST TEMPS .
```

```
DEC 05, 2003 13.46.39
*****
* REQUEST NAME - TEMPS
* INPUT STREAM REQUEST
*****
```

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK CONTROL	BR TMP	RNT TMP	LINE NUMBER

(ER)	COMMENT *****	; TEMP FIELD DEFINITIONS						(N)

Figure A-2 Program Listings for Sample Applications (Page 72 of 84)

```

COMMENT ***** SYNCFLD1: FIELD C 7 *****
COMMENT ***** SYNCFLD2: FIELD C 7 *****
COMMENT ***** SCREENNO: FIELD Z 2 *****
COMMENT ***** NOSCREEN: FIELD Z 2 *****
COMMENT ***** MSG: FIELD C 74 *****
COMMENT ***** SCRNSLCT: FIELD C 1 *****
COMMENT ***** UPDATE: FIELD C 1 INIT 'U' *****
COMMENT ***** BACKORD: FIELD C 1 INIT 'B' *****
COMMENT ***** MENU: FIELD C 1 INIT 'M' *****
COMMENT ***** ERROR: FIELD C 1 INIT 'E' *****
COMMENT ***** FIRSTOCC: FIELD P 2 *****
COMMENT ***** LASTOCC: FIELD P 2 *****
COMMENT ***** COUNTY: FIELD P 2 *****
COMMENT ***** QANONHAND: FIELD C 7 *****
COMMENT *****

```

STMT TYPE	FIELD NAME	FIELD LENGTH	FLD TYP	DEC PLC	OUTPT EDIT	EDIT LGTH	INITIAL VALUE	COMMENTS	SEQUENCE NUMBER
(TF)	(SYNCFLD1)	(7)	(C)						
(TF)	(SYNCFLD2)	(7)	(C)						
(TF)	(SCREENNO)	(2)	(Z)						
(TF)	(NOSCREEN)	(2)	(Z)						
(TF)	(MSG)	(74)	(C)						
(TF)	(SCRNSLCT)	(1)	(C)						
(TF)	(UPDATE)	(1)	(C)			(U)			
(TF)	(BACKORD)	(1)	(C)			(B)			
(TF)	(MENU)	(1)	(C)			(M)			
(TF)	(ERROR)	(1)	(C)			(E)			
(TF)	(FIRSTOCC)	(2)	(P)						
(TF)	(LASTOCC)	(2)	(P)						
(TF)	(COUNTX)	(2)	(P)						
(TF)	(QANONHAND)	(7)	(C)						

```

DEC 05, 2003 13.46.39
*****
* SCREEN CONTROL *
*****

```

INPUT SCREEN	STMT TYPE	SEQ NO.	LOG INVL	CON CTR	.OPERAND A.. QLF FIELD	OPER ATTN	OPERAND B..... QLF FIELD NAME/CONSTANT	OUTPUT SCREEN	.OPERAND C.. QLF FLD/CNST	PARTL STR	FLD NM	OP	SEQUENCE 73 TO 80
(ITMUPDT4)	(SC)					(XR)	(1,DELIRIG)						
(ITMUPDT4)	(SC)				(F,ATTNID)	(EQ)	(C,ENTER)						
(ITMUPDT4)	(SC)					(OS)						(ITMUPDT4)	
(ITMUPDT4)	(SC)			(E)	(F,ATTNID)	(EQ)	(C,PF010)						
(ITMUPDT4)	(SC)	(1)	(O)		(F,ATTNID)	(EQ)	(C,PF011)						
(ITMUPDT4)	(SC)					(OS)						(ORDMENU4)	

SCREEN NAME	STMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80
(ITMUPDT4)	(SD)	(R)		()	(@)			(M504)			(ITMUPDT)	

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80
(ITMUPDT4,SL)	(1)	(0)			(ITMUPDT4		
(ITMUPDT4,SL)	(1)	(1)			(*****		
(ITMUPDT4,SL)	(2)	(0)			(		
(ITMUPDT4,SL)	(2)	(1)			(* MODIFY RECORD=ENTER *		
(ITMUPDT4,SL)	(3)	(0)			(ITEM NUMBER		
(ITMUPDT4,SL)	(3)	(1)			(* RECORD DELETE=PF11 *		
(ITMUPDT4,SL)	(4)	(0)			(PRICE		
(ITMUPDT4,SL)	(4)	(1)			(* RETURN TO MENU=PF10 *		
(ITMUPDT4,SL)	(5)	(0)			(QUANTITY/BOX		
(ITMUPDT4,SL)	(5)	(1)			(*****		
(ITMUPDT4,SL)	(6)	(0)			(QUANTITY ON HAND		
(ITMUPDT4,SL)	(7)	(0)			(SHIPPED YTD		
(ITMUPDT4,SL)	(8)	(0)			(ORDERED YTD		
(ITMUPDT4,SL)	(9)	(0)			(QUANTITY BACK-ORDERED		
(ITMUPDT4,SL)	(10)	(0)			(RE-ORDER POINT		
(ITMUPDT4,SL)	(11)	(0)			(PRODUCT CODE		

Figure A-2 Program Listings for Sample Applications (Page 73 of 84)

** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES												SEQUENCE 73 TO 80	
(ITMUPDT4,SS) (1) (1)																
SCREEN NAME	SIMT TYPE	SEG NO.	V A	S QLF	DEST FLD NAME	I O	R D	C E	P M	N TDM	A T	L V	C V	PARTIAL FIELD ST NO OP		SEQUENCE 73 TO 80
													OPERAND B.....			
													QLF FIELD NAME OR CONSTANT			
(ITMUPDT4,S0)	(1)			(T,DELETE)	(B)	(D)		(Y,Y)			(P)	(NM)	(DELETE)			
(ITMUPDT4,S0)	(1)			(1,ITEMNO)	(B)			(Y)			(U)	(NM)	(ITEMNO)			
(ITMUPDT4,S1)											(D)	(IF)	(F,ATTNID)	(EQ)	(C,PF011)	
(ITMUPDT4,S1)											(X)	(IF)	(F,ATTNID)	(EQ)	(C,PF010)	
(ITMUPDT4,S0)	(1)			(1,ITMPRICE)	(B)						(P)	(NM)	(ITMPRICE)			
(ITMUPDT4,S0)	(1)			(1,PACKBOX)	(B)						(P)	(NM)	(PACKBOX)			
(ITMUPDT4,S0)	(1)			(1,QANONHND)	(B)						(P)	(NM)	(QANONHND)			
(ITMUPDT4,S0)	(1)			(1,SHIPYTD)	(B)						(P)	(NM)	(SHIPYTD)			
(ITMUPDT4,S0)	(1)			(1,ORDYTD)	(B)						(P)	(NM)	(ORDYTD)			
(ITMUPDT4,S0)	(1)			(1,QTYEKORD)	(O)							(NM)	(QTYEKORD)			
(ITMUPDT4,S0)	(1)			(1,REORDPT)	(B)						(P)	(NM)	(REORDPT)			
(ITMUPDT4,S0)	(1)			(1,PRODCODE)	(B)						(P)	(NM)	(PRODCODE)			
(ITMUPDT4,S0)	(1)	(A)		(F,MSGLINE)				(H)								(*SYSTEM*)
**MSX901 TYPE 0 TEMPORARY FIELD DELETE DEFINED - FORMAT C, LENGTH 1, DECIMALS 0, EDIT LENGTH 1.																

** M5X901 TYPE 0 TEMPORARY FIELD DELETE DEFINED - FORMAT C, LENGTH 1, DECIMALS 0, EDIT LENGTH 1.

SCREEN NAME	SIMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80
(ORDMENU4) (SD) (R) () (0) (M510) (M510)												
SCREEN NAME	SIMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5..... <-TEXT (1-57)							SEQUENCE 73 TO 80
					..6....7....8....9....0....1.... <-TEXT (58-114)							

(ORDMENU4,SL)	(1)	(0)		(ORDMENU4								
(ORDMENU4,SL)	(3)	(0)			CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS:							
(ORDMENU4,SL)	(5)	(0)			PF1---CUSTOMER ADD/UPDATE							
(ORDMENU4,SL)	(7)	(0)			PF2---ITEM BROWSE BY PRODUCT KEY							
(ORDMENU4,SL)	(9)	(0)			PF3---BACK-ORDERED ITEM							
(ORDMENU4,SL)	(11)	(0)			PF4---ITEM ADD/UPDATE							
(ORDMENU4,SL)	(13)	(0)			PF5---END TRANSACTION							
(ORDMENU4,SL)	(16)	(0)			ENTER CUSTOMER OR ITEM NUMBER							
(ORDMENU4,SL)	(18)	(0)			(ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2)							
(ORDMENU4,SL)	(18)	(1)			(IS USED)							

** M5WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES												SEQUENCE 73 TO 80	
(ORDMENU4,SS) (1) (1)																
SCREEN NAME	SIMT TYPE	SEG NO.	V A	S QLF	DEST FLD NAME	I O	R D	C I	E R	P R	M TDM	A T	L V	C V	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80
(ORDMENU4,S0)	(1)			(N,CUSTNO)	(I)							(X)	(NM)	(ITEMNO)		
(ORDMENU4,S1)												(V)	(IF)	(S,PFKEY)	(EQ) (C,PF001)	
(ORDMENU4,S0)	(1)	(S)		(1,ITEMNO)	(I)							(X)				
(ORDMENU4,S1)												(M)	(IF)	(S,PFKEY)	(EQ) (C,PF003)	
(ORDMENU4,S1)												(V)	(IF)	(S,PFKEY)	(EQ) (C,PF004)	
(ORDMENU4,S0)	(1)	(V)			(I)							(N)	(NM)	(PFKEY)		
(ORDMENU4,S1)													(LN)	(5)		
(ORDMENU4,S1)													(EQ)	(C,PF001)		
(ORDMENU4,S1)													(1,O,EQ)	(C,PF002)		
(ORDMENU4,S1)													(1,O,EQ)	(C,PF003)		
(ORDMENU4,S1)													(1,O,EQ)	(C,PF004)		
(ORDMENU4,S1)													(1,O,EQ)	(C,PF005)		
(ORDMENU4,S0)	(1)	(A)		(F,MSGLINE)				(H)								(*SYSTEM*)

Figure A-2 Program Listings for Sample Applications (Page 74 of 84)

DEC 05, 2003 13.46.39

* REQUEST NAME - DELTRIG *
* INPUT STREAM REQUEST *

STMT	CAT		REQ	SCREEN	BACK	BR	RNT	LINE
TYPE	ACT	REQUEST DESCRIPTION	TYP	NAME	CONTROL	TMP		NUMBER

```
(ER) (1)
COMMENT ***** ; THIS PROCEDURE TRIGGERS THE DELETE OF AN ITEM FROM THE *****
COMMENT ***** ; ITEM DATA BASE BY MOVING 'PF011' IN S.DELETE WHEN *****
COMMENT ***** ; THE TERMINAL OPERATOR PRESSES PF11. *****
COMMENT ***** ; IT ALSO VALIDATES THE FUNCTION KEY PRESSED. *****
COMMENT ***** ; *****
COMMENT ***** IF F.ATTNID EQ 'ENTER', 'PF011', 'PF010' *****
COMMENT ***** ELSE *****
COMMENT ***** MSG 'INVALID FUNCTION KEY PRESSED' *****
COMMENT ***** END *****
COMMENT ***** ; *****
COMMENT ***** IF F.ATTNID EQ 'PF011' AND S.DELETE EQ ' ' *****
COMMENT ***** LET S.DELETE = 'PF011' *****
COMMENT ***** SET S.ITEMNO *****
COMMENT ***** LET T.MSG = 'PRESS PF11 AGAIN TO DELETE ITEM' *****
COMMENT ***** LET PF(T.MSG,32,31) = ';PRESS PF10 TO CANCEL DELETION' *****
COMMENT ***** MSG T.MSG *****
COMMENT ***** END *****
COMMENT ***** *****
```

STMT	SEQ	LOG	CON	..OPERAND A...	OPER	OPERAND B.....	RESULT....	PARTIAL FIELD	SEQ
NUMB	NO.	LEV	CIR	QLF FIELD NAME	ATN	QLF FIELD NAME OR CONSTANT	QLF FIELD NAME	STRT NUMB OPR	NUMBER

(	1	(100)		(F,ATTNID)	(EQ)	(C,ENTER	)		
(	2	(101)			(GS)	(108	)		
(	3	(102)		(F,ATTNID)	(EQ)	(C,PF011	)		
(	4	(103)			(GS)	(108	)		
(	5	(104)		(F,ATTNID)	(EQ)	(C,PF010	)		
(	6	(105)			(NS)	(107	)		
(	7	(106)			(GO)	(108	)		
(	8	(107)			(MS)	(C,INVALID FUNCTION KEY PRESSED	)		
(	9	(108)		(F,ATTNID)	(EQ)	(C,PF011	)		
(	10	(109)			(NS)	(END	)		
(	11	(110)		(S,DELETE)	(EQ)	(C)			
(	12	(111)			(NS)	(END	)		
(	13	(112)			(R)	(C,PF011	)	(S,DELETE)	
(	14	(113)		(S,ITEMNO)	(SE)				
(	15	(114)			(R)	(C,PRESS PF11 AGAIN TO DELETE ITEM)	(T,MSG	)	
(	16	(115)			(R)	(C,;PRESS PF10 TO CANCEL DELETION)	(T,MSG	)	(32) (31) (C)
(	17	(116)			(MS)	(T,MSG	)		

** MK5X01 TYPE 0 DECODING REQUEST DELTRIG FOR SCREEN ITMUPDT4 .

** MK5TJ05 TYPE 0 THE FOLLOWING CICS TS SYSTEM DEFINITION UTILITY INPUT STATEMENTS ARE REQUIRED FOR THIS APPLICATION.
DEFINE GROUP(???),DUMP(NO),TWASIZE(0000),TRANSACTION(M504),
INDOUBT(BACKOUT),PROGRAM(M504)

** MK5TJ03 TYPE 0 THE IMS PSB GENERATION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -
PCB TYPE=DB,POS=4,KEYLEN=0005,PROCOPT=AP,DEBNAME=CUSTOMDB
SENSE NAME=CUSTOMER
PCB TYPE=DB,POS=4,KEYLEN=0007,PROCOPT=AP,DEBNAME=ITEMDB
SENSE NAME=ITEMROOT
PSBGEN LANG=ASSEM,MAXQ=001,PSNAME=M14CSIT
END

** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - QANONHD COUNIX LASTOCC FIRSTOCC ERROR MENU .
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - BACKORD UPDATE SCRNSLCT NOSCREEN SCREENNO SYNCFLD2 .
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - SYNCFLD1 .

** MK5N204 TYPE 0 NUMBER OF MESSAGES PRINTED IS 11.

M5INPUT	-	48 RECORDS INPUT
M5PUNCH	-	383 RECORDS OUTPUT
M5MAP	-	0 RECORDS OUTPUT
M5TPTEL	-	9 RECORDS OUTPUT
M5PREP	-	0 RECORDS OUTPUT
M5COMMAR	-	37 RECORDS OUTPUT
M5LIST	-	315 RECORDS OUTPUT

75 TRACKS ASSIGNED TO M5LIB - 23 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 800.
** MK5N209 TYPE 0 8896 BYTES OF TYPE A STORAGE USED DURING APPLICATION GENERATION

Figure A-2 Program Listings for Sample Applications (Page 75 of 84)

RUN NAME	STMT TYPE	LIB OPEN	SYS DELIM	DEBUG LEVEL	SRC LST	LINKED PGMS	SYS ROLLB	MSG KEYS	CICS PROC	POS PCB	RUN TYPE	TRG MON	TRG SYS	PSB NAME	PROGRAM NAME	SEQUENCE 73 TO 80
(M508	,RC)			(#)							(G)	(C)				
RUN NAME	STMT TYPE	FILE NAME	FILE TYPE	SYNCHRONIZING FIELD 1 FIELD 2		SYNC QUAL	SYNC TYPE	DL/I PATH	RECORD LENGTH	FLD FLD	STOR TYPE	SQL AUTH-ID	SEQUENCE 73 TO 80			
(M508	,RF)	(CUSTOMER)	(FILE)										(U)			
(M508	,RF)	(ITEMX)	(FILE1)										(U)			
(M508	,RF)	(ITEMX)	(FILE2)	(SYNCFLD1)	(SYNCFLD2)		(T)									
***** BEGIN ASL PROCEDURE FEKODVAL. ***** Using ASL Translator Version 4.50 ***** ; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED ; AND THE ENTRY CHOSEN BY THE TERMINAL OPERATOR ; IF F.ATTNID EQ 'PF010', 'ENTER' AND S.CHOICE EQ 'Y', 'N' ELSE MSG 'INVALID KEY OR CHOICE - RETRY' END ** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST FEKODVAL.																
***** BEGIN ASL PROCEDURE FILLINV . ***** Using ASL Translator Version 4.50 ***** ; THIS PROCEDURE SUPPLIES THE INVOICE DATE AND AMOUNT ; IF F.ATTNID EQ 'PF010' CONTINUE ELSE LET T.SYNCFLD1 = S.ITEMNO LET T.SYNCFLD2 = S.ITEMNO IF S.CHOICE EQ 'Y' LET INVDAT = F.TODAY LET INAMT = S.ITMPRICE * S.QTYBKORD END END ** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST FILLINV .																
***** BEGIN ASL PROCEDURE TEMPS . ***** Using ASL Translator Version 4.50 ***** ; TEMP FIELD DEFINITIONS SYNCFLD1: FIELD C 7 SYNCFLD2: FIELD C 7 SCREENNO: FIELD Z 2 NOSCREEN: FIELD Z 2 MSG: FIELD C 74 SCRNSLCT: FIELD C 1 UPDATE: FIELD C 1 INIT 'U' BACKORD: FIELD C 1 INIT 'B' MENU: FIELD C 1 INIT 'M' ERROR: FIELD C 1 INIT 'E' FIRSTOCC: FIELD P 2 LASTOCC: FIELD P 2 COUNTX: FIELD P 2 QANONHND: FIELD C 7 ** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST TEMPS .																
***** * REQUEST NAME - TEMPS * * INPUT STREAM REQUEST * *****																
DEC 05, 2003 13.50.06																
STMT TYPE	CAT ACT	REQUEST DESCRIPTION										REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)																
COMMENT *****		; TEMP FIELD DEFINITIONS										*****	(N)			
COMMENT *****		SYNCFLD1: FIELD C 7										*****				
COMMENT *****		SYNCFLD2: FIELD C 7										*****				

Figure A-2 Program Listings for Sample Applications (Page 76 of 84)

```

COMMENT ***** SCREENNO: FIELD Z 2 *****
COMMENT ***** NOSCREEN: FIELD Z 2 *****
COMMENT ***** MSG: FIELD C 74 *****
COMMENT ***** SCRNSLCT: FIELD C 1 *****
COMMENT ***** UPDATE: FIELD C 1 INIT 'U' *****
COMMENT ***** BACKORD: FIELD C 1 INIT 'B' *****
COMMENT ***** MENU: FIELD C 1 INIT 'M' *****
COMMENT ***** ERROR: FIELD C 1 INIT 'E' *****
COMMENT ***** FIRSTOCC: FIELD P 2 *****
COMMENT ***** LASTOCC: FIELD P 2 *****
COMMENT ***** COUNIX: FIELD P 2 *****
COMMENT ***** QANONHND: FIELD C 7 *****
COMMENT *****

```

STMT TYPE	FIELD NAME	FIELD INITH	FLD TYP	DEC PLC	OUTPT EDIT	EDIT LGTH	INITIAL VALUE	COMMENTS	SEQUENCE NUMBER
(TF)	(SYNCFD1)	(7)	(C)						
(TF)	(SYNCFD2)	(7)	(C)						
(TF)	(SCREENNO)	(2)	(Z)						
(TF)	(NOSCREEN)	(2)	(Z)						
(TF)	(MSG)	(74)	(C)						
(TF)	(SCRNSLCT)	(1)	(C)						
(TF)	(UPDATE)	(1)	(C)			(U)			
(TF)	(BACKORD)	(1)	(C)			(B)			
(TF)	(MENU)	(1)	(C)			(M)			
(TF)	(ERROR)	(1)	(C)			(E)			
(TF)	(FIRSTOCC)	(2)	(P)						
(TF)	(LASTOCC)	(2)	(P)						
(TF)	(COUNIX)	(2)	(P)						
(TF)	(QANONHND)	(7)	(C)						

```

*****
*          SCREEN CONTROL          *
*****
DEC 05, 2003 13.50.06

```

INPUT SCREEN	STMT TYPE	SEQ NO.	LOG LVL	CON CTR	OPERAND A. QLF FIELD	OPER ATN	OPERAND B. QLF FIELD NAME/CONSTANT	OUTPUT SCREEN	OPERAND C. QLF FLD/CONST	PARTIAL STR NM	FLD NM	SEQUENCE 73 TO 80
(FILEKOD4)	(SC)					(XR)	(1,FBKODVAL)					
(FILEKOD4)	(SC)					(XR)	(M,FILLINV)					
(FILEKOD4)	(SC)				(F,ATTNID)	(EQ)	(C,ENTER)					
(FILEKOD4)	(SC)					(OS)					(ITMRWS4)	
(FILEKOD4)	(SC)				(E)(F,ATTNID)	(EQ)	(C,PF010)					
(FILEKOD4)	(SC)					(OS)					(ORDMENU4)	

SCREEN NAME	STMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80
(FILEKOD4)	(SD)	(R)		()	(@)							

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80
(FILEKOD4,SL)	(1)	(0)			(FILEKOD4	DO YOU WANT TO FILL	
(FILEKOD4,SL)	(1)	(1)			(FILL BACK-ORDER='Y' OR		
(FILEKOD4,SL)	(2)	(0)			(	THIS BACK-ORDER (Y OR N)? _	
(FILEKOD4,SL)	(2)	(1)			(	'N' + ENTER	
(FILEKOD4,SL)	(3)	(1)			(RETURN TO MENU=PF10		
(FILEKOD4,SL)	(4)	(0)			(	ITEM NUMBER _____	
(FILEKOD4,SL)	(5)	(0)			(	BACK-ORDER NUMBER _____	
(FILEKOD4,SL)	(6)	(0)			(	ITEM NAME _____	
(FILEKOD4,SL)	(7)	(0)			(	PRICE _____	
(FILEKOD4,SL)	(8)	(0)			(	QUANTITY BACK-ORDERED _____	
(FILEKOD4,SL)	(9)	(0)			(	QUANTITY ON HAND _____	
(FILEKOD4,SL)	(10)	(0)			(	CUSTOMER NUMBER _____	
(FILEKOD4,SL)	(11)	(0)			(	CUSTOMER NAME _____	
(FILEKOD4,SL)	(11)	(1)			(		
(FILEKOD4,SL)	(12)	(0)			(	ORDER NUMBER _____	
(FILEKOD4,SL)	(13)	(0)			(	SHIPMENT NUMBER _____	
(FILEKOD4,SL)	(14)	(0)			(	SHIPPER _____	
(FILEKOD4,SL)	(15)	(0)			(	INVOICE NUMBER _____	

Figure A-2 Program Listings for Sample Applications (Page 77 of 84)


```

(ITMWRWS4,SL) ( 20) (1) ( _____ )
(ITMWRWS4,SL) ( 21) (0) ( _____ )
(ITMWRWS4,SL) ( 21) (1) ( _____ )
(ITMWRWS4,SL) ( 22) (0) ( _____ )
(ITMWRWS4,SL) ( 22) (1) ( _____ )
(ITMWRWS4,SL) ( 23) (0) ( _____ )
(ITMWRWS4,SL) ( 23) (1) ( _____ )
** M$WLL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES 73 TO 80

(ITMWRWS4,SS) ( 1) ( 1)
-----
V I R C E P M N A L C
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... PARTIAL
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ITMWRWS4,S0) ( 1) (T,SYNCFD1) (B) (P) (NM) (SYNCFD1) )
(ITMWRWS4,S0) ( 1) (T,SYNCFD2) (B) (P) (NM) (SYNCFD2) )
(ITMWRWS4,S0) ( 1) (A) (F,MSGLINE) (H) (*SYSTEM*)
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES 73 TO 80

(ITMWRWS4,SS) ( 2) ( 15)
-----
V I R C E P M N A L C
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... PARTIAL
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ITMWRWS4,S0) ( 2) (I) (Y) (N) (NM) (ACTION) )
(ITMWRWS4,S0) ( 2) (1,ITEMNO) (I) (X) (NM) (ITEMNO) )
(ITMWRWS4,S1) (M) (IF) (S,ACTION) (EQ) (C,U) )
(ITMWRWS4,S1) (M) (IF) (S,ACTION) (EQ) (C,M) )
(ITMWRWS4,S0) ( 2) (S) (2,ITEMNO) (O)
(ITMWRWS4,S0) ( 2) (2,ITEMPRICE) (O) (NM) (ITEMPRICE) )
(ITMWRWS4,S0) ( 2) (2,QANONHND) (O) (NM) (QANONHND) )
(ITMWRWS4,S0) ( 2) (2,SHIPYTD) (O) (NM) (SHIPYTD) )
(ITMWRWS4,S0) ( 2) (2,ORDYTD) (O) (NM) (ORDYTD) )
(ITMWRWS4,S0) ( 2) (2,QTYEKORD) (O) (NM) (QTYEKORD) )
-----
SCREEN STMT CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN PROC TERMINAL SEQUENCE
NAME TYPE ACTN TYPE DESIG DESIG FORMAT CONTROL CODE ID NULL DEVICE 73 TO 80

(ORDMENU4) (SD) (R) ( ) (0) (M510) (M510) )
-----
SCREEN STMT LINE CONT- EXTN .....1.....2.....3.....4.....5..... <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6.....7.....8.....9.....0.....1..... <-TEXT (58-114) 73 TO 80

(ORDMENU4,SL) ( 1) (0) ( ORDMENU4 )
(ORDMENU4,SL) ( 3) (0) ( CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS:)
(ORDMENU4,SL) ( 5) (0) ( PF1---CUSTOMER ADD/UPDATE )
(ORDMENU4,SL) ( 7) (0) ( PF2---ITEM BROWSE BY PRODUCT KEY )
(ORDMENU4,SL) ( 9) (0) ( PF3---BACK-ORDERED ITEM )
(ORDMENU4,SL) (11) (0) ( PF4---ITEM ADD/UPDATE )
(ORDMENU4,SL) (13) (0) ( PF5---END TRANSACTION )
(ORDMENU4,SL) (16) (0) ( ENTER CUSTOMER OR ITEM NUMBER )
(ORDMENU4,SL) (18) (0) ( (ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2) )
(ORDMENU4,SL) (18) (1) ( IS USED )
** M$WLL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES 73 TO 80

(ORDMENU4,SS) ( 1) ( 1)

```

Figure A-2 Program Listings for Sample Applications (Page 79 of 84)

```

-----
V      I R C E P M N      A L C      PARTIAL
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ORDMENU4,S0) ( 1) (N,CUSTNO ) (I) (X) (NM) (ITEMNO )
(ORDMENU4,S1) (V) (IF) (S,PFKEY ) (EQ) (C,PF001 )
(ORDMENU4,S0) ( 1) (S) (1,ITEMNO ) (I) (X)
(ORDMENU4,S1) (M) (IF) (S,PFKEY ) (EQ) (C,PF003 )
(ORDMENU4,S1) (V) (IF) (S,PFKEY ) (EQ) (C,PF004 )
(ORDMENU4,S0) ( 1) (V) (I) (N) (NM) (PFKEY )
(ORDMENU4,S1) (LN) (5)
(ORDMENU4,S1) (EQ) (C,PF001 )
(ORDMENU4,S1) (1,O,EQ) (C,PF002 )
(ORDMENU4,S1) (1,O,EQ) (C,PF003 )
(ORDMENU4,S1) (1,O,EQ) (C,PF004 )
(ORDMENU4,S1) (1,O,EQ) (C,PF005 )
(ORDMENU4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)

DEC 05, 2003 13.50.06 *****
* REQUEST NAME - FBKODVAL *
* INPUT STREAM REQUEST *
*****

-----
STMT CAT REQ SCREEN BACK BR RNT LINE
TYPE ACT REQUEST DESCRIPTION TYP NAME CONTROL TMP NUMBER

(ER) (1)
COMMENT ***** ; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED *****
COMMENT ***** ; AND THE ENTRY CHOSEN BY THE TERMINAL OPERATOR *****
COMMENT ***** ; *****
COMMENT ***** IF F.ATINID EQ 'PF010', 'ENTER' AND S.CHOICE EQ 'Y', 'N' *****
COMMENT ***** ELSE *****
COMMENT ***** MSG 'INVALID KEY OR CHOICE - RETRY' *****
COMMENT ***** END *****
COMMENT ***** *****

-----
STMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD SEQ
NUMB NO. LEV CTR QLF FIELD NAME ATN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OPR NUMBER

( 1) (100) (F,ATINID ) (EQ) (C,PF010 )
( 2) (101) ( ) (GS) (104 )
( 3) (102) (F,ATINID ) (EQ) (C,ENTER )
( 4) (103) ( ) (NS) (109 )
( 5) (104) (S,CHOICE ) (EQ) (C,Y )
( 6) (105) ( ) (GS) (END )
( 7) (106) (S,CHOICE ) (EQ) (C,N )
( 8) (107) ( ) (NS) (109 )
( 9) (108) ( ) (GO) (END )
( 10) (109) ( ) (MS) (C,INVALID KEY OR CHOICE - RETRY )
** MK5BX01 TYPE 0 DECODING REQUEST FBKODVAL FOR SCREEN FILEKOD4 .

DEC 05, 2003 13.50.06 *****
* REQUEST NAME - FILLINV *
* INPUT STREAM REQUEST *
*****

-----
STMT CAT REQ SCREEN BACK BR RNT LINE
TYPE ACT REQUEST DESCRIPTION TYP NAME CONTROL TMP NUMBER

(ER) (M)
COMMENT ***** ; THIS PROCEDURE SUPPLIES THE INVOICE DATE AND AMOUNT *****
COMMENT ***** ; *****
COMMENT ***** IF F.ATINID EQ 'PF010' *****
COMMENT ***** CONTINUE *****
COMMENT ***** ELSE *****
COMMENT ***** LET T.SYNCFD1 = S.ITEMNO *****
COMMENT ***** LET T.SYNCFD2 = S.ITEMNO *****
COMMENT ***** IF S.CHOICE EQ 'Y' *****
COMMENT ***** LET INVDAT = F.TODAY *****
COMMENT ***** LET INVAMT = S.ITMPRICE * S.QTYBKORD *****
COMMENT ***** END *****
COMMENT ***** *****

```

Figure A-2 Program Listings for Sample Applications (Page 80 of 84)

```

COMMENT *****
*****

STMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD SEQ
NUMB NO. LEV CTR QLF FIELD NAME ATN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OPR NUMBER

( 1) (100) (F,ATINID ) (BQ) (C,PF010 )
( 2) (101) (NS) (103 )
( 3) (102) (GO) (END )
( 4) (103) (R ) (S,ITEMNO ) (T,SYNCFD1)
( 5) (104) (R ) (S,ITEMNO ) (T,SYNCFD2)
( 6) (105) (S,CHOICE ) (BQ) (C,Y )
( 7) (106) (NS) (END )
( 8) (107) (R ) (F,TODAY ) (INVDAT )
( 9) (108) (S,ITMERICE) (*) (S,QTYBKORD ) (INAMT )
** MK5BX01 TYPE 0 DECODING REQUEST FILLINV FOR SCREEN FILEKOD4 .

** MK5TJ05 TYPE 0 THE FOLLOWING CICS TS SYSTEM DEFINITION UTILITY INPUT STATEMENTS ARE REQUIRED FOR THIS APPLICATION.
      DEFINE GROUP(???) ,DUMP(NO),TWASIZE(00000),TRANSACTION(M508) , *
      INDOUBT(BACKOUT),PROGRAM(M508)
      DEFINE GROUP(???) ,LANGUAGE(ASSEMBLER),PROGRAM(M508)
** MK5TJ03 TYPE 0 THE IMS PSB GENERATION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -
      PCB TYPE=DB,POS=4,KEYLEN=0014,PROCOPT=AP,DBDNAME=CUSTOMDB
      SENSEG NAME=CUSTOMER
      SENSEG NAME=ORDER,PARENT=CUSTOMER
      SENSEG NAME=SHIPIN,PARENT=ORDER
      PCB TYPE=DB,POS=4,KEYLEN=0014,PROCOPT=AP,DBDNAME=ITEMDB
      SENSEG NAME=ITEMROOT
      SENSEG NAME=BKORDERS,PARENT=ITEMROOT
      PCB TYPE=DB,POS=4,KEYLEN=0007,PROCOPT=GP,DBDNAME=ITEMDB
      SENSEG NAME=ITEMROOT
      PSBGEN LANG=ASSEM,MAXQ=002,PSBNAME=M508
      END
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - QANONHND COUNIX LASTOCC FIRSTOCC ERROR MENU .
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - BACKORD UPDATE SCRNSLCT MSG NOSCREEN SCREENNO.

** MK5W204 TYPE 0 NUMBER OF MESSAGES PRINTED IS 12.
      M5INPUT - 54 RECORDS INPUT
      M5PUNCH - 480 RECORDS OUTPUT
      M5MAP - 0 RECORDS OUTPUT
      M5TPTEL - 14 RECORDS OUTPUT
      M5PREP - 0 RECORDS OUTPUT
      M5COMMAR - 69 RECORDS OUTPUT
      M5LIST - 436 RECORDS OUTPUT
      75 TRACKS ASSIGNED TO M5LIB - 23 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 800.
** MK5W209 TYPE 0 16392 BYTES OF TYPE A STORAGE USED DURING APPLICATION GENERATION

-----
RUN STMT LIB SYS DEBUG SRC LKED SYS CICS MSG POS RUN TRG TRG PSB PROGRAM SEQUENCE
NAME TYPE OPEN DELIM LEVEL LST PGMS ROLLB KEYS PROC PCB TYPE MON SYS NAME NAME 73 TO 80

(M509 ,RC) (#) (G) (C)

-----
RUN STMT FILE FILE SYNCHRONIZING SYNC SYNC DL/I RECORD STOR SQL SEQUENCE
NAME TYPE NAME TYPE FIELD 1 FIELD 2 QUAL TYPE PATH LENGTH FLD TYPE AUTH-ID 73 TO 80

(M509 ,RF) (CUSTOMER) (FILE ) (U)
(M509 ,RF) (ITEMX ) (FILE1 ) (U)

***** BEGIN ASL PROCEDURE TEMPS .
***** Using ASL Translator Version 4.50 *****
; TEMP FIELD DEFINITIONS
SYNCFD1: FIELD C 7
SYNCFD2: FIELD C 7
SCREENNO: FIELD Z 2
NOSCREEN: FIELD Z 2
MSG: FIELD C 74
SCRNSLCT: FIELD C 1
UPDATE: FIELD C 1 INIT 'U'
BACKORD: FIELD C 1 INIT 'B'
MENU: FIELD C 1 INIT 'M'
ERROR: FIELD C 1 INIT 'E'
FIRSTOCC: FIELD P 2
LASTOCC: FIELD P 2
COUNIX: FIELD P 2

```

Figure A-2 Program Listings for Sample Applications (Page 81 of 84)

QANONHND: FIELD C 7
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST TEMPS .

DEC 05, 2003 13.54.15

* REQUEST NAME - TEMPS *
* INPUT STREAM REQUEST *

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(N)
COMMENT	*****	; TEMP FIELD DEFINITIONS					*****
COMMENT	*****	SYNCFD1: FIELD C 7					*****
COMMENT	*****	SYNCFD2: FIELD C 7					*****
COMMENT	*****	SCREENNO: FIELD Z 2					*****
COMMENT	*****	NOSCREEN: FIELD Z 2					*****
COMMENT	*****	MSG: FIELD C 74					*****
COMMENT	*****	SCRNSLCT: FIELD C 1					*****
COMMENT	*****	UPDATE: FIELD C 1 INIT 'U'					*****
COMMENT	*****	BACKORD: FIELD C 1 INIT 'B'					*****
COMMENT	*****	MENU: FIELD C 1 INIT 'M'					*****
COMMENT	*****	ERROR: FIELD C 1 INIT 'E'					*****
COMMENT	*****	FIRSTOCC: FIELD P 2					*****
COMMENT	*****	LASTOCC: FIELD P 2					*****
COMMENT	*****	COUNTX: FIELD P 2					*****
COMMENT	*****	QANONHND: FIELD C 7					*****
COMMENT	*****						*****

STMT TYPE	FIELD NAME	FIELD LENGTH	FLD TYP	DEC PLC	OUTPT EDIT	EDIT LGTH	INITIAL VALUE	COMMENTS	SEQUENCE NUMBER
(TF)	(SYNCFD1)	(7)	(C)						
(TF)	(SYNCFD2)	(7)	(C)						
(TF)	(SCREENNO)	(2)	(Z)						
(TF)	(NOSCREEN)	(2)	(Z)						
(TF)	(MSG)	(74)	(C)						
(TF)	(SCRNSLCT)	(1)	(C)						
(TF)	(UPDATE)	(1)	(C)			(U)			
(TF)	(BACKORD)	(1)	(C)			(B)			
(TF)	(MENU)	(1)	(C)			(M)			
(TF)	(ERROR)	(1)	(C)			(E)			
(TF)	(FIRSTOCC)	(2)	(P)						
(TF)	(LASTOCC)	(2)	(P)						
(TF)	(COUNTX)	(2)	(P)						
(TF)	(QANONHND)	(7)	(C)						

DEC 05, 2003 13.54.15

* SCREEN CONTROL *

INPUT SCREEN	STMT TYPE	SEQ NO.	LOG LVL	CON CTR	.OPERAND A.. QLF	OPER ATN	OPERAND B..... QLF	OUTPUT SCREEN	.OPERAND C.. QLF	PARTL FLD FLD/CNST	SEQUENCE 73 TO 80
(ADDEKOD4)	(SC)					(OS)				(ORDMENU4)	

SCREEN NAME	STMT TYPE	CAT ACTIN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80
(ADDEKOD4)	(SD)	(R)		()	(@)							

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....+..	<-TEXT (1-57)	SEQUENCE 73 TO 80
(ADDEKOD4,SL)	(1)	(0)		(ADDEKOD4			
(ADDEKOD4,SL)	(5)	(0)		(	CUSTOMER NUMBER		
(ADDEKOD4,SL)	(7)	(0)		(	ORDER NUMBER		
(ADDEKOD4,SL)	(9)	(0)		(	ITEM NUMBER		

Figure A-2 Program Listings for Sample Applications (Page 82 of 84)

```

(ADDEKOD4,SL) ( 11) (0) ( BACK-ORDER NUMBER )
(ADDEKOD4,SL) ( 13) (0) ( QUANTITY )
(ADDEKOD4,SL) ( 17) (0) ( $$$$$$$$$$$$$$$$$$$$$$$$$$$$ )
(ADDEKOD4,SL) ( 18) (0) ( $ ADD BACK-ORDER=ENTER $ )
(ADDEKOD4,SL) ( 19) (0) ( $ RETURN TO MENU=PF10 $ )
(ADDEKOD4,SL) ( 20) (0) ( $$$$$$$$$$$$$$$$$$$$$$$$$$$$ )
** M$WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES 73 TO 80
-----
(ADDEKOD4,SS) ( 1) ( 1)
-----
V I R C E P M N A L C PARTIAL
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80
-----
(ADDEKOD4,S0) ( 1) (1,CUSTNO ) (I) (P) (NM) (CUSTNO )
(ADDEKOD4,S1) (NE) (C)
(ADDEKOD4,S1) (MS) (C,CUSTOMER NUMBER REQUIRED)
(ADDEKOD4,S0) ( 1) (S) (N,CUSTNO ) (I) (X)
(ADDEKOD4,S1) (M) (IF (F,ATTNID ) (EQ) (C,ENTER ) )
(ADDEKOD4,S0) ( 1) (1,ORDERNO ) (I) (P) (NM) (ORDERNO )
(ADDEKOD4,S1) (NE) (C)
(ADDEKOD4,S1) (MS) (C,ORDER NUMBER REQUIRED )
(ADDEKOD4,S0) ( 1) (S) (N,ORDERNO ) (I) (X)
(ADDEKOD4,S1) (M) (IF (F,ATTNID ) (EQ) (C,ENTER ) )
(ADDEKOD4,S0) ( 1) (1,ITEMNO ) (B) (Y) (X) (NM) (ITEMNO )
(ADDEKOD4,S1) (U) (IF (F,ATTNID ) (EQ) (C,ENTER ) )
(ADDEKOD4,S0) ( 1) (S) (N,ITEMORD ) (I) (Y) (X)
(ADDEKOD4,S1) (M) (IF (F,ATTNID ) (EQ) (C,ENTER ) )
(ADDEKOD4,S0) ( 1) (1,BKORDRNO) (I) (X) (NM) (BKORDRNO )
(ADDEKOD4,S1) (I) (IF (F,ATTNID ) (EQ) (C,ENTER ) )
(ADDEKOD4,S0) ( 1) (N,QTYBKORD) (I) (A) (NM) (QTYBKORD )
(ADDEKOD4,S1) (NE) (C)
(ADDEKOD4,S1) (A,BQ) (QTYBKORD )
(ADDEKOD4,S1) (MS) (C,QUANTITY BACK-ORDERED IN)
(ADDEKOD4,S1) (MS) (C,CORRECT - REENTER )
(ADDEKOD4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)
-----
SCREEN STMT CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN PROC TERMINAL SEQUENCE
NAME TYPE ACTN TYPE DESIG DESIG FORMAT CONTROL CODE ID NULL DEVICE 73 TO 80
-----
(ORDMENU4) (SD) (R) ( ) (B) (M510 ) (M510 )
-----
SCREEN STMT LINE CONT- EXIN ....+...1...+...2...+...3...+...4...+...5...+... <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6...+...7...+...8...+...9...+...0...+...1... <-TEXT (58-114) 73 TO 80
-----
(ORDMENU4,SL) ( 1) (0) ( ORDMENU4 )
(ORDMENU4,SL) ( 3) (0) ( CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS:)
(ORDMENU4,SL) ( 5) (0) ( PF1---CUSTOMER ADD/UPDATE )
(ORDMENU4,SL) ( 7) (0) ( PF2---ITEM BROWSE BY PRODUCT KEY )
(ORDMENU4,SL) ( 9) (0) ( PF3---BACK-ORDERED ITEM )
(ORDMENU4,SL) ( 11) (0) ( PF4---ITEM ADD/UPDATE )
(ORDMENU4,SL) ( 13) (0) ( PF5---END TRANSACTION )
(ORDMENU4,SL) ( 16) (0) ( ENTER CUSTOMER OR ITEM NUMBER )
(ORDMENU4,SL) ( 18) (0) ( (ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2) )
(ORDMENU4,SL) ( 18) (1) ( IS USED)
** M$WL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES 73 TO 80
-----
(ORDMENU4,SS) ( 1) ( 1)
-----
V I R C E P M N A L C PARTIAL
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80
-----

```

Figure A-2 Program Listings for Sample Applications (Page 83 of 84)


```

(ORDMENU4,S0) ( 1) (N,CUSTNO ) (I) (X) (NM) (ITEMNO )
(ORDMENU4,S1) (V) (IF (S,PFKEY ) (EQ) (C,PF001 )
(ORDMENU4,S0) ( 1) (S) (1,ITEMNO ) (I) (X)
(ORDMENU4,S1) (M) (IF (S,PFKEY ) (EQ) (C,PF003 )
(ORDMENU4,S1) (V) (IF (S,PFKEY ) (EQ) (C,PF004 )
(ORDMENU4,S0) ( 1) (V) (I) (N) (NM) (PFKEY )
(ORDMENU4,S1) (IN) (5 )
(ORDMENU4,S1) (EQ) (C,PF001 )
(ORDMENU4,S1) (1,O,EQ) (C,PF002 )
(ORDMENU4,S1) (1,O,EQ) (C,PF003 )
(ORDMENU4,S1) (1,O,EQ) (C,PF004 )
(ORDMENU4,S1) (1,O,EQ) (C,PF005 )
(ORDMENU4,S0) ( 1) (A) (F,MSGLINE ) (H) (*SYSTEM*)

** MK5TJ05 TYPE 0 THE FOLLOWING CICS TS SYSTEM DEFINITION UTILITY INPUT STATEMENTS ARE REQUIRED FOR THIS APPLICATION.
    DEFINE GROUP(???),DUMP(NO),TWASIZE(00000),TRANSACTION(M509),
    INDOUBT(BACKOUT),PROGRAM(M509)
    DEFINE GROUP(???),LANGUAGE(ASSEMBLER),PROGRAM(M509)
** MK5TJ03 TYPE 0 THE IMS PSB GENERATION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -
    PCB TYPE=DB,POS=M,KEYLEN=0017,PROCOPT=AP,DBDNAME=CUSTOMDB
    SENSEG NAME=CUSTOMER
    SENSEG NAME=ORDER,PARENT=CUSTOMER
    SENSEG NAME=ITEMORD,PARENT=ORDER
    PCB TYPE=DB,POS=M,KEYLEN=0014,PROCOPT=AP,DBDNAME=ITEMDB
    SENSEG NAME=ITEMROOT
    SENSEG NAME=BKORDERS,PARENT=ITEMROOT
    PSBGEN LANG=ASSEM,MAXQ=002,PSBNAME=M509
    END
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - QANONHND COUNIX LASTOCC FIRSTOCC ERROR MENU .
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - BACKORD UPDATE SCRNSLCT MSG NOSCREEN SCREENNO.
** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - SYNCFLD2 SYNCFLD1 .

** MK5W204 TYPE 0 NUMBER OF MESSAGES PRINTED IS 8.
    M5INPUT - 23 RECORDS INPUT
    M5PUNCH - 284 RECORDS OUTPUT
    M5MAP - 0 RECORDS OUTPUT
    M5TPTEBL - 12 RECORDS OUTPUT
    M5PREP - 0 RECORDS OUTPUT
    M5COMMAR - 27 RECORDS OUTPUT
    M5LIST - 246 RECORDS OUTPUT
    75 TRACKS ASSIGNED TO M5LIB - 23 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 800.
** MK5W209 TYPE 0 7760 BYTES OF TYPE A STORAGE USED DURING APPLICATION GENERATION

```

Figure A-2 Program Listings for Sample Applications (Page 84 of 84)

Order/Item Application Code Design Manual

The Design Manual is a comprehensive set of reports you create that provides summary information about a generated application. Including an RP statement with the parameter MANUAL in your application generation run tells the VISION:Transact generator to create several summary reports that form the core of the Design Manual. You then augment these reports with screen definition glossaries and other application generation reports to document screen-to-screen flow, data base usage, and field usage.

VISION:Transact generates the following reports for the Design Manual: Cover Page, Table of Contents, Screen-to-Screen Flow Summary, Data Base Usage Summary, Field Reference Index, and for each screen, Screen Synchronized Data Base Summary and Detailed Cross Reference.

The Cover Page indicates the program name (load module name), the transaction name, the application type (CICS or IMS TM), and the target environment (IMS TM, CICS, OS/VS, or CICS/DOS/VS), in addition to a line for your name as author of the application.

The Table of Contents lists all the reports in the Design Manual, prefaced with a section number, including the reports that you add to complete the manual. Note that the reports are not produced in the order shown on the Table of Contents. Therefore, you must rearrange the reports so they appear in the order specified. Also, you must number the pages of the report, if necessary.

The Screen-to-Screen Flow Summary graphically shows which screens can be output from a given input screen. The flow is presented in a matrix, where each row of the matrix corresponds to an input screen and each column to an output screen. For each input screen, an X is entered in the appropriate column for each of its possible output screens.

The Data Base Usage Summary shows how each data base is used in the application. Among the information provided is:

- The file name (FILE, FILE1, FILE2, . . . , FILE9).
- The type of file (DL/I, VSAM, etc.).
- The FD name.

- The names of segments referenced.
- How the data base is used (screen/dynamic synchronization pseudo file).
- If the data base is dynamically synchronized, the synchronizing field name.

The Field Reference Index lists all fields referenced in the application sequenced by field name within qualifier. If the field is referenced in Screen Control statements, YES appears in the SCREEN CONTROL column of the report. The names of all screens that reference the field are also listed in the SCREEN NAME columns. (A field is considered referenced by a screen if it appears on an Sn statement or in a request associated with the screen.) If a field can be modified by a screen, one of three characters will be printed in the MOD column. If the field can be modified in at least one request, an R is printed. If the field can be modified by a field action, via Sn statements, an S is printed. If the field can be modified by both, a B is printed.

The Screen Synchronized Data Base Summary illustrates the structure associated with the input screen. This report is only produced for screens that can be input to the application. For each screen, it lists the data base segments matched by the screen segment and all possible segment actions.

The Detailed Cross Reference is produced for each screen in the application. All fields referenced by that screen (in Sn statements or in a request associated with the screen) appear in field name sequence within qualifier. For each field the following data is listed:

- The name and type of the requests that reference the field, and an indication of whether it is modified by the request.
- If the field is referenced by VISION:Transact interaction processing (input validation, input processing, or output processing), then **INPVAL, **INPROC, or **OUTPUT appears instead of a request name.

Order/Item Application Code Design Manual Listings

[Figure B-1](#) shows the order/item application code design manual listings.

```

DEC 04, 2003 14.33.04

*****
* Advantage VISION:Transact for z/OS - Version 7.8 *
* (C) 2003 Computer Associates International, Inc. (CA) *
*****

M5 RELEASE 7.8, 12/04/2003                                     Build Number = 103338-14.18.32.9

Advantage VISION:Transact APARs and PTFs Applied

NONE.

Common Library Subsystem Information

COMLIB Version C4.5  LIBRARY ACCESS METHOD = B/V  RESERVE = NO  ITEM TRACKING = 0

COMLIB System Modifications Applied

201,202,203,204,205,206,207,208,209,210,212,213,214,215,216,218

```

Figure B-1 Order/Item Application Code Design Manual Listings (Page 1 of 24)


```

        LET T.COUNTX = T.COUNTX + 1
        IF T.COUNTX > 1
            MSG 'ONLY ONE 'X'' PER PF6'
        END
    END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST FINDX .

***** BEGIN ASL PROCEDURE ONEXCUPD.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE MAKES SURE THAT AT LEAST ONE 'X'
; WAS ENTERED IF PF6 WAS PRESSED.
;
IF F.ATTNID EQ 'PF006' AND,
    T.COUNTX NE 1
    MSG 'AT LEAST ONE 'X'' PER PF6'
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST ONEXCUPD.

***** BEGIN ASL PROCEDURE SYNCLOAD.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE SETS UP THE SYNCHRONIZATION FIELDS FOR
; THE ITEM BROWSE WHEN APPROPRIATE.
;
IF F.ATTNID EQ 'PF006'
    LET T.SYNCFD1 = ITEMORD
    LET T.SYNCFD2 = ITEMORD
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST SYNCLOAD.

***** BEGIN ASL PROCEDURE OUR1N1 .
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED
; BY THE OPERATOR FOR SCREEN ORCUPDT4
;
IF F.ATTNID EQ 'ENTER', 'PF001', 'PF010'
; IF PF1 OR PF10 HAVE BEEN PRESSED, MAKE SURE ACTION IS BLANK
    IF F.ATTNID NE 'ENTER'
        LET S.ACTION = ' '
    END
ELSE
    MSG 'INVALID KEY PRESSED - RETRY'
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OUR1N1 .

***** BEGIN ASL PROCEDURE OUR2N1 .
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE MAKES SURE THAT AN ITEM (WHOSE QUANTITY
; ORDERED HAS BEEN CHANGED) HASN'T ALREADY BEEN SHIPPED.
;
IF F.ATTNID EQ 'ENTER' AND S.ACTION EQ 'U'
; DOES THIS ITEM ALREADY EXIST ON THIS ORDER
    IF ITEMORD EQ ITEMORD
; IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED
        IF S.ITEMQTYOR EQ ITEMQTYOR
            CONTINUE
        ELSE
; IF NOT, HAS THE ITEM BEEN SHIPPED?
            IF ITEMORD EQ ITEMSHIP
                LET T.MSG = 'CANNOT CHANGE QUANTITY ORDERED '
                LET PF(T.MSG, 32, 31) = '- ITEM ALREADY SHIPPED'
                SET S.ITEMORD
                MSG T.MSG
            END
        END
    END
    IF S.ITEMPRICE EQ ' '
        LET S.ITEMPRICE = 1.ITEMPRICE
    END
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OUR2N1 .
***** BEGIN ASL PROCEDURE OUR2N2 .
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE CONTROLS THE UPDATE OF THE QUANTITY ON
; HAND FIELD IN THE ITEM DATA BASE
;
IF F.ATTNID EQ 'ENTER' AND S.ACTION EQ 'U'
; IS ITEM ALREADY ON THIS ORDER?
    IF ITEMORD EQ ITEMORD
; IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED
        IF S.ITEMQTYOR EQ ITEMQTYOR
            CONTINUE

```

Figure B-1 Order/Item Application Code Design Manual Listings (Page 3 of 24)

```

        ELSE
            LET 1.QANONHND = 1.QANONHND + ITMQTYOR
        END
    END
    LET T.QANONHND = 1.QANONHND - S.ITMQTYOR
    IF T.QANONHND LT 0
        IF ITEMORD EQ ITEMORD
            LET 1.QANONHND = 1.QANONHND - ITMQTYOR
        END
        LET 1.QANONHND = T.QANONHND * -1
        LET T.MSG = 'QUANTITY ORDERED IS '
        LET PF(T.MSG,21,7) = T.QANONHND
        LET PF(T.MSG,31,31) = 'MORE THAN QUANTITY ON HAND'
        SET S.ITMQTYOR
        MSG T.MSG
    END
    LET 1.QANONHND = T.QANONHND
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OUR2N2 .

***** BEGIN ASL PROCEDURE VLDORDAD.
***** Using ASL Translator Version 4.50 *****
; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED
; FOR THE ORDADDR SCREEN
;
IF F.ATTNID NE 'ENTER', 'PF001', 'PF006', 'PF010'
MSG 'FUNCTION KEY INVALID - RETRY'
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST VLDORDAD.

***** BEGIN ASL PROCEDURE OAR2N1 .
***** Using ASL Translator Version 4.50 *****
; IF PRICE HAS NOT BEEN ENTERED ON THIS NEW ITEM, GET
; CURRENT PRICE FROM THE DATA BASE.
;
IF S.ITMPRICE EQ ' '
LET S.ITMPRICE = 1.ITMPRICE
END
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OAR2N1 .

***** BEGIN ASL PROCEDURE OUTYPE2 .
***** Using ASL Translator Version 4.50 *****
; CALL TYPE2 PROCEDURES FOR ORDUPDT4
;
CALL OUR2N1
CALL OUR2N2
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST OUTYPE2 .

***** BEGIN ASL PROCEDURE CUTYPE1 .
***** Using ASL Translator Version 4.50 *****
; CALL TYPE 1 PROCEDURES FOR CUSTUPD4
;
CALL VALDCUPD
CALL FINDX
CALL QNEXCUPD
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST CUTYPE1 .

***** BEGIN ASL PROCEDURE TEMPS .
***** Using ASL Translator Version 4.50 *****
; TEMP FIELD DEFINITIONS
SYNCFLD1: FIELD C 7
SYNCFLD2: FIELD C 7
SCREENNO: FIELD Z 2
NOSCREEN: FIELD Z 2
MSG: FIELD C 74
SCRNSLCT: FIELD C 1
UPDATE: FIELD C 1 INIT 'U'
BACKORD: FIELD C 1 INIT 'B'
MENU: FIELD C 1 INIT 'M'
ERROR: FIELD C 1 INIT 'E'
FIRSTOCC: FIELD P 2
LASTOCC: FIELD P 2
COUNTX: FIELD P 2
QANONHND: FIELD C 7
** MK5CP01 TYPE 0 PREPROCESSING INPUT STREAM REQUEST TEMPS .

*****
* REQUEST NAME - TEMPS *
* INPUT STREAM REQUEST *

```

Figure B-1 Order/Item Application Code Design Manual Listings (Page 4 of 24)

STMT TYPE	CAT ACT	REQUEST DESCRIPTION				REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER				

(ER)						(N)								
COMMENT	*****	; TEMP FIELD DEFINITIONS				*****								
COMMENT	*****	SYNCFLD1: FIELD C 7				*****								
COMMENT	*****	SYNCFLD2: FIELD C 7				*****								
COMMENT	*****	SCREENNO: FIELD Z 2				*****								
COMMENT	*****	NOSCREEN: FIELD Z 2				*****								
COMMENT	*****	MSG: FIELD C 74				*****								
COMMENT	*****	SCRNSLCT: FIELD C 1				*****								
COMMENT	*****	UPDATE: FIELD C 1 INIT 'U'				*****								
COMMENT	*****	BACKORD: FIELD C 1 INIT 'B'				*****								
COMMENT	*****	MENU: FIELD C 1 INIT 'M'				*****								
COMMENT	*****	ERROR: FIELD C 1 INIT 'E'				*****								
COMMENT	*****	FIRSTOCC: FIELD P 2				*****								
COMMENT	*****	LASTOCC: FIELD P 2				*****								
COMMENT	*****	COUNTX: FIELD P 2				*****								
COMMENT	*****	QANONHND: FIELD C 7				*****								
COMMENT	*****					*****								

STMT TYPE	FIELD NAME	FIELD LENGTH	FLD TYP	DEC PLC	OUTPT EDIT	EDIT LGTH	INITIAL VALUE	COMMENTS	SEQUENCE NUMBER					

(TF)	(SYNCFLD1)	(7)	(C)											
(TF)	(SYNCFLD2)	(7)	(C)											
(TF)	(SCREENNO)	(2)	(Z)											
(TF)	(NOSCREEN)	(2)	(Z)											
(TF)	(MSG)	(74)	(C)											
(TF)	(SCRNSLCT)	(1)	(C)											
(TF)	(UPDATE)	(1)	(C)				(U)	)						
(TF)	(BACKORD)	(1)	(C)				(B)	)						
(TF)	(MENU)	(1)	(C)				(M)	)						
(TF)	(ERROR)	(1)	(C)				(E)	)						
(TF)	(FIRSTOCC)	(2)	(P)											
(TF)	(LASTOCC)	(2)	(P)											
(TF)	(COUNTX)	(2)	(P)											
(TF)	(QANONHND)	(7)	(C)											
DEC 04, 2003 14.33.04							*****							
							* SCREEN CONTROL *							

INPUT SCREEN	STMT TYPE	SEQ NO.	LOG LVL	CON CTR	OPERAND A.. QLF FIELD	OPER ATTN	 QLF	OPERAND B..... FIELD NAME/CONSTANT	OUTPUT SCREEN	..OPERAND C.. QLF FLD/CNST	PARTL STR	FLD NM	OP	SEQUENCE 73 TO 80

(CUSTUPD4)	(SC)					(XR)	(1,CUTYPE1	)						
(CUSTUPD4)	(SC)				(F,ATTNID)	(EQ)	(C,ENTER	)						
(CUSTUPD4)	(SC)					(OS)			(CUSTUPD4)					
(CUSTUPD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF006	)						
(CUSTUPD4)	(SC)					(OS)	(O,SYNCLoad	)	(ORDUPDT4)					
(CUSTUPD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF007	)						
(CUSTUPD4)	(SC)					(OS)			(ORDADD4)					
(CUSTUPD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF010	)						
(CUSTUPD4)	(SC)					(OS)			(ORDMENU4)					
(ORDUPDT4)	(SC)					(XR)	(1,CUR1N1	)						
(ORDUPDT4)	(SC)					(XR)	(2,OUTYPE2	)						
(ORDUPDT4)	(SC)				(F,ATTNID)	(EQ)	(C,PF001	)						
(ORDUPDT4)	(SC)					(OS)			(CUSTUPD4)					
(ORDUPDT4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF010	)						
(ORDUPDT4)	(SC)					(OS)			(ORDMENU4)					
(ORDUPDT4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,ENTER	)						
(ORDUPDT4)	(SC)					(OS)			(ORDUPDT4)					
(ORDADD4)	(SC)					(XR)	(1,VLDORDAD	)						
(ORDADD4)	(SC)					(XR)	(2,QAR2N1	)						
(ORDADD4)	(SC)				(F,ATTNID)	(EQ)	(C,PF001	)						
(ORDADD4)	(SC)					(OS)			(CUSTUPD4)					
(ORDADD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF006	)						
(ORDADD4)	(SC)					(OS)			(ORDUPDT4)					
(ORDADD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,PF010	)						
(ORDADD4)	(SC)					(OS)			(ORDMENU4)					
(ORDADD4)	(SC)				(E) (F,ATTNID)	(EQ)	(C,ENTER	)						
(ORDADD4)	(SC)					(OS)			(ORDADD4)					

Figure B-1 Order/Item Application Code Design Manual Listings (Page 5 of 24)

SCREEN NAME	SMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80
(CUSTUPD4) (SD) (R) () (@) (M501) (CUSTSET)												
SCREEN NAME	SMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	...+...1...+...2...+...3...+...4...+...5...+.. <-TEXT (1-57)						SEQUENCE 73 TO 80	
					..6...+...7...+...8...+...9...+...0...+...1... <-TEXT (58-114)							
(CUSTUPD4,SL)	(1)	(0)	(CUSTUPD4)									
(CUSTUPD4,SL)	(3)	(0)	(CUSTOMER NUMBER)									
(CUSTUPD4,SL)	(4)	(0)	(CUSTOMER NAME)									
(CUSTUPD4,SL)	(5)	(0)	(CUSTOMER PHONE)									
(CUSTUPD4,SL)	(7)	(0)	(ACT ORDER P.O. DUE INVOICE)									
(CUSTUPD4,SL)	(7)	(1)	(ORDER)									
(CUSTUPD4,SL)	(8)	(0)	(ION NUMBER DATE SELLER NUMBER DATE GENERATED)									
(CUSTUPD4,SL)	(8)	(1)	(COMPL)									
(CUSTUPD4,SL)	(9)	(0)	(- - - - -)									
(CUSTUPD4,SL)	(9)	(1)	(- - - - -)									
(CUSTUPD4,SL)	(11)	(0)	(- - - - -)									
(CUSTUPD4,SL)	(11)	(1)	(- - - - -)									
(CUSTUPD4,SL)	(13)	(0)	(- - - - -)									
(CUSTUPD4,SL)	(13)	(1)	(- - - - -)									
(CUSTUPD4,SL)	(15)	(0)	(- - - - -)									
(CUSTUPD4,SL)	(15)	(1)	(- - - - -)									
(CUSTUPD4,SL)	(17)	(0)	(- - - - -)									
(CUSTUPD4,SL)	(17)	(1)	(- - - - -)									
(CUSTUPD4,SL)	(19)	(0)	(- - - - -)									
(CUSTUPD4,SL)	(19)	(1)	(- - - - -)									
(CUSTUPD4,SL)	(21)	(0)	(NEXT SCREEN OF ORDERS=PA1 UPDATE=U ACT + ENTER)									
(CUSTUPD4,SL)	(22)	(0)	(ORDER DETAILS=X ACT + PF6 ADD ORDER=PF7)									
(CUSTUPD4,SL)	(22)	(1)	(RETURN TO MENU=PF10)									
** M5SWL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.												
SCREEN NAME	SMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES									SEQUENCE 73 TO 80
(CUSTUPD4,SS) (1) (1)												
SCREEN NAME	SMT TYPE	SEG NO.	V A	IR Q	CE H	PM R	N O	ALC C	LO E	V V	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80
SOURCE/DEST / Q H U R R O U C L O E/V OPERAND B.....												
QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT												
(CUSTUPD4,S0)	(1)	(N,CUSTNO)	(B)					(X)	(NM)	(CUSTNO)		
(CUSTUPD4,S1)								(M)	(IF)	(F,ATTNID)	(EQ)	(C,PF006)
(CUSTUPD4,S1)								(M)	(IF)	(F,ATTNID)	(EQ)	(C,PF007)
(CUSTUPD4,S1)								(U)	(IF)	(F,ATTNID)	(EQ)	(C,ENTER)
(CUSTUPD4,S0)	(1)	(N,CUSTNAME)	(B)					(P)	(NM)	(CUSTNAME)		
(CUSTUPD4,S0)	(1)	(N,CUSTPH)	(B)					(P)	(NM)	(CUSTPH)		
(CUSTUPD4,S0)	(1)	(A)	(F,MSGLINE)	(H)								(*SYSTEM*)
SCREEN NAME	SMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES									SEQUENCE 73 TO 80
(CUSTUPD4,SS) (2) (5)												
SCREEN NAME	SMT TYPE	SEG NO.	V A	IR Q	CE H	PM R	N O	ALC C	LO E	V V	PARTIAL FIELD ST NO OP	SEQUENCE 73 TO 80
SOURCE/DEST / Q H U R R O U C L O E/V OPERAND B.....												
QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT												
(CUSTUPD4,S0)	(2)		(I)					(N)	(NM)	(ACTION)		
(CUSTUPD4,S0)	(2)	(N,ORDERNO)	(B)					(X)	(NM)	(ORDERNO)		
(CUSTUPD4,S1)								(U)	(IF)	(S,ACTION)	(EQ)	(C,U)
(CUSTUPD4,S1)								(M)	(IF)	(S,ACTION)	(EQ)	(C,X)
(CUSTUPD4,S0)	(2)	(N,ORDRDATE)	(B)					(P)	(NM)	(ORDRDATE)		
(CUSTUPD4,S0)	(2)	(N,ORPERSON)	(B)					(P)	(NM)	(ORPERSON)		
(CUSTUPD4,S0)	(2)	(N,ORDPONUM)	(B)					(P)	(NM)	(ORDPONUM)		
(CUSTUPD4,S0)	(2)	(N,ORDDUDAT)	(B)					(P)	(NM)	(ORDDUDAT)		

Figure B-1 Order/Item Application Code Design Manual Listings (Page 6 of 24)

(CUSTUPD4,S0) (2) (N,ORDINMGN) (B) (P) (NM) (ORDINMGN) ()
 (CUSTUPD4,S0) (2) (N,ORDCMPLT) (B) (P) (NM) (ORDCMPLT) ()

SCREEN NAME	STMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

(ORDUPDT4) (SD) (R) () ()

SCREEN NAME	STMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80

(ORDUPDT4,SL) (1) (0) (ORDUPDT4)
 (ORDUPDT4,SL) (1) (1) (UPDATE=U ACT + ENTER)
 (ORDUPDT4,SL) (2) (0) (CUSTOMER NUMBER)
 (ORDUPDT4,SL) (2) (1) (CUST UPDT SCRN=PFI)
 (ORDUPDT4,SL) (3) (0) (ORDER NUMBER)
 (ORDUPDT4,SL) (3) (1) (RETURN TO MENU=PF10)
 (ORDUPDT4,SL) (4) (0) (CUSTOMER NAME)
 (ORDUPDT4,SL) (5) (1) (QTY)
 (ORDUPDT4,SL) (6) (0) (ACT ITEM ITEM PRICE QUANTITY)
 (ORDUPDT4,SL) (6) (1) (BACK ON)
 (ORDUPDT4,SL) (7) (0) (ION NUMBER NAME ORDERED)
 (ORDUPDT4,SL) (7) (1) (ORDERED HAND)
 (ORDUPDT4,SL) (8) (0) (- - - - -)
 (ORDUPDT4,SL) (8) (1) (- - - - -)
 (ORDUPDT4,SL) (9) (0) (- - - - -)
 (ORDUPDT4,SL) (9) (1) (- - - - -)
 (ORDUPDT4,SL) (10) (0) (- - - - -)
 (ORDUPDT4,SL) (10) (1) (- - - - -)
 (ORDUPDT4,SL) (11) (0) (- - - - -)
 (ORDUPDT4,SL) (11) (1) (- - - - -)
 (ORDUPDT4,SL) (12) (0) (- - - - -)
 (ORDUPDT4,SL) (12) (1) (- - - - -)
 (ORDUPDT4,SL) (13) (0) (- - - - -)
 (ORDUPDT4,SL) (13) (1) (- - - - -)
 (ORDUPDT4,SL) (14) (0) (- - - - -)
 (ORDUPDT4,SL) (14) (1) (- - - - -)
 (ORDUPDT4,SL) (15) (0) (- - - - -)
 (ORDUPDT4,SL) (15) (1) (- - - - -)
 (ORDUPDT4,SL) (16) (0) (- - - - -)
 (ORDUPDT4,SL) (16) (1) (- - - - -)
 (ORDUPDT4,SL) (17) (0) (- - - - -)
 (ORDUPDT4,SL) (17) (1) (- - - - -)
 (ORDUPDT4,SL) (18) (0) (- - - - -)
 (ORDUPDT4,SL) (18) (1) (- - - - -)
 (ORDUPDT4,SL) (19) (0) (- - - - -)
 (ORDUPDT4,SL) (19) (1) (- - - - -)
 (ORDUPDT4,SL) (20) (0) (- - - - -)
 (ORDUPDT4,SL) (20) (1) (- - - - -)
 (ORDUPDT4,SL) (21) (0) (- - - - -)
 (ORDUPDT4,SL) (21) (1) (- - - - -)
 (ORDUPDT4,SL) (22) (0) (- - - - -)
 (ORDUPDT4,SL) (22) (1) (- - - - -)
 (ORDUPDT4,SL) (23) (0) (- - - - -)
 (ORDUPDT4,SL) (23) (1) (- - - - -)
 ** M5W14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

SCREEN NAME	STMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

(ORDUPDT4,SS) (1) (1)

SCREEN NAME	STMT TYPE	V SEG S	IR SOURCE/DEST	CE P H U R R O U D I R R T D M	ALC C L O E/V	P M N T V N O P	PARTIAL FIELD SEQUENCE
(ORDUPDT4,S0) (1)	(N,CUSTNO) (B)	(H,Y)	(X)	(NM)	(CUSTNO)	()	
(ORDUPDT4,S1)	(M)	(IF) (F,ATTNID)	(EQ)	(C,PF001)	()		
(ORDUPDT4,S1)	(M)	(IF) (F,ATTNID)	(EQ)	(C,ENTER)	()		
(ORDUPDT4,S0) (1)	(N,ORDERNO) (B)	(Y)	(X)	(NM)	(ORDERNO)	()	
(ORDUPDT4,S1)	(Q)	(IF) (F,ATTNID)	(EQ)	(C,ENTER)	()		

Figure B-1 Order/Item Application Code Design Manual Listings (Page 7 of 24)

```

(ORDUPDT4,S0) ( 1) (N,CUSTNAME) (O) (NM) (CUSTNAME) )
(ORDUPDT4,S0) ( 1) (A) (F,MSGLINE) (H) (*SYSTEM*)
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES
73 TO 80

(ORDUPDT4,SS) ( 2) ( 15)
-----
V I R C E P M N A L C PARTIAL
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ORDUPDT4,S0) ( 2) (I) (H) (N) (NM) (ACTION )
(ORDUPDT4,S0) ( 2) (N,ITEMORD) (B) (X) (NM) (ITEMORD )
(ORDUPDT4,S1) (V) (IF) (S,ACTION ) (EQ) (C,U )
(ORDUPDT4,S0) ( 2) (S) (1,ITEMNO ) (I) (X)
(ORDUPDT4,S1) (U) (IF) (S,ACTION ) (EQ) (C,U )
(ORDUPDT4,S0) ( 2) (2,ITEMNAME) (O) (NM) (ITEMNAME )
(ORDUPDT4,S0) ( 2) (N,ITEMPRICE) (B) (P) (NM) (ITEMPRICE )
(ORDUPDT4,S0) ( 2) (N,ITEMQTYOR) (B) (P) (NM) (ITEMQTYOR )
(ORDUPDT4,S0) ( 2) (N,QTYEKORD) (B) (P) (NM) (QTYEKORD )
(ORDUPDT4,S0) ( 2) (2,QANONHND) (O) (NM) (QANONHND )
-----
SCREEN STMT CAT GLOSS FIELD HILITE AUTO OUTPUT TRANSACT SCREEN PROC TERMINAL SEQUENCE
NAME TYPE ACTN TYPE DESIG DESIG FORMAT CONTROL CODE ID NULL DEVICE
73 TO 80

(ORDADD4 ) (SD) (R) ( ) (8)
-----
SCREEN STMT LINE CONT- EXTIN .....1.....2.....3.....4.....5..... <-TEXT (1-57) SEQUENCE
NAME TYPE NO. INUE ATTR ..6.....7.....8.....9.....0..... <-TEXT (58-114) 73 TO 80

(ORDADD4 ,SL) ( 1) (0) ( ORDADD4 )
(ORDADD4 ,SL) ( 1) (1) (ADD NEW DATA=ENTER )
(ORDADD4 ,SL) ( 2) (1) (CUST UPDT SCRN=PF1 )
(ORDADD4 ,SL) ( 3) (0) ( CUSTOMER NUMBER )
(ORDADD4 ,SL) ( 3) (1) (VIEW ADDED ITEMS=PF6 )
(ORDADD4 ,SL) ( 4) (0) ( CUSTOMER NAME )
(ORDADD4 ,SL) ( 4) (1) (RETURN TO MENU=PF10 )
(ORDADD4 ,SL) ( 5) (0) ( CUSTOMER PHONE )
(ORDADD4 ,SL) ( 7) (0) ( ORDER NUMBER )
(ORDADD4 ,SL) ( 8) (0) ( P.O. NUMBER )
(ORDADD4 ,SL) ( 10) (0) ( ITEM PRICE QUANTITY BACK )
(ORDADD4 ,SL) ( 11) (0) ( NUMBER ORDERED ORDERED )
(ORDADD4 ,SL) ( 12) (0) ( )
(ORDADD4 ,SL) ( 13) (0) ( )
(ORDADD4 ,SL) ( 14) (0) ( )
(ORDADD4 ,SL) ( 15) (0) ( )
(ORDADD4 ,SL) ( 16) (0) ( )
(ORDADD4 ,SL) ( 17) (0) ( )
(ORDADD4 ,SL) ( 18) (0) ( )
(ORDADD4 ,SL) ( 19) (0) ( )
(ORDADD4 ,SL) ( 20) (0) ( )
(ORDADD4 ,SL) ( 21) (0) ( )
(ORDADD4 ,SL) ( 22) (0) ( )
(ORDADD4 ,SL) ( 23) (0) ( )
** M$NL14 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.
-----
SCREEN STMT SEGMENT NO. OF SEQUENCE
NAME TYPE NO. OCCURRENCES
73 TO 80

(ORDADD4 ,SS) ( 1) ( 1)
-----
V I R C E P M N A L C PARTIAL
SCREEN STMT SEG S SOURCE/DEST / Q H U R R O U C L O E/V .....OPERAND B..... FIELD SEQUENCE
NAME TYPE NO. A QLF FLD NAME O D I R R T D M T V N O P QLF FIELD NAME OR CONSTANT ST NO OP 73 TO 80

(ORDADD4 ,S0) ( 1) (N,CUSTNO ) (B) (Y) (X) (NM) (CUSTNO )
(ORDADD4 ,S1) (M) (IF) (F,ATTNID ) (EQ) (C,PF001 )
(ORDADD4 ,S1) (M) (IF) (F,ATTNID ) (EQ) (C,PF006 )
(ORDADD4 ,S1) (M) (IF) (F,ATTNID ) (EQ) (C,ENTER )
(ORDADD4 ,S0) ( 1) (N,CUSTNAME) (O) (NM) (CUSTNAME )
(ORDADD4 ,S0) ( 1) (N,CUSTPH ) (O) (NM) (CUSTPH )

```

Figure B-1 Order/Item Application Code Design Manual Listings (Page 8 of 24)

```

(ORDADD4 ,S0) ( 1)      (N,ORDERNO ) (B)          (X)  (NM)  (ORDERNO          )
(ORDADD4 ,S1)           (M)  (IF) (F,ATTNID ) (EQ) (C,PF006      )
(ORDADD4 ,S1)           (V)  (IF) (F,ATTNID ) (EQ) (C,ENTER      )
(ORDADD4 ,S0) ( 1)      (N,ORPERSON) (B)          (P)  (NM)  (ORPERSON          )
(ORDADD4 ,S0) ( 1)      (N,ORDPONUM) (B)          (P)  (NM)  (ORDPONUM          )
(ORDADD4 ,S0) ( 1)      (N,ORDDUAT) (B)           (P)  (NM)  (ORDDUAT           )
(ORDADD4 ,S0) ( 1)      (N,ORDRDATE) (B)          (P)  (NM)  (ORDRDATE          )
(ORDADD4 ,S0) ( 1) (A)  (F,MSGLINE ) (H)                                (*SYSTEM*)

```

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

```

(ORDADD4 ,SS) ( 2)      ( 11)

```

SCREEN NAME	SIMT TYPE	SEG NO.	S QLF	V FLD	IR NAME	CE O D	PM I R	LN R R	ALC T	LC V N	OP QLF	OPERAND B. FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO	SEQUENCE 73 TO 80
(ORDADD4 ,S0) (2)			(N,ITEMORD)	(I)					(X)	(NM)	(ITEMORD)			
(ORDADD4 ,S1)									(I)	(IF)	(F,ATTNID)	(EQ) (C,ENTER)		
(ORDADD4 ,S0) (2) (S)			(1,ITEMNO)	(I)					(X)					
(ORDADD4 ,S1)									(U)	(IF)	(F,ATTNID)	(EQ) (C,ENTER)		
(ORDADD4 ,S0) (2)			(N,ITEMPRICE)	(I)					(R)	(NM)	(ITEMPRICE)			
(ORDADD4 ,S0) (2)			(N,ITEMQTYOR)	(I)					(R)	(NM)	(ITEMQTYOR)			
(ORDADD4 ,S0) (2) (S)			(1,QANONHND)	(I)					(S)					
(ORDADD4 ,S1)										(LE)	(1,QANONHND)			
(ORDADD4 ,S1)										(MS)	(C,NOT ENOUGH IN STOCK)			
(ORDADD4 ,S0) (2)			(N,QTYBKORD)	(I)					(R)	(NM)	(QTYBKORD)			

SCREEN NAME	SIMT TYPE	CAT ACTN	GLOSS TYPE	FIELD DESIG	HILITE DESIG	AUTO FORMAT	OUTPUT CONTROL	TRANSACTION CODE	SCREEN ID	PROC NULL	TERMINAL DEVICE	SEQUENCE 73 TO 80

```

(ORDMENU4) (SD) (R)      ( _ ) ( @ )      (M510 ) (M510 )

```

SCREEN NAME	SIMT TYPE	LINE NO.	CONT- INUE	EXTN ATTR	1.....2.....3.....4.....5.....	<-TEXT (1-57)	SEQUENCE 73 TO 80
					..6.....7.....8.....9.....0.....1.....	<-TEXT (58-114)	

```

(ORDMENU4,SL) ( 1) (0)      ( ORDMENU4          )
(ORDMENU4,SL) ( 3) (0)      ( CHOOSE SCREEN USING THE FOLLOWING FUNCTION KEYS: )
(ORDMENU4,SL) ( 5) (0)      ( PF1---CUSTOMER ADD/UPDATE          )
(ORDMENU4,SL) ( 7) (0)      ( PF2---ITEM BROWSE BY PRODUCT KEY   )
(ORDMENU4,SL) ( 9) (0)      ( PF3---BACK-ORDERED ITEM           )
(ORDMENU4,SL) (11) (0)      ( PF4---ITEM ADD/UPDATE              )
(ORDMENU4,SL) (13) (0)      ( PF5---END TRANSACTION              )
(ORDMENU4,SL) (16) (0)      ( ENTER CUSTOMER OR ITEM NUMBER     )
(ORDMENU4,SL) (18) (0)      ( (ITEM NUMBER MAY BE ONE LETTER PRODUCT CODE IF PF2) )
(ORDMENU4,SL) (18) (1)      ( IS USED )
** M5W114 TYPE 0 FIELD DESIGNATORS HAVE BEEN AUTOMATICALLY INSERTED FOR THE MESSAGE LINE.

```

SCREEN NAME	SIMT TYPE	SEGMENT NO.	NO. OF OCCURRENCES	SEQUENCE 73 TO 80

```

(ORDMENU4,SS) ( 1)      ( 1)

```

SCREEN NAME	SIMT TYPE	SEG NO.	S QLF	V FLD	IR NAME	CE O D	PM I R	LN R R	ALC T	LC V N	OP QLF	OPERAND B. FIELD NAME OR CONSTANT	PARTIAL FIELD ST NO	SEQUENCE 73 TO 80
(ORDMENU4 ,S0) (1)			(N,CUSTNO)	(I)					(X)	(NM)	(ITEMNO)			
(ORDMENU4 ,S1)									(V)	(IF)	(S,PFKEY)	(EQ) (C,PF001)		
(ORDMENU4 ,S0) (1) (S)			(1,ITEMNO)	(I)					(X)					
(ORDMENU4 ,S1)									(M)	(IF)	(S,PFKEY)	(EQ) (C,PF003)		
(ORDMENU4 ,S1)									(V)	(IF)	(S,PFKEY)	(EQ) (C,PF004)		
(ORDMENU4 ,S0) (1) (V)					(I)				(N)	(NM)	(PFKEY)			
(ORDMENU4 ,S1)										(LN)	(5)			
(ORDMENU4 ,S1)										(EQ)	(C,PF001)			

Figure B-1 Order/Item Application Code Design Manual Listings (Page 9 of 24)

```

(ORDMENU4,S1)                                (1,O,EQ) (C,PF002)      )
(ORDMENU4,S1)                                (1,O,EQ) (C,PF003)      )
(ORDMENU4,S1)                                (1,O,EQ) (C,PF004)      )
(ORDMENU4,S1)                                (1,O,EQ) (C,PF005)      )
(ORDMENU4,S0) ( 1) (A) (F,MSGLINE ) (H)                                (*SYSTEM*)

```

```

DEC 04, 2003 14.33.04                      *****
                                           * REQUEST NAME - VALDCUPD *
                                           * INPUT STREAM REQUEST *
                                           *****

```

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(S)
COMMENT	*****	; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED					*****
COMMENT	*****	;					*****
COMMENT	*****	IF F.ATINID NE 'ENTER', 'PF006', 'PF007', 'PF010'					*****
COMMENT	*****	MSG 'INVALID KEY PRESSED - RETRY'					*****
COMMENT	*****	END					*****
COMMENT	*****						*****

STMT NUMB	SEQ NO.	LOG LEV	CON CTR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER
(1)	(100)			(F,ATINID)	(NE)	(C,ENTER				)
(2)	(101)				(NS)	(END				)
(3)	(102)			(F,ATINID)	(NE)	(C,PF006				)
(4)	(103)				(NS)	(END				)
(5)	(104)			(F,ATINID)	(NE)	(C,PF007				)
(6)	(105)				(NS)	(END				)
(7)	(106)			(F,ATINID)	(NE)	(C,PF010				)
(8)	(107)				(NS)	(END				)
(9)	(108)				(NS)	(C,INVALID KEY PRESSED - RETRY				)
** MK5BX01 TYPE 0 DECODING REQUEST VALDCUPD FOR SCREEN CUSTUPD4 .										

```

DEC 04, 2003 14.33.04                      *****
                                           * REQUEST NAME - FINDX *
                                           * INPUT STREAM REQUEST *
                                           *****

```

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(S)
COMMENT	*****	; THIS PROCEDURE MAKES SURE THAT ONLY ONE 'X' IS ENTERED					*****
COMMENT	*****	; WHEN PF6 IS PRESSED ON SCREEN CUSTUPD4					*****
COMMENT	*****	;					*****
COMMENT	*****	IF F.ATINID EQ 'PF006' AND,					*****
COMMENT	*****	S.ACTION EQ 'X'					*****
COMMENT	*****	LET T.COUNTX = T.COUNTX + 1					*****
COMMENT	*****	IF T.COUNTX > 1					*****
COMMENT	*****	MSG 'ONLY ONE 'X' PER PF6'					*****
COMMENT	*****	END					*****
COMMENT	*****	END					*****
COMMENT	*****						*****

STMT NUMB	SEQ NO.	LOG LEV	CON CTR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER
(1)	(100)			(F,ATINID)	(EQ)	(C,PF006				)
(2)	(101)				(NS)	(END				)
(3)	(102)			(S,ACTION)	(EQ)	(C,X				)
(4)	(103)				(NS)	(END				)
(5)	(104)			(T,COUNTX)	(+)	(D,1		(T,COUNTX)		)
(6)	(105)			(T,COUNTX)	(GT)	(D,1				)
(7)	(106)				(NS)	(END				)
(8)	(107)				(NS)	(C,ONLY ONE 'X' PER PF6				)
** MK5BX01 TYPE 0 DECODING REQUEST FINDX FOR SCREEN CUSTUPD4 .										

```

*****

```

Figure B-1 Order/Item Application Code Design Manual Listings (Page 10 of 24)

[illegible][illegible]

SMT	CAT			REQ	SCREEN	BACK	ER	RNT	LINE
TYPE	ACT	REQUEST	DESCRIPTION	TYPE	NAME	CONTROL	TMP		NUMBER
(ER)									(1)
COMMENT	*****		; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED						*****
COMMENT	*****		; BY THE OPERATOR FOR SCREEN ORUPTD4						*****
COMMENT	*****		;						*****
COMMENT	*****		IF F.ATTNID EQ 'ENTER', 'PF001', 'PF010'						*****
COMMENT	*****		; IF PF1 OR PF10 HAVE BEEN PRESSED, MAKE SURE ACTION IS BL						*****
COMMENT	*****		IF F.ATTNID NE 'ENTER'						*****
COMMENT	*****		LET S.ACTION = ' '						*****
COMMENT	*****		END						*****

Figure B-1 Order/Item Application Code Design Manual Listings (Page 11 of 24)

```

COMMENT ***** ELSE *****
COMMENT ***** MSG 'INVALID KEY PRESSED - RETRY' *****
COMMENT ***** END *****
COMMENT *****

-----
SIMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD SEQ
NUMB NO. LEV CTR QLF FIELD NAME ATN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OPR NUMBER
-----
( 1)(100) (F,ATTNID ) (EQ) (C,ENTER )
( 2)(101) (GS) (106 )
( 3)(102) (F,ATTNID ) (EQ) (C,PF001 )
( 4)(103) (GS) (106 )
( 5)(104) (F,ATTNID ) (EQ) (C,PF010 )
( 6)(105) (NS) (110 )
( 7)(106) (F,ATTNID ) (NE) (C,ENTER )
( 8)(107) (NS) (END )
( 9)(108) (R ) (C) (S,ACTION )
(10)(109) (GO) (END )
(11)(110) (MS) (C,INVALID KEY PRESSED - RETRY )
** MK5BX01 TYPE 0 DECODING REQUEST OURINI FOR SCREEN ORUPT4 .

*****
DEC 04, 2003 14.33.04 * REQUEST NAME - OUR2N1 *
* INPUT STREAM REQUEST *
*****

-----
SIMT CAT REQ SCREEN BACK BR RNT LINE
TYPE ACT REQUEST DESCRIPTION TYP NAME CONTROL TMP NUMBER
-----
(ER) (S)
COMMENT ***** ; THIS PROCEDURE MAKES SURE THAT AN ITEM (WHOSE QUANTITY *****
COMMENT ***** ; ORDERED HAS BEEN CHANGED) HASN'T ALREADY BEEN SHIPPED. *****
COMMENT ***** ; *****
COMMENT ***** IF F.ATTNID EQ 'ENTER' AND S.ACTION EQ 'U' *****
COMMENT ***** ; DOES THIS ITEM ALREADY EXIST ON THIS ORDER *****
COMMENT ***** IF ITEMORD EQ ITEMORD *****
COMMENT ***** ; IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED *****
COMMENT ***** IF S.ITMQTYOR EQ ITMQTYOR *****
COMMENT ***** CONTINUE *****
COMMENT ***** ELSE *****
COMMENT ***** ; IF NOT, HAS THE ITEM BEEN SHIPPED? *****
COMMENT ***** IF ITEMORD EQ ITEMSHIP *****
COMMENT ***** LET T.MSG = 'CANNOT CHANGE QUANTITY ORDERED ' *****
COMMENT ***** LET PF(T.MSG,32,31) = '- ITEM ALREADY SHIPPED' *****
COMMENT ***** SET S.ITEMORD *****
COMMENT ***** MSG T.MSG *****
COMMENT ***** END *****
COMMENT ***** END *****
COMMENT ***** END *****
COMMENT ***** IF S.ITMPRICE EQ ' ' *****
COMMENT ***** LET S.ITMPRICE = 1.ITMPRICE *****
COMMENT ***** END *****
COMMENT ***** END *****
COMMENT *****

-----
SIMT SEQ LOG CON ..OPERAND A... OPER .....OPERAND B..... ....RESULT.... PARTIAL FIELD SEQ
NUMB NO. LEV CTR QLF FIELD NAME ATN QLF FIELD NAME OR CONSTANT QLF FIELD NAME STRT NUMB OPR NUMBER
-----
( 1)(100) (F,ATTNID ) (EQ) (C,ENTER )
( 2)(101) (NS) (END )
( 3)(102) (S,ACTION ) (EQ) (C,U )
( 4)(103) (NS) (END )
( 5)(104) (ITEMORD ) (EQ) (ITEMORD )
( 6)(105) (NS) (115 )
( 7)(106) (S,ITMQTYOR ) (EQ) (ITMQTYOR )
( 8)(107) (NS) (109 )
( 9)(108) (GO) (END )
(10)(109) (ITEMORD ) (EQ) (ITEMSHIP )
(11)(110) (NS) (115 )
(12)(111) (R ) (C,CANNOT CHANGE QUANTITY ORDERED ) (T,MSG )
(13)(112) (R ) (C,- ITEM ALREADY SHIPPED ) (T,MSG ) (32) (31) (C)
(14)(113) (S,ITEMORD ) (SE) ( )
(15)(114) (MS) (T,MSG )
(16)(115) (S,ITMPRICE) (EQ) (C)
(17)(116) (NS) (END )
(18)(117) (R ) (1,ITMPRICE ) (S,ITMPRICE)
** MK5BX01 TYPE 0 DECODING REQUEST OUR2N1 FOR SCREEN ORUPT4 .

```

Figure B-1 Order/Item Application Code Design Manual Listings (Page 12 of 24)

DEC 04, 2003 14.33.04

* REQUEST NAME - OURZM2 *
* INPUT STREAM REQUEST *

SIMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR RNT CONTROL TMP	LINE NUMBER
(ER)						(S)
COMMENT	*****	; THIS PROCEDURE CONTROLS THE UPDATE OF THE QUANTITY ON	*****			*****
COMMENT	*****	; HAND FIELD IN THE ITEM DATA BASE	*****			*****
COMMENT	*****	;	*****			*****
COMMENT	*****	IF F.ATINID EQ 'ENTER' AND S.ACTION EQ 'U'	*****			*****
COMMENT	*****	; IS ITEM ALREADY ON THIS ORDER?	*****			*****
COMMENT	*****	IF ITEMORD EQ ITEMORD	*****			*****
COMMENT	*****	;IF SO, HAS THE QUANTITY ORDERED BEEN CHANGED	*****			*****
COMMENT	*****	IF S.ITMOTYOR EQ ITMOTYOR	*****			*****
COMMENT	*****	CONTINUE	*****			*****
COMMENT	*****	ELSE	*****			*****
COMMENT	*****	LET 1.QANONHND = 1.QANONHND + ITMOTYOR	*****			*****
COMMENT	*****	END	*****			*****
COMMENT	*****	END	*****			*****
COMMENT	*****	LET T.QANONHND = 1.QANONHND - S.ITMOTYOR	*****			*****
COMMENT	*****	IF T.QANONHND LT 0	*****			*****
COMMENT	*****	IF ITEMORD EQ ITEMORD	*****			*****
COMMENT	*****	LET 1.QANONHND = 1.QANONHND - ITMOTYOR	*****			*****
COMMENT	*****	END	*****			*****
COMMENT	*****	LET 1.QANONHND = T.QANONHND * -1	*****			*****
COMMENT	*****	LET T.MSG = 'QUANTITY ORDERED IS '	*****			*****
COMMENT	*****	LET PF(T.MSG,21,7) = T.QANONHND	*****			*****
COMMENT	*****	LET PF(T.MSG,31,31) = 'MORE THAN QUANTITY ON HAND'	*****			*****
COMMENT	*****	SET S.ITMOTYOR	*****			*****
COMMENT	*****	MSG T.MSG	*****			*****
COMMENT	*****	END	*****			*****
COMMENT	*****	LET 1.QANONHND = T.QANONHND	*****			*****
COMMENT	*****	END	*****			*****
COMMENT	*****		*****			*****

SIMT NUMB	SEQ NO.	LOG CTR	CON LEV	OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB OPR	SEQ NUMBER
(	1	(100)		(F,ATINID)	(EQ)	(C,ENTER	)		
(	2	(101)			(NS)	(END	)		
(	3	(102)		(S,ACTION)	(EQ)	(C,U	)		
(	4	(103)			(NS)	(END	)		
(	5	(104)		(ITEMORD)	(EQ)	(ITEMORD	)		
(	6	(105)			(NS)	(110	)		
(	7	(106)		(S,ITMOTYOR)	(EQ)	(ITMOTYOR	)		
(	8	(107)			(NS)	(109	)		
(	9	(108)			(GO)	(END	)		
(	10	(109)		(1,QANONHND)	(+)	(ITMOTYOR	)	(1,QANONHND)	
(	11	(110)		(1,QANONHND)	(-)	(S,ITMOTYOR	)	(T,QANONHND)	
(	12	(111)		(T,QANONHND)	(LT)	(D,0	)		
(	13	(112)			(NS)	(122	)		
(	14	(113)		(ITEMORD)	(EQ)	(ITEMORD	)		
(	15	(114)			(NS)	(116	)		
(	16	(115)		(1,QANONHND)	(-)	(ITMOTYOR	)	(1,QANONHND)	
(	17	(116)		(T,QANONHND)	(*)	(D,-1	)	(1,QANONHND)	
(	18	(117)			(R)	(C,QUANTITY ORDERED IS	)	(T,MSG)	

SIMT NUMB	SEQ NO.	LOG CTR	CON LEV	OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB OPR	SEQ NUMBER
(	19	(118)			(R)	(T,QANONHND	)	(T,MSG)	(21) (7) (C)
(	20	(119)			(R)	(C,MORE THAN QUANTITY ON HAND	)	(T,MSG)	(31) (31) (C)
(	21	(120)		(S,ITMOTYOR)	(SE)				
(	22	(121)			(MS)	(T,MSG	)		
(	23	(122)			(R)	(T,QANONHND	)	(1,QANONHND)	
** M5BX01 TYPE 0 DECODING REQUEST OURZM2 FOR SCREEN ORCDPT4 .									

Figure B-1 Order/Item Application Code Design Manual Listings (Page 13 of 24)


```

DEC 04, 2003 14.33.04          * REQUEST NAME - VLDORDAD *
                                * INPUT STREAM REQUEST *
                                *****

-----
STMT  CAT                                REQ  SCREEN  BACK BR RNT  LINE
TYPE ACT REQUEST DESCRIPTION          TYP  NAME    CONTROL TMP  NUMBER
-----
(ER)
COMMENT ***** ; THIS PROCEDURE VALIDATES THE FUNCTION KEY PRESSED ***** (1)
COMMENT ***** ; FOR THE ORDADDR SCREEN *****
COMMENT ***** ; *****
COMMENT ***** IF F,ATINID NE 'ENVER', 'PF001', 'PF006', 'PF010' *****
COMMENT ***** MSG 'FUNCTION KEY INVALID - RETRY' *****
COMMENT ***** END *****
COMMENT ***** *****

-----
STMT  SEQ  LOG CON ..OPERAND A... OPER  ....OPERAND B..... ....RESULT.... PARTIAL FIELD  SEQ
NUMB NO.  LEV CTR QLF FIELD NAME  ATN   QLF  FIELD NAME OR CONSTANT  QLF FIELD NAME STRT NUMB OPR  NUMBER
-----
( 1) (100)          (F,ATINID )  (NE)  (C,ENVER                      )
( 2) (101)          (NS)          (END                          )
( 3) (102)          (F,ATINID )  (NE)  (C,PF001                      )
( 4) (103)          (NS)          (END                          )
( 5) (104)          (F,ATINID )  (NE)  (C,PF006                      )
( 6) (105)          (NS)          (END                          )
( 7) (106)          (F,ATINID )  (NE)  (C,PF010                      )
( 8) (107)          (NS)          (END                          )
( 9) (108)          (NS)          (C,FUNCTION KEY INVALID - RETRY )
** MK5BX01 TYPE 0 DECODING REQUEST VLDORDAD FOR SCREEN ORDADD4 .

*****
DEC 04, 2003 14.33.04          * REQUEST NAME - QAR2N1 *
                                * INPUT STREAM REQUEST *
                                *****

-----
STMT  CAT                                REQ  SCREEN  BACK BR RNT  LINE
TYPE ACT REQUEST DESCRIPTION          TYP  NAME    CONTROL TMP  NUMBER
-----
(ER)
COMMENT ***** ; IF PRICE HAS NOT BEEN ENTERED ON THIS NEW ITEM, GET ***** (2)
COMMENT ***** ; CURRENT PRICE FROM THE DATA BASE. *****
COMMENT ***** ; *****
COMMENT ***** IF S.ITMPRICE EQ ' ' *****
COMMENT ***** LET S.ITMPRICE = 1.ITMPRICE *****
COMMENT ***** END *****
COMMENT ***** *****

-----
STMT  SEQ  LOG CON ..OPERAND A... OPER  ....OPERAND B..... ....RESULT.... PARTIAL FIELD  SEQ
NUMB NO.  LEV CTR QLF FIELD NAME  ATN   QLF  FIELD NAME OR CONSTANT  QLF FIELD NAME STRT NUMB OPR  NUMBER
-----
( 1) (100)          (S,ITMPRICE) (EQ)  (C)
( 2) (101)          (NS)          (END                          )
( 3) (102)          (R )          (1,ITMPRICE                    ) (S,ITMPRICE)
** MK5BX01 TYPE 0 DECODING REQUEST QAR2N1 FOR SCREEN ORDADD4 .

*****
DEC 04, 2003 14.33.04          * REQUEST NAME - OUTYPE2 *
                                * INPUT STREAM REQUEST *
                                *****

-----
STMT  CAT                                REQ  SCREEN  BACK BR RNT  LINE
TYPE ACT REQUEST DESCRIPTION          TYP  NAME    CONTROL TMP  NUMBER
-----
(ER)
COMMENT ***** ; CALL TYPE2 PROCEDURES FOR ORDUPDT4 ***** (2)
COMMENT ***** ; *****
COMMENT ***** CALL OUR2N1 *****
COMMENT ***** CALL OUR2N2 *****
COMMENT ***** *****

```

Figure B-1 Order/Item Application Code Design Manual Listings (Page 14 of 24)

STMT NUMB	SEQ NO.	LOG LEV	CON CIR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER
(1)	(100)				(GO)	(SUB OUR2N1			)	
(2)	(101)				(GO)	(SUB OUR2N2			)	
** MK5BX01 TYPE 0 DECODING REQUEST OUTYPE2 FOR SCREEN ORCUPDT4 .										

DEC 04, 2003 14.33.04

* REQUEST NAME - CUTYPE1 *
* INPUT STREAM REQUEST *

STMT TYPE	CAT ACT	REQUEST DESCRIPTION	REQ TYP	SCREEN NAME	BACK BR CONTROL	RNT TMP	LINE NUMBER
(ER)							(1)
COMMENT	*****	; CALL TYPE 1 PROCEDURES FOR CUSTUPD4	*****				
COMMENT	*****	;	*****				
COMMENT	*****	CALL VALDCUPD	*****				
COMMENT	*****	CALL FINDX	*****				
COMMENT	*****	CALL ONEXCUPD	*****				
COMMENT	*****		*****				

STMT NUMB	SEQ NO.	LOG LEV	CON CIR	..OPERAND A... QLF FIELD NAME	OPER ATN	OPERAND B..... QLF FIELD NAME OR CONSTANT	RESULT.... QLF FIELD NAME	PARTIAL FIELD STRT NUMB	FIELD OPR	SEQ NUMBER
(1)	(100)				(GO)	(SUB VALDCUPD			)	
(2)	(101)				(GO)	(SUB FINDX			)	
(3)	(102)				(GO)	(SUB ONEXCUPD			)	
** MK5BX01 TYPE 0 DECODING REQUEST CUTYPE1 FOR SCREEN CUSTUPD4 .										

** MK5TJ05 TYPE 0 THE FOLLOWING CICS TS SYSTEM DEFINITION UTILITY I NPUT STATEMENTS ARE REQUIRED FOR THIS APPLICATION.
 DEFINE GROUP(???) ,DUMP(NO),TWASIZE(00000),TRANSACTION(M501), *
 INDOUBT(BACKOUT),PROGRAM(M501)
 DEFINE GROUP(???) ,LANGUAGE(ASSEMBLER),PROGRAM(M501)

** MK5TJ03 TYPE 0 THE IMS PSB GENERATION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -
 PCB TYPE=DB, POS=M, KEYLEN=0021, PROCOPT=AP, DBDNAME=CUSTOMDB
 SENSEG NAME=CUSTOMER
 SENSEG NAME=ORDER, PARENT=CUSTOMER
 SENSEG NAME=SHIPINV, PARENT=ORDER
 SENSEG NAME=ITEMSHIP, PARENT=SHIPINV
 SENSEG NAME=ITEMORD, PARENT=ORDER
 PCB TYPE=DB, POS=M, KEYLEN=0007, PROCOPT=AP, DBDNAME=ITEMDB
 SENSEG NAME=ITEMROOT
 PCB TYPE=DB, POS=M, KEYLEN=0007, PROCOPT=GP, DBDNAME=ITEMDB
 SENSEG NAME=ITEMROOT
 PSBGEN LANG=ASSEM, MAXQ=016, PSBNAME=M14CSIT
 END

** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - LASTOCC FIRSTOCC ERROR MENU BACKORD UPDATE .
 ** MK5FU02 TYPE 1 THESE TEMPORARY FIELDS ARE UNREFERENCED - SCRNSLCT NOSCREEN SCREENNO .

VISION:Transact

APPLICATION

DESIGN

MANUAL

Figure B-1 Order/Item Application Code Design Manual Listings (Page 15 of 24)

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE GENERATED : DEC 04, 2003
TARGET ENVIRONMENT : CICS TS
PREPARED BY : _____

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 1

*
* VISION:Transact APPLICATION DESIGN MANUAL *
* TABLE OF CONTENTS *
*

SECTION

- 1 RUN CONTROL LISTING (ATTACH LISTING OF RC, RP, AND RF STATEMENTS)
- 2 SCREEN CONTROL LISTING (ATTACH LISTING OF SC STATEMENTS)
- 3 SCREEN-TO-SCREEN FLOW SUMMARY
- 4 DATABASE USAGE SUMMARY
- 5 FIELD REFERENCE INDEX
- 6 REPORTS FOR SCREEN : CUSTUPD4
 - 6.1 SCREEN GLOSSARY (ATTACH SCREEN GLOSSARY FOR CUSTUPD4)
 - 6.2 SCREEN REQUESTS (ATTACH REQUEST LISTINGS FOR CUSTUPD4)
 - 6.3 SCREEN SYNCHRONIZED DATABASE SUMMARY
 - 6.4 DETAILED CROSS REFERENCE
- 7 REPORTS FOR SCREEN : ORDUPDT4
 - 7.1 SCREEN GLOSSARY (ATTACH SCREEN GLOSSARY FOR ORDUPDT4)
 - 7.2 SCREEN REQUESTS (ATTACH REQUEST LISTINGS FOR ORDUPDT4)
 - 7.3 SCREEN SYNCHRONIZED DATABASE SUMMARY
 - 7.4 DETAILED CROSS REFERENCE
- 8 REPORTS FOR SCREEN : ORDADD4
 - 8.1 SCREEN GLOSSARY (ATTACH SCREEN GLOSSARY FOR ORDADD4)
 - 8.2 SCREEN REQUESTS (ATTACH REQUEST LISTINGS FOR ORDADD4)
 - 8.3 SCREEN SYNCHRONIZED DATABASE SUMMARY
 - 8.4 DETAILED CROSS REFERENCE
- 9 REPORTS FOR SCREEN : ORDMENU4
 - 9.1 SCREEN GLOSSARY (ATTACH SCREEN GLOSSARY FOR ORDMENU4)

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 2

Figure B-1 Order/Item Application Code Design Manual Listings (Page 16 of 24)

** * * * * *
 *
 * VISION:Transact APPLICATION DESIGN MANUAL *
 * TABLE OF CONTENTS *
 * CONTINUED *
 ** * * * * *

SECTION

- 9.2 SCREEN REQUESTS (ATTACH REQUEST LISTINGS FOR ORDMENU4)
- 9.3 DETAILED CROSS REFERENCE

PROGRAM NAME : M501
 TRANSACTION NAME : M501
 DATE : DEC 04, 2003

PAGE 3

* * * * *
 *
 * SCREEN-TO-SCREEN FLOW SUMMARY *
 *
 * * * * *

O| C O O O
 U| U R R R
 T| S D D D
 P| T U A M
 U| U P D E
 T| P D D N
 | D T 4 U
 INPUT | 4 4 . 4

CUSTUPD4| X X X X
 ORDUPDT4| X X - X
 ORDADD4 | X X X X

PROGRAM NAME : M501
 TRANSACTION NAME : M501
 DATE : DEC 04, 2003

PAGE 4

* * * * *
 *
 * DATABASE USAGE SUMMARY *
 *
 * * * * *

DATABASE USAGE : SCREEN SYNCHRONIZED (FILE)
 FILE DEF NAME : CUSTOMER ROOT SEGMENT : CUSTOMER
 FILE TYPE : DL/I DBD NAME : CUSTOMDB

SEGMENTS REFERENCED

LEVEL2	LEVEL3	LEVEL4	LEVEL5	LEVEL6	LEVEL7	LEVEL8	LEVEL9
ORDER		ITEMSHIP					
	ITEMORD						

DATABASE USAGE : SCREEN SYNCHRONIZED (FILE1)
 FILE DEF NAME : ITEMX ROOT SEGMENT : ITEMROOT
 FILE TYPE : DL/I DBD NAME : ITEMDB

NO SEGMENTS REFERENCED BELOW THE ROOT

Figure B-1 Order/Item Application Code Design Manual Listings (Page 17 of 24)

DATABASE USAGE : DYNAMICALLY SYNCHRONIZED (RANGE) (FILE2)
FILE DEF NAME : ITEMX ROOT SEGMENT : ITEMROOT
FILE TYPE : DL/I DED NAME : ITEMDB

DYNAMIC SYNCHRONIZATION FIELD(S) : T,SYNCFD1 AND T,SYNCFD2

NO SEGMENTS REFERENCED BELOW THE ROOT

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 5

*
* SCREEN SYNCHRONIZED DATABASE SUMMARY FOR SCREEN: CUSTUPD4 *
*

*** SCREEN SEGMENT ***			***** DATABASE *****		-- ACTIONS --					
NUMBER	LEVEL	OCCURS	QUAL	SEGMENT NAME	LEVEL	M	U	D	I	X
-01-	1	1	N	CUSTOMER	1	M	U			X
-02-	2	5	N	ORDER	2	M	U			X

WARNING : ANY 'Q' ACTIONS SPECIFIED FOR A SEGMENT HAVE BEEN MIGRATED INTO THE POSSIBLE ACTIONS FOR THAT SEGMENT, AND THEREFORE DO NOT APPEAR IN THIS SYNCHRONIZED DATABASE SUMMARY.

ANY 'V' ACTIONS SPECIFIED FOR A SEGMENT HAVE BEEN RESTATED AS POSSIBLE 'U' AND 'I' ACTIONS.

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 6

*
* DETAILED CROSS REFERENCE FOR SCREEN: CUSTUPD4 *
*

FIELD	REQUEST	REQUEST	REQUEST
QUAL NAME	T-NAME	T-NAME	T-NAME
-----	-----	-----	-----
F ATINID	**INIVAL S-ONEXCUPD	S-VALDCUPD	S-FINDX
F MSGLINE	**OUTPUT		
N CUSTNAME	**INPROC Y	**OUTPUT	
N CUSINO	**OUTPUT		
N CUSTPH	**INPROC Y	**OUTPUT	
N ORDCMPLT	**INPROC Y	**OUTPUT	
N ORDDUDAT	**INPROC Y	**OUTPUT	
N ORDERNO	**OUTPUT		
N ORDINVEN	**INPROC Y	**OUTPUT	
N ORDPONUM	**INPROC Y	**OUTPUT	
N ORDRDATE	**INPROC Y	**OUTPUT	
N ORRPERSON	**INPROC Y	**OUTPUT	
S ACTION	**INIVAL	S-FINDX	

Figure B-1 Order/Item Application Code Design Manual Listings (Page 18 of 24)

S CUSTNAME **INPROC
S CUSTNO **INPVAL
S CUSTPH **INPROC
S ORDCMPLT **INPROC
S ORDDUAT **INPROC
S ORDERNO **INPVAL
S ORDINCN **INPROC
S ORDPNUM **INPROC
S ORDRDATE **INPROC

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 7

*
* DETAILED CROSS REFERENCE FOR SCREEN: CUSTUPD4 *
* CONTINUED *

FIELD	REQUEST		REQUEST	REQUEST
QUAL NAME	T-NAME	MOD?	T-NAME	T-NAME
S ORPERSON	**INPROC			
T COUNTX	S-FINIX	Y	S-ONEXCUPD	

SYMBOL LEGEND:

'Y' - (UNDER MOD?) INDICATES A FIELD WAS MODIFIED IN THE REQUEST IMMEDIATELY TO THE LEFT OF THE 'Y'
**INPVAL - INDICATES THE PROCESSING BETWEEN REQUEST TYPES 1 AND 2
**INPROC - INDICATES THE PROCESSING BETWEEN REQUEST TYPES 2 AND M
**OUTPUT - INDICATES THE PROCESSING REQUIRED TO MOVE DATA TO THE OUTPUT SCREEN

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 8

*
* SCREEN SYNCHRONIZED DATABASE SUMMARY FOR SCREEN: ORDUPDT4 *
*

*** SCREEN SEGMENT ***	***** DATABASE *****		-- ACTIONS --							
NUMBER	LEVEL	OCCURS	QUAL	SEGMENT NAME	LEVEL	M	U	D	I	X
-01-	1	1	N	CUSTOMER	1	M				X
			N	ORDER	2	M				X
-02-	2	15	N	ITEMORD	3		U		I	X
			1	ITEMROOT	1		U			X

WARNING : ANY 'Q' ACTIONS SPECIFIED FOR A SEGMENT HAVE BEEN MIGRATED INTO THE POSSIBLE ACTIONS FOR THAT SEGMENT, AND THEREFORE DO NOT APPEAR IN THIS SYNCHRONIZED DATABASE SUMMARY.

ANY 'V' ACTIONS SPECIFIED FOR A SEGMENT HAVE BEEN RESTATED AS POSSIBLE 'U' AND 'I' ACTIONS.

Figure B-1 Order/Item Application Code Design Manual Listings (Page 19 of 24)

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 9

*
* DETAILED CROSS REFERENCE FOR SCREEN: ORDUPDT4 *
*

FIELD QUAL NAME	REQUEST T-NAME	MOD?	REQUEST T-NAME	MOD?	REQUEST T-NAME	MOD?
F ATINID	1-OUR1N1 S-OUR2N1		**INPVAL S-OUR2N2		O-SYNCLDAD	
F ERROR	S-OUR2N1	Y	S-OUR2N2	Y		
F MSGLINE	**OUTPUT					
N CUSTNAME	**OUTPUT					
N CUSTNO	**OUTPUT					
N ITEMORD	O-SYNCLDAD S-OUR2N2		**OUTPUT		S-OUR2N1	
N ITEMSHIP	S-OUR2N1					
N ITMPRICE	**INPROC	Y	**OUTPUT			
N ITMTYOR	**INPROC S-OUR2N2	Y	**OUTPUT		S-OUR2N1	
N ORDERNO	**OUTPUT					
N QTYEKORD	**INPROC	Y	**OUTPUT			
S ACTION	1-OUR1N1 S-OUR2N2	Y	**INPVAL		S-OUR2N1	
S CUSTNO	**INPVAL					
S ITEMORD	**INPVAL					
S ITMPRICE	**INPROC		S-OUR2N1	Y		
S ITMTYOR	**INPROC		S-OUR2N1		S-OUR2N2	
S ORDERNO	**INPVAL					
S QTYEKORD	**INPROC					
T MSG	S-OUR2N1	Y	S-OUR2N2	Y		
T QANONHND	S-OUR2N2	Y				

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 10

*
* DETAILED CROSS REFERENCE FOR SCREEN: ORDUPDT4 *
* CONTINUED *

FIELD QUAL NAME	REQUEST T-NAME	MOD?	REQUEST T-NAME	MOD?	REQUEST T-NAME	MOD?
T SYNCFLD1	O-SYNCLDAD	Y				
T SYNCFLD2	O-SYNCLDAD	Y				
1 ITMPRICE	S-OUR2N1					

Figure B-1 Order/Item Application Code Design Manual Listings (Page 20 of 24)

```
1 QANONHND S-OUR2N2 Y
2 ITEMNAME **OUTPUT
2 QANONHND **OUTPUT
```

SYMBOL LEGEND:

'Y' - (UNDER MOD?) INDICATES A FIELD WAS MODIFIED IN THE REQUEST
IMMEDIATELY TO THE LEFT OF THE 'Y'
**INPVAL - INDICATES THE PROCESSING BETWEEN REQUEST TYPES 1 AND 2
**INPROC - INDICATES THE PROCESSING BETWEEN REQUEST TYPES 2 AND M
**OUTPUT - INDICATES THE PROCESSING REQUIRED TO MOVE DATA TO THE
OUTPUT SCREEN

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 11

```
* * * * *
*
* SCREEN SYNCHRONIZED DATABASE SUMMARY FOR SCREEN: ORDADD4 *
*
* * * * *
```

*** SCREEN SEGMENT ***			***** DATABASE *****		-- ACTIONS --					
NUMBER	LEVEL	OCCURS	QUAL	NAME	LEVEL	M	U	D	I	X
-01-	1	1	N	CUSTOMER	1	M				X
			N	ORDER	2	M	U		I	X
-02-	2	11	N	ITEMORD	3				I	X
			1	ITEMROOT	1		U			X

WARNING : ANY 'Q' ACTIONS SPECIFIED FOR A SEGMENT HAVE BEEN MIGRATED
INTO THE POSSIBLE ACTIONS FOR THAT SEGMENT, AND THEREFORE
DO NOT APPEAR IN THIS SYNCHRONIZED DATABASE SUMMARY.

ANY 'V' ACTIONS SPECIFIED FOR A SEGMENT HAVE BEEN RESTATED
AS POSSIBLE 'U' AND 'I' ACTIONS.

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 12

```
* * * * *
*
* DETAILED CROSS REFERENCE FOR SCREEN: ORDADD4 *
*
* * * * *
```

FIELD	REQUEST	REQUEST	REQUEST
QUAL NAME	T-NAME	T-NAME	T-NAME
-----	-----	-----	-----
F ATINID	1-VLDORDAD	**INPVAL	
F MSGLINE	**OUTPUT		
N CUSTNAME	**OUTPUT		
N CUSINO	**OUTPUT		
N CUSIPH	**OUTPUT		
N ITMPRICE	**INPROC Y		
N ITMOTYOR	**INPROC Y		
N ORDDUDAT	**INPROC Y	**OUTPUT	
N ORDERNO	**OUTPUT		

Figure B-1 Order/Item Application Code Design Manual Listings (Page 21 of 24)


```
N  ORDPNUM  **INPROC  Y      **OUTPUT
N  ORDRDATE  **INPROC  Y      **OUTPUT
N  ORPERSON  **INPROC  Y      **OUTPUT
N  QTYEKORD  **INPROC  Y
S  CUSTNO    **INVAL
S  ITEMORD   **INVAL
S  ITMPRICE  2-QAR2N1  Y      **INPROC
S  ITMQTYOR  **INVAL      **INPROC
S  ORDDUDAT  **INPROC
S  ORDERNO   **INVAL
S  ORDPNUM   **INPROC
S  ORDRDATE  **INPROC
S  ORPERSON  **INPROC
```

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 13

```
* * * * *
*
* DETAILED CROSS REFERENCE FOR SCREEN: ORDADD4 *
*          CONTINUED          *
* * * * *
```

FIELD QUAL NAME	REQUEST T-NAME	MOD?	REQUEST T-NAME	MOD?	REQUEST T-NAME	MOD?
S QTYEKORD	**INPROC					
1 ITMPRICE	2-QAR2N1					
1 QANONHND	**INVAL		**INPROC	Y		

SYMBOL LEGEND:

'Y' - (UNDER MOD?) INDICATES A FIELD WAS MODIFIED IN THE REQUEST
IMMEDIATELY TO THE LEFT OF THE 'Y'
**INVAL - INDICATES THE PROCESSING BETWEEN REQUEST TYPES 1 AND 2
**INPROC - INDICATES THE PROCESSING BETWEEN REQUEST TYPES 2 AND M
**OUTPUT - INDICATES THE PROCESSING REQUIRED TO MOVE DATA TO THE
OUTPUT SCREEN

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 14

```
* * * * *
*
* DETAILED CROSS REFERENCE FOR SCREEN: ORDMENU4 *
*          *
* * * * *
```

FIELD QUAL NAME	REQUEST T-NAME	MOD?	REQUEST T-NAME	MOD?	REQUEST T-NAME	MOD?
F MSGLINE	**OUTPUT					

SYMBOL LEGEND:

'Y' - (UNDER MOD?) INDICATES A FIELD WAS MODIFIED IN THE REQUEST
IMMEDIATELY TO THE LEFT OF THE 'Y'

Figure B-1 Order/Item Application Code Design Manual Listings (Page 22 of 24)

**INVAL - INDICATES THE PROCESSING BETWEEN REQUEST TYPES 1 AND 2
**INPROC - INDICATES THE PROCESSING BETWEEN REQUEST TYPES 2 AND M
**OUTPUT - INDICATES THE PROCESSING REQUIRED TO MOVE DATA TO THE
OUTPUT SCREEN

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 15

* * * * *
*
* FIELD REFERENCE INDEX *
*
* * * * *

FIELD QUAL NAME	SCREEN CONTROL	SCREEN NAME	MOD?	SCREEN NAME	MOD?	SCREEN NAME	MOD?
F ATINID	YES	CUSTUPD4		ORDUPDT4		ORDADD4	
F ERROR		ORDUPDT4	R				
F MSGLINE		CUSTUPD4 ORDMENU4		ORDUPDT4		ORDADD4	
N CUSTNAME		CUSTUPD4	S	ORDUPDT4		ORDADD4	
N CUSINO		CUSTUPD4		ORDUPDT4		ORDADD4	
N CUSTPH		CUSTUPD4	S	ORDADD4			
N ITEMORD		ORDUPDT4					
N ITEMSHIP		ORDUPDT4					
N ITMPRICE		ORDUPDT4	S	ORDADD4	S		
N ITMQTYOR		ORDUPDT4	S	ORDADD4	S		
N ORDCOMPLT		CUSTUPD4	S				
N ORDDUAT		CUSTUPD4	S	ORDADD4	S		
N ORDENO		CUSTUPD4		ORDUPDT4		ORDADD4	
N ORDINVEN		CUSTUPD4	S				
N ORDPONUM		CUSTUPD4	S	ORDADD4	S		
N ORORDATE		CUSTUPD4	S	ORDADD4	S		
N ORPERSON		CUSTUPD4	S	ORDADD4	S		
N QTYEKORD		ORDUPDT4	S	ORDADD4	S		
T COUNIX		CUSTUPD4	R				
T MSG		ORDUPDT4	R				
T QANONHND		ORDUPDT4	R				
T SYNCFLD1		ORDUPDT4	R				

PROGRAM NAME : M501
TRANSACTION NAME : M501
DATE : DEC 04, 2003

PAGE 16

* * * * *
*
* FIELD REFERENCE INDEX *
* CONTINUED *
* * * * *

FIELD QUAL NAME	SCREEN CONTROL	SCREEN NAME	MOD?	SCREEN NAME	MOD?	SCREEN NAME	MOD?

Figure B-1 Order/Item Application Code Design Manual Listings (Page 23 of 24)

```

T  SYNCFLD2      ORDUPDT4  R
1  ITMPRICE      ORDUPDT4      ORDADD4
1  QANONHND      ORDUPDT4  R  ORDADD4  S
2  ITEMNAME      ORDUPDT4
2  QANONHND      ORDUPDT4

```

SYMBOL LEGEND (UNDER MOD?) :

'S' - MODIFIED BY FIELD ACTIONS FOR THE SCREEN
'R' - MODIFIED IN AT LEAST ONE USER REQUEST FOR THE SCREEN
'B' - MODIFIED BY BOTH FIELD ACTIONS AND AT LEAST ONE USER
REQUEST FOR THE SCREEN

```

** MK5W204  TYPE 0  NUMBER OF MESSAGES PRINTED IS      31.
                    M5INPUT  -      194 RECORDS INPUT
                    MSPUNCH  -     1048 RECORDS OUTPUT
                    MSMAP    -         0 RECORDS OUTPUT
                    MSTPTHEL -        15 RECORDS OUTPUT
                    MSPREP    -         0 RECORDS OUTPUT
                    M5COMPAR  -         97 RECORDS OUTPUT
                    M5LIST    -     1403 RECORDS OUTPUT
                    75 TRACKS ASSIGNED TO M5LIB - 23 TRACKS NOT FULL. THE LIBRARY DIRECTORY BLOCKING FACTOR IS 800.
** MK5W209  TYPE 0  17928 BYTES OF TYPE A STORAGE USED DURING APPLICATION GENERATION

```

Figure B-1 Order/Item Application Code Design Manual Listings (Page 24 of 24)

VISION:Transact Interaction Cycle Diagram

[Figure C-1](#) contains a comprehensive diagram of the VISION:Transact interaction cycle. This diagram is useful for developing and testing VISION:Transact generated applications.

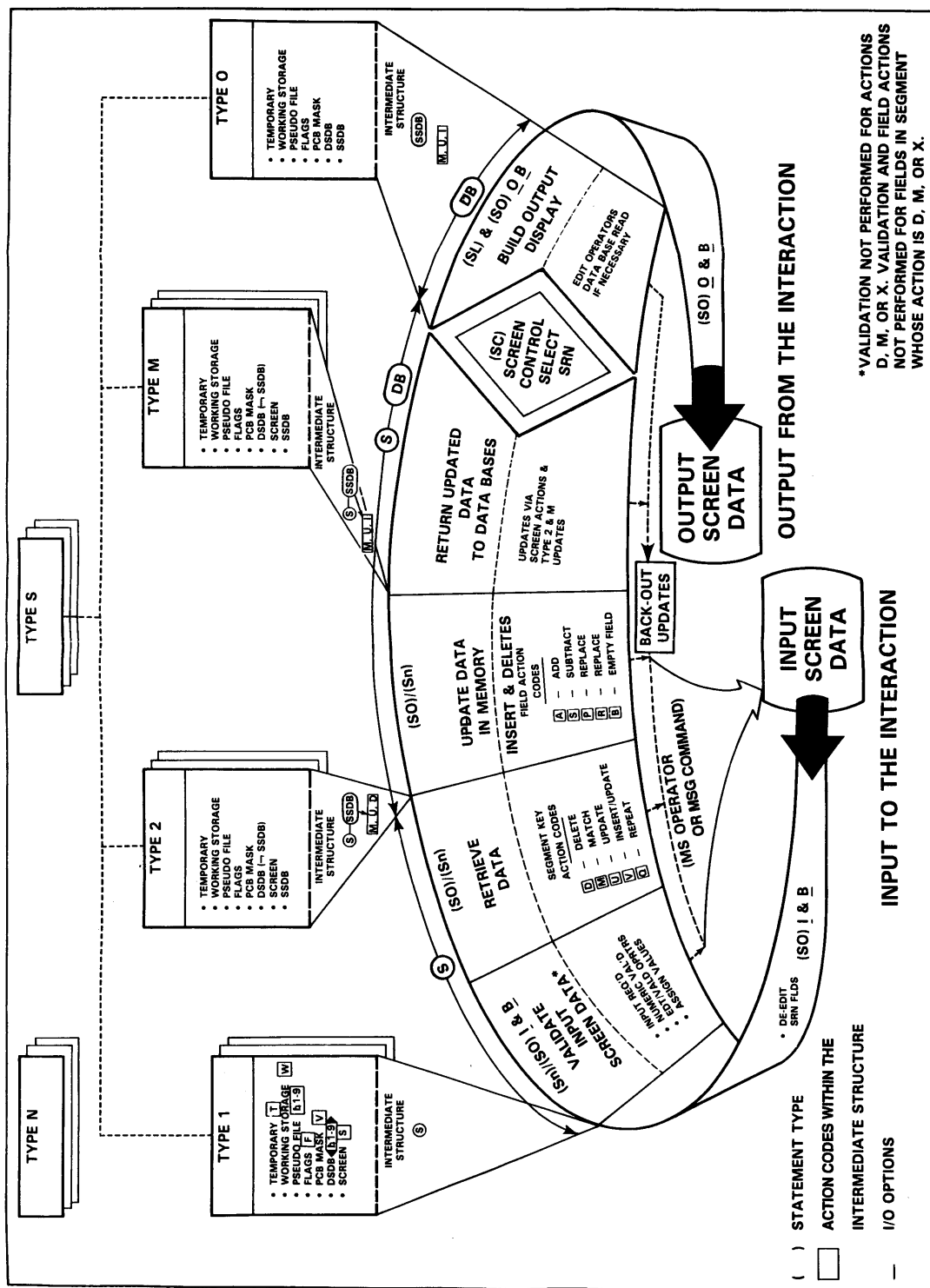


Figure C-1 VISION:Transact Interaction Cycle

Glossary

The following is a glossary of terms that have special meaning in the context of VISION:Transact documentation. Some of these terms are unique to VISION:Transact; others have special, unique meanings when used in reference to concepts related to VISION:Transact. They are presented here as an aid to your understanding of the more complete discussion of the concepts presented earlier in this guide. For a discussion of the use and coding of specific VISION:Transact source statements, see the *Advantage VISION:Transact Reference Summary* or the *Advantage VISION:Transact ASL Reference Guide*.

Action Code

The entry on a screen processing definition that specifies the kind of data manipulation that takes place between a screen field and its associated destination field or segment.

Alias

The name entered on the L0 statement is used to interface a DBMS file. DBMS requires unique field names within segments not the entire file. To accommodate the VISION:Transact requirement for unique field names across the entire file, the VISION:Transact field name is entered in the FIELD NAME specification on the L0 statement and the DBMS field name is entered under the ALIAS.

APPGEN

See Application Generation.

Application Generation

The process of compiling an application with VISION:Transact and generating an object module ready for link edit.

Branching

A technique of altering the normal sequence of request operations from one PR statement to the next. Branch operators are GO, NS, and GS.

Browse

Synchronization

A form of dynamic synchronization in which the object is to retrieve multiple records from a dynamically synchronized data base (DSDB) when reference is made to a field from that data base. The records are retrieved based on the value of one (in a generic browse) or two (in a range browse) synchronizing fields.

Call Option

Entered on a PR statement in requests or CALL command in procedures. The Call option allows you to procedurally call an external routine from a VISION:Transact generated application. Calls may be issued to user-written routines, DL/I data base system modules, or to subroutines written in PL/I. In procedures, calls can also be issued to VISION:Transact subroutines.

Catalog Retrieval

Catalog retrieval (CR) statements retrieve items from the library. Catalog retrieval statements are processed according to the run type specified on the RC statement.

COMMAREA

For CICS TS programs, a storage area is used to pass input data to a program and receive output data from a program.

Composite Segment

An occurrence of a segment of the intermediate structure, made up of screen and/or (depending on the timing in the interaction) data base fields.

Content/Action Specifications

Specifications that name the source or destination, segment or field actions, etc., of screen fields used in a screen processing definition.

Concatenated Segment

See composite segment.

Data Base

A generic term, as used in this guide, referring to any external file (other than an IMS scratch pad area) requiring I/O operations to be accessed from an application; contrast with IMS (or DL/I) data base, a term used here to refer exclusively to files accessible through Data Language/I.

DB2 Relational Data Base

A term used in this guide to apply to a data base accessed through SQL. This term applies to both SQL/DS data bases used in DOS/VSE and DB2 data bases used in MVS.

De-edit/Edit Pattern

Using the ED or EN operators on Sn statements allows you to specify an edit pattern or grouping characters for screen fields. The ED operator inserts characters in the screen field on output. If the screen field is also input to the application, the edit pattern defines the de-editing done on input. The EN operator inserts grouping characters to numeric source fields prior to output. If a non-blank screen field is input, the EN right-justifies the field and fills it with leading zeroes.

Definition Run

The process of cataloging file definitions and/or screen processing definitions on a VISION:Transact library. When a screen processing definition is successfully defined to VISION:Transact, the output also includes MFS or BMS source input statements.

Destination Field or Segment

A data field or segment to which screen input is directed, via field or segment action specifications in a screen processing definition.

Direct Synchronization

A type of dynamic synchronization in which a single record is retrieved from a referenced dynamically synchronized data base (DSDB). The record retrieval is based on the value of a single synchronizing field at the time of the reference.

DSDB

See dynamically synchronized data base.

Dynamic Selection of Action Codes

A condition coded into a VISION:Transact generated application in which a segment or field action is selected when the application is executed online. The IF operator on an Sn statement enables the execution-time selection of alternative actions. The selection is based on control field values determined by the application or entered on the screen by the terminal user.

Dynamic Synchronization

A type of read-only access to a data base that is based on a root key value (or a portion thereof) which is placed in a synchronizing field or fields specified on an RF statement; contrast with screen synchronization. Unlike screen synchronization, the retrieval of segments from data base accessed through dynamic synchronization is dependent on the timing of reference to fields in the data base. See also Browse Synchronization and Direct Synchronization.

Dynamically Synchronized Data Base (DSDB)

A data base which is accessed through dynamic synchronization.

Extended Attribute

This entry on a screen layout statement specifies the extended attributes (color, extended highlighting and programmed symbols) assigned to the text entries on some 3270 terminals.

File Definition

File definition statements (FD, LS, L0, Ln, and LR) are used to define the data structure and characteristics of the files used by an application.

Flags

These are internal indicators which you may interrogate or set to indicate the existence of certain conditions. Flag fields are identified to the application by the "F" qualifier.

Generic Browse

A type of browse synchronization in which records are retrieved using the leading (high order) portion of the key/search field; contrast with range browse.

Glossary

The formatted listing produced when a data structure or screen is defined to VISION:Transact.

IF Operator

See dynamic selection of action codes.

Implicit Definition (of a temporary field)

The assignment of characteristics to a temporary field when that temporary field is only referenced in a PR, procedure, SC, or S0/Sn statement, rather than being explicitly defined in a TF statement (requests) or FIELD command (procedures).

Implicit Reference

A reference to a parent segment which is forced by an explicit reference to one of its children in a screen request or procedure or a screen processing definition.

IMS data base

A term used in this guide to apply to data bases accessed through Data Language/I. This term applies both to the DL/I data bases used in DOS/VSE and IMS data bases used in MVS.

Input Screen

The screen display that is at the beginning of the interaction under discussion.

Input Screen Content

The screen fields that pass data from the input screen to the interaction. These fields are distinguished by their having an I/O specification of I or B in the screen processing definition for the input screen.

Interaction

The link between two consecutive screen displays; it includes the processing of the input screen, data base manipulation, selection of the output screen, and the building of the output screen display.

Intermediate Structure

The set of screen and/or screen synchronized data base segments available to the application during the interaction, the exact contents being determined by the timing of interaction functions and request or procedure references to data base fields.

Invalid Field

A field that has been marked by your VISION:Transact generated application as a result of an attempt to place invalid data into the field. References that can cause a field to be set invalid include: attempting to move an invalid or missing field into it, making it the result of a division by zero, and causing the field to overflow in an arithmetic operation.

Item Tracking

Used in a definition run or application generation run, an IT statement allows you to maintain and report creation and expiration dates and times, and other information for items cataloged in the library.

Library Utilities

The utility program (M5UTIL) is used to create and maintain VISION:Transact libraries. The functions of the library utilities are accomplished using utility control (UC) statements. Multiple UC statements may be included in a single utility run.

Linkage Section

See PCB mask.

Load Module

A term used in this guide to refer to the contents of both MVS load libraries and DOS/VSE core image libraries.

Looping

The automatic processing of multiple occurrences of recurring segments by your VISION:Transact generated application.

Missing Field or Segment

A data base field or segment that is unavailable to process even though a slot exists for it in the intermediate structure, or dynamic synchronization has been attempted to make it available.

Multi-path Nested Looping

Concurrent looping on segments in different paths of a data base or the intermediate structure.

Multiple Occurrence Management

See looping.

Nested Looping

Concurrent looping on more than one recurring segment.

Null Segment Processing

This entry on the SD statement controls the processing of null (empty) input screen segment occurrences.

Output Control Operators

Specified on screen control statements, output control operators are used to control output interaction functions, to select a type O request or procedure for an output screen or to select type 1, 2, or M requests or procedures when processing an input screen.

Output Screen

The screen display that is at the end of the interaction under discussion.

Output Screen Content

The fields that pass data from the application to the output screen. These fields are distinguished by having an I/O specification of O or B in the screen processing definition for the output screen.
Partial Fielding

To limit processing of a character type field by specifying a start character and a specific number of characters within that field.

Pattern

A pattern is used to describe the contents of the character type field named in a relational expression. A pattern may be specified on S0, SN, SC, PR, and procedure statements. Identified by the P qualifier, the pattern may be used as the B Operand or field of one of the relational operators EQ or NE.

PCB Mask

A file definition consisting of the PCBs used by an application. The PCB data is maintained for the duration of the interaction and may not be modified by the application program.

Procedure

A module of statements in an APPGEN run that is used for application specific processing beyond the automatic functions of any VISION:Transact generated application and those of the screen processing definition. Procedures are written using the free-form language ASL.

Pseudo Files

A kind of structured working storage used to simulate a structured data base record in storage. They can be saved between interactions in a SPA for IMS TM or in Temporary Storage for CICS.

Range Browse

A type of browse synchronization in which the keys of retrieved records are greater than or equal to the value in one synchronizing field (synchronizing field 1 on the RF statement) and less than or equal to the value in another synchronizing field (synchronizing field 2 on the RF statement); contrast with generic browse.

Re-initialize

A specification on a screen request or procedure ER statement that concerns temporary fields defined in the TF statements of that request or FIELD commands of that procedure. If a TF statement or FIELD command explicitly provides an initial value, the temporary field is initialized to that value. For other temporary fields defined in that request or procedure, numeric temporary fields are zeroed, and character temporary fields are blanked.

Relational Operator

An operator in a screen processing definition, procedure, or request that compares one value to another for validation or data selection.

Request

A module of statements in an APPGEN that is used for application-specific processing beyond the automatic functions of any VISION:Transact generated application and those of the screen processing definition. A request is written using fixed form statements for procedural code (PR), temporary field definition (TF), and comments (AA).

Run Control Group

The run control group identifies the run type, teleprocessing monitor, run files, run parameters, or item tracking statements required for the application. The run type entered on the RC statement controls how additional statement types included in the run control group are processed (i.e., RF, RP, IT).

Run Type

The run type entry on the RC statement specifies to VISION:Transact that the run is either a definition run, application generation run or source statement retrieval run.

Screen Control

(1) The statements in an APPGEN that select the output screen of an interaction, and (2) The process of the application selecting the output screen of an interaction.

Screen Field

A field that contains data that is passed to the interaction during the processing of an input screen.

Screen Matching (also, Screen Matched)

See screen synchronization.

Screen Procedure

A procedure capable of performing data selection and manipulation that is associated with the processing of a particular screen, and whose execution is determined by its type (see type 1, type 2, type M, and type O request/procedure).

Screen Processing Definition (also, screen definition)

Statements that describe the layout and structure of a screen display used in a VISION:Transact generated application, and provide the basis for the validation and processing of that display.

Screen Request

A request capable of performing data selection and manipulation that is associated with the processing of a particular screen, and whose execution is determined by its type (see type 1, type 2, type M, and type O request/procedure).

Screen Synchronization (also, screen synchronized)

The process of retrieving data base segments and associating them with specific screen segments for processing by an interaction; this process is performed automatically by a VISION:Transact generated application. The basis for screen synchronization is the segment actions specified in a screen processing definition.

Screen Synchronized Data Base (SSDB)

A data base accessed through the use of screen synchronization.

Segment Operator

Specified on screen control statements, the SG (segment operator) is used to supply the segment function indicated in Operand B to the segment named by Operand A. For example, the SG operator used with the SELECT ALL function releases the segment (other than a root segment) named in Operand A and all its children from the intermediate structure.

Source Field or Segment

A data field or segment from which screen output is derived, via an I/O specification of O or B in a screen processing definition.

Source Statement Retrieval Run

A source statement retrieval (SSR) run is used to retrieve items from a VISION:Transact library and output the items in source statement form to the M5SSOUT data set.

Split Screen Segment

Specifying an A in the V/S/A entry on the S0 statement allows you to display data from the parent segment of the screen after all its dependent segments. SSDB
See screen synchronized data base.

Structured (Hierarchical) Screen

A screen display consisting of one root segment and up to eight possibly recurring subordinate segments, analogous in structure to a single-leg IMS data base record.

Subroutine Request/Procedure

See type S request/procedure.

Synchronizing Field

A field that provides a key value used to retrieve a record or records from a dynamically synchronized data base.

Temporary Field

A field (sometimes the source or destination for a screen field) that exists only in memory and not on any external file. It is referenced with a T qualifier in a request or procedure, screen processing definition, or screen control statement. Contrast with working storage.

Type M Request/Procedure (also, Type M time)

A screen request or procedure that is executed after the in-memory update of screen synchronized data bases, but before the physical update of the data bases themselves.

Type O Request/Procedure (also, Type O Time)

A screen request or procedure that is executed during the building of the output screen of an interaction.

Type S Request/Procedure

A request or procedure not associated with a particular screen. This subroutine request/procedure is invoked only by another request or procedure, to which control returns following type S request execution.

Type 1 Request/Procedure (also, Type 1 Time)

A screen request or procedure that is executed after an input screen is input, but before any other interaction processing takes place.

Type 2 Request/Procedure (also, Type 2 Time)

A screen request or procedure that is executed after the screen processing definition-based validation of screen fields and screen synchronization take place, but before the in-memory update of any destination fields.

Unstructured (Flat) Screen

A screen display consisting of only one segment.

Virtual Field

A screen field that is not displayed or laid-out on a screen processing definition; it is used to manipulate or validate data during the processing of an input screen by the interaction.

Virtual Key

A key defined for a root data base segment, which does not physically reside in that segment. It is used to provide the VISION:Transact generated application with information with which to build segment search arguments (SSA) for that data base.

**Virtual/
Same/After**

For a non-virtual screen field, this entry relates an S0 statement to a series of field designators (on one or more SL statements). This field is displayed on the screen. S0 statements specifying virtual screen fields are not related to the field attributes on the SL statements. A V entry tells VISION:Transact not to assign this S0 statement to any displayed screen field.

**VISION:Transact
Executor**

Component of the VISION:Transact system that operates as a collection of library subroutines that participate in the link edit and execution of the generated application. To the target communications monitor, the executing modules are just programs that use the standard facilities of that online environment.

**VISION:Transact
Generator**

Component of VISION:Transact system that you use to build an application. Runs using the generator produce control statements and an object module to be used in the installation of the application.

Working Storage

Fields defined via a VISION:Transact file definition that exist only in memory and only during an interaction, unless they are made SPA-resident in an IMS conversational application or stored in temporary storage during a CICS application. They are referenced with a W qualifier in a request, screen processing definition, or screen control statement. Contrast with temporary field.

Index

Symbols

* operator, 7-4
+ operator, 7-4
- operator, 7-4
/ operator, 7-4
?C operator, 7-5, 12-3
?T statement, 1-13, 12-1, 12-2, 12-3
?X operator, 7-5, 12-3

A

A action, 4-8
AA statement, 1-17, 7-1
ACB, 1-6, 1-7, 1-8, 1-9
Accessibility, 1-13
Action code
 Action, 4-7
 Actions, 4-6
Actions, 4-6, 4-7, 4-8, 4-9, 4-16, 4-18, 4-28, 4-30, 6-10, 6-15, 9-4
Add, 7-9
Add action, 4-8
After screen field, 6-5
ALIAS entry, 10-4
Alphabetic field
 Character field, 9-1
Alphanumeric field
 Character field, 9-1
Alternate index path, 10-17
Alternate index view, 10-17
Alternate response PCB, 10-12
AND, 7-7
APPGEN, 1-3, 1-4, 1-5, 1-6, 1-7, 1-8, 1-10, 1-13, 1-14, 1-16, 1-17, 4-4, 4-6, 4-23, 10-19, 11-4, 11-5, 12-1, 12-2
APPLCTN, 1-7
Application
 sample of, 2-1
Application components, 1-10, 1-11
Application Control Block
 ACB, 1-6
Application development, 1-1, 1-3, 1-7, 1-8
Application flow, 1-11, 1-13, 3-2, 4-14, 4-22, 4-26, 5-23, 11-1
Application Flow Trace, 12-1
Application generation
 APPGEN, 1-3
Application linking, 4-26, 5-2
Application load module, 1-5
Application name, 1-14
Application run file, 1-14, 6-20
Application termination, 4-25, 12-2
Applications, passing control, 4-23, 4-28
Applicationsviewofdata', 6-13

Arithmetic expressions, 7-9, 9-4
Arithmetic operations, 7-9, 9-1, 9-3, 9-4
Arithmetic operators, 7-4
Ascending order, 10-2, 10-19
ASL commands, 7-3, 7-5
Assembly, 1-6, 1-8
Assignment of screen field names, 9-8
Assignment of screen fields, 5-4
ATTNID flag, 7-20
Attribute byte, 5-4
Attributes, 5-6, 5-7, 5-13, 5-23
Automatic Format, 5-2, 5-11
Automatic validation, 7-10
Availability of data, 7-53

B

B action, 4-8
Back branch, 7-8
Back Branch Limit, 7-30
Back branch limit, 7-14
Back branching, 7-30
Backout updates, 4-4, 4-5, 7-10, 7-33
Backward branch, 7-8, 7-14
Base cluster, 10-17
Basic Mapping Support
 BMS, 1-8
Batch Terminal Simulator
 BTS, 12-3
Binary fields, 9-1, 9-2, 9-3
Bind, 1-4, 1-7, 1-9
Blinking, 5-7
BMS, 1-8, 1-16, 5-16
BMS mapset, 1-8
BMS mapset/map, 5-2
Boolean logical operators, 7-7

Boolean notation, 7-7
Both file, 8-5
Both input and output
 Both screen fields, 4-20
Both screen fields, 4-6, 4-19, 4-20
Branch backwards, 7-8
Branch locations, 7-29, 7-31
Branching locations, 7-6, 7-8
Branching operators, 7-4, 7-28, 7-31
Browsing, 2-2, 6-1, 6-19, 8-1, 8-2, 8-3, 8-6, 8-7, 8-10, 8-13, 8-14
BTS, 12-3
BTSOUT data set, 12-3
Buffer size, 10-2, 10-4
Bypass display of segment occurrence, 7-26

C

Calculation for output, 7-25
Calculations, 7-3
CALL command, 7-6, 7-9
CALL PROCEDURE command, 7-16, 7-17, 7-28
CALL statement, 7-8, 7-9
CASE command, 7-6, 7-9
Catalog maintenance
 Definition run, 1-9
Character field, 7-10, 9-1, 9-3
Character fields, 9-4, 9-8
CICS conversational processing, 11-2
CICS Temporary Storage, 10-19
CICS temporary storage, 10-18
CICS transaction initiation, 11-2
CICS/VS executor, 1-8, 1-9
Cluster, 10-15
Color, 5-7
COMLIB, 1-3

Comments, 1-17, 7-2
Comparison operations, 9-1, 9-3
Comparison operators, 7-4, 9-10
Components of a procedure, 7-15
Components of a request, 7-12
Components of an application, 1-10, 1-11
Composite segment, 6-15, 6-18, 6-19, 7-45, 7-49, 7-52
Concatenating screen fields, 7-42
Connector, 4-22, 4-23, 4-24, 5-16, 5-17, 7-7
Consistency of segment actions, 4-11
Constant, 5-18
CONTINUE command, 7-16, 7-17
CONTINUE command, 7-6, 7-8, 7-26
Control field, 4-9
CONV, 11-2
Conversational processing, 10-19, 11-2
Conversions, 9-1, 9-3, 9-4, 9-8
Count field, 10-14
Count fields, 10-17, 10-18
CSTATUS flag, 7-9
CT statement, 11-6
Cursor position, 5-6, 5-19
Customer data base, 2-1, 2-2

D

D action, 4-7, 5-18, 7-36
Dark intensity, 5-6
Data accessibility, 1-11, 4-30, 6-7
Data availability, 6-10, 6-12
Data base, 8-1, 8-13, 10-1
Data base backout, 7-33
 Backout updates, 4-4
Data base calls, 10-2, 12-2
Data Base Definition
 DBD, 1-6

Data base path, 7-45
Data base read
 Retrieval, 6-15
Data base segment, 6-19
Data Base Segments, 6-23, 6-24
Data base segments, 6-7, 6-14, 6-18, 6-19, 7-35, 7-39
Data base usage summary
 Design Manual, 1-4
Data bases, 1-15
Data Entry Data Base
 DEDB, 10-10
Data flow, 4-9, 4-14, 4-16, 4-20
Data structures, 1-4, 1-9, 1-15
DB2 data base, 1-15, 9-4, 10-1, 10-12
DB2 Data bases, 1-3
DB2 data bases, 1-4, 1-5, 1-6, 1-7, 1-8, 1-9, 3-1, 10-2
DB2 precompiler, 1-6, 1-7, 1-8, 1-9
DB2 table, 10-2
DBD, 1-6, 1-8, 1-15, 3-1, 10-3, 10-4, 10-5, 10-7, 10-8, 10-10
DBD name, 10-4
DBDNAME, 1-15
De-editing, 5-20, 7-19
De-editing screen field, 5-13
Debug level, 1-14
Debugging, 1-4, 1-13
 Tracing, 12-1
Decimal point, 9-3
Decimal point character, 5-14
Decimal validation, 5-14
DEDB, 10-10, 10-11
Default action, 4-10, 4-11
Default device, 5-2
Default values, 7-22
Default values for screen field, 5-12
Defining, 1-4

- data bases, 1-14
- data structures, 3-2
- DB2 data bases, 1-16, 3-1, 10-2, 10-12
- Fast Path data bases, 10-10
- fast path data bases, 1-16, 10-2
- files, 1-4, 1-14
- IMS data bases, 1-16, 3-1, 10-2, 10-3, 10-4
- ISAM data bases, 1-16, 3-1, 10-2
- PCB masks, 1-14, 1-16, 10-19
- pseudo files, 1-14, 1-16, 10-18
- screen definition, 1-4
- structured records, 10-2, 10-14
- temporary fields, 9-6, 10-19
- VSAM data bases, 3-1, 10-2, 10-16, 10-17
- Working storage, 10-19
- working storage, 1-14, 1-16
- Definition run, 1-3, 1-4, 1-5, 1-6, 1-7, 1-8, 1-9, 1-14, 1-16, 1-17, 5-2, 5-4, 11-6
- Delete action, 4-7, 5-18, 7-36
- Deleted occurrences, 7-36
- Deleted segment occurrences, 7-22, 7-24, 7-35, 7-38
- Descending order, 10-2, 10-19
- Design Manual, 1-4, 1-13
- Destination fields, 4-2, 4-4, 4-7, 4-8, 4-11, 4-13, 4-14, 4-16, 4-19, 4-30, 5-4, 5-11, 5-12, 7-22, 7-42, 9-8
- Destination of output screen, 4-25, 11-1
- Development
 - Application development, 1-7
- Device default, 5-2
- Device group, 5-8
- Device ID, 5-2, 5-8
- Device name, 5-2
- Device Table, 5-2, 5-8
- DF operator, 5-12, 5-20, 7-41
- Diagnostic message
 - Error message, 4-4
- DIF/DOF names, 5-2
- Direct DSDB
 - Direct synchronization, 8-6
- Direct synchronization, 8-1, 8-2, 8-4, 8-5, 8-7, 8-13
- Disable tracing, 12-2
- Display Modified, 5-20
- Display output screen, 7-26
- Display repetitive information, 5-23
- Display segment occurrence, 7-26
- Displaying field contents, 12-3
- Displaying literals, 12-3
- Displaying multiple records, 6-1
- Displaying repetitive information, 6-1
- Divide, 7-9
- DL/I calls, 1-15, 7-9, 10-3, 10-7, 10-19, 12-2
- DL/I data base
 - IMS data base, 10-19
- DLI statement, 7-9
- DO command, 7-6, 7-8, 7-9, 7-43, 7-45
- DO FORALL command, 7-43, 7-45
- DS operator, 5-11
- DSDB, 1-11, 1-13, 4-2, 6-13, 7-35, 7-53, 8-1, 8-2, 8-3, 8-4, 8-5, 8-7, 8-9, 8-12, 8-14, 9-8, 10-16, 10-17, 12-2
- DSDB and SSDB, 8-5
- DSDB I/O call trace, 12-2
- DSDB looping trace, 12-1
- DSDB retrieval trace, 12-1
- Duplicate insert, 5-20
- Dynamic actions
 - Dynamic selection of actions, 4-8
- Dynamic and screen synchronization, 8-5
- Dynamic partial field flags, 7-11
- Dynamic partial fielding, 7-11
- Dynamic selection of actions, 4-8, 4-9, 6-10
- Dynamic selection of segment actions, 7-36
- Dynamic Synchronization, 8-12
- Dynamic synchronization, 2-2, 6-20, 8-1, 8-5, 8-9
- Dynamic transaction codes, 4-28
- Dynamically selected actions, 7-37

Dynamically synchronized, 8-14, 10-17

Dynamically Synchronized Data Base
DSDB, 1-11

E

E connector, 4-22

EBCDIC character, 9-1

EC operator, 5-20

ED operator, 5-9, 5-13

Edit insertion characters, 5-9, 5-13

Edit mask, 5-9

Edit pattern

 Edit mask, 5-9

Editing for output, 7-25

Editing screen fields, 5-9

ELSE command, 7-6

Empty segment, 10-14

EN operator, 5-13

Enable tracing, 12-2

End an application, 4-25

END branch, 7-8, 7-26, 7-27, 7-31

END command, 7-6

Entry Sequenced Data Set
ESDS, 10-16

EQ operator, 7-4

ER statement, 7-2, 7-3, 7-14, 7-28, 7-30, 11-1, 11-4,
11-5

Error conditions, 5-20

Error correction, 2-9

Error cycle, 2-9, 4-4, 4-5, 4-7, 5-13, 5-14, 5-16, 5-19,
5-20, 7-10, 7-33, 7-53

Error flag, 7-10, 7-33

Error handling, 5-20, 7-10, 7-23, 7-33

Error handling attribute, 7-33

Error line, 7-10

Error message, 2-8, 4-4, 4-5, 4-7, 4-11, 5-4, 5-14, 5-16,

5-17, 5-18, 5-19, 5-20, 7-10, 7-23, 7-24, 7-33, 9-9

Error message line, 7-33

Error messages, 1-5

Error processing, 5-19, 5-20, 5-22, 7-23

Error screen, 9-9

ESDS, 10-16, 10-17

ET operator, 5-20

Execute Request or Procedure operator
 XR operator, 11-4

Execution, 7-5

Execution order of request/procedures, 11-4

Execution trace operators, 7-5

Execution tracing, 12-1

 Tracing, 1-4

Executor, 1-1, 1-5, 1-6, 1-7, 1-9

Existence, 9-10

Existence test, 7-23

Explicit return from subroutine, 7-28

Explicit temporary fields, 7-2, 9-6

Explicit temporary field, 10-19

Exponent, 9-2

Exponential field
 Floating point field, 9-1

Extended Attributes, 5-6

Extended attributes, 5-7, 5-23

Extended highlighting, 5-7

External calls, 7-9

External subroutine, 7-9

F

Fast path application, 10-12

Fast path data base, 1-15

FCT, 1-15, 3-1, 4-4

FCT name, 10-15, 10-16, 10-17

FD statement, 3-1, 10-2, 10-4, 10-10, 10-11, 10-15,

10-16, 10-17

FDestination field, 4-19

Field, 9-2

Field Actions, 6-23

Field actions, 1-11, 4-7, 4-8, 4-11, 4-19, 4-30, 7-22, 7-24, 9-4, 9-9, 10-4, 10-18, 12-2

FIELD command, 7-2, 7-6, 9-6, 10-19

Field content trace, 12-3

Field conversions, 9-1, 9-3, 9-4, 9-8

Field cross references

- Design Manual, 1-4

Field description for automatically formatted screens, 5-11

Field designators, 5-2, 5-3, 5-9, 5-12

Field handling, 9-1

Field invalidity

- Invalid, 9-3

Field length, 10-3, 10-4, 10-9

Field location, 10-4, 10-8

Field names, 1-15, 7-6, 10-3, 10-4, 10-5, 10-19

Field overflow, 9-9, 9-10

Field sensitivity, 10-3

FIELD statement, 10-4

Field types, 9-1, 10-3, 10-4, 10-7, 10-9

Field validity, 7-35

Fields location, 10-3

File Control Table

- FCT, 1-15

File definition, 1-4, 1-10, 1-15, 1-18, 3-1, 4-7, 7-2, 10-1, 10-2, 10-3, 10-4, 10-8, 10-10, 10-15, 10-16, 10-17, 10-18, 10-19

File identification, 1-15, 10-4, 10-15, 10-16, 10-17

File identification name, 3-1

File type, 6-20, 7-6

Fixed length records, 10-15, 10-17

Fixed occurring segments, 10-14, 10-18

Fixed point binary fields

Binary fields, 9-1

Flag fields, 1-11, 1-13, 3-2, 4-2, 4-10, 4-11, 4-23, 7-39, 9-9

Flat screen, 6-1

Floating point field, 9-1, 9-2, 9-4

Floating point validation, 5-14

Format library, 1-16

Formatted listing, 10-2

Function key, 2-4, 2-5, 4-4, 4-14, 4-23, 7-20

G

GE operator, 7-4

Generator, 1-1, 1-3, 1-4, 1-5, 1-9, 1-13

Generic browsing, 2-5, 8-3, 8-4, 8-7, 8-14

Generic key, 2-5

Generic search, 8-1

GENLIB

- Generator, 1-3

Glossary, 1-4, 10-2, 11-6

Glossary of terms, 22-1

GO command, 7-6, 7-8, 7-16, 7-17

GO operator, 7-4, 7-8, 7-13, 7-29, 7-31

Go-select

- GS operator, 7-4

GOSC flag, 7-8

Grouping character, 5-14

GS operator, 7-4, 7-7, 7-13, 7-29, 7-31

GT operator, 7-4

H

Hierarchical levels, 10-2

Hierarchical screen

- Structured screen, 6-1

Hierarchical structure

- Hierarchy, 10-3

Hierarchically structured files, 3-1

Hierarchy, 10-2, 10-3, 10-8, 10-14

High intensity, 5-6

Intensity, 5-3

Highlight, 5-16, 5-18

Highlight designator, 5-2, 5-3

Highlighting, 5-6, 5-7, 5-19, 7-33

I

I action, 4-7, 6-18, 7-36, 10-17

I/O areas, 12-1, 12-2

I/O calls, 7-53, 12-2

I/O PCB, 10-12

I/O specification, 9-8

IF command, 7-6, 7-9

IF operator, 4-8, 4-9, 4-10, 4-13

IGNORE action, 4-11

Ignore action, 4-8, 4-19, 4-20, 5-13, 5-18, 6-15, 6-18, 7-27, 7-35, 7-38

Ignored segment occurrences, 7-38

Illegal field data, 9-9

Implicit return from subroutine, 7-28

Implicit temporary fields, 7-2, 9-6, 9-7

IMS data bases, 1-9, 1-15, 3-1, 10-1, 10-2, 10-3, 10-19

IMS fast path application, 10-12

IMS fast path data bases, 10-10

IMS secondary indexing, 10-8

IMS sequential dependent, 10-11

IMS structure, 10-3

IMS/DC conversational processing, 10-18, 10-19, 11-2

IMS/DC executor, 1-6, 1-7

In-memory insertion, 7-25

In-memory update

Updates in memory, 6-15

Inconsistent segment actions, 4-11

Inherit action, 4-8, 4-11, 4-12, 4-13

Inhibit keyboard input, 5-6

Initial entry, 2-5

Initialization of working storage, 10-19

Initializing a pseudo file, 7-19, 10-18

Initiating loops, 7-45

Input fields

Input screen fields, 4-18

Input processing, 6-19

Input required, 5-18, 5-20

Input screen, 4-2, 4-19, 4-22, 4-23, 6-1, 6-18, 6-19, 6-22, 11-4

Input screen content, 4-2, 4-28

Input screen data, 1-11

Input screen error processing

Redisplay input screen, 5-19

Input screen field validation, 5-13

Input screen fields, 4-2, 4-5, 4-7, 4-16, 4-18, 4-19, 4-20, 4-23, 5-4, 5-6, 5-11, 5-13, 5-18, 5-20, 6-14, 7-10, 7-19, 7-33, 7-41, 9-8

Input screen redisplay

Redisplay input screen, 5-19

Input Screen Requests/Procedures, 7-3

Input screen requests/procedures, 11-4, 11-5

Input sequence for statements, 11-5

Input Transaction Code, 5-2

Input transaction code, 4-27

Input validation

Validation, 4-4

Input virtual screen fields, 9-8

Input/Output specification, 4-6, 4-16, 4-18

Insert action, 4-7, 6-18, 7-36, 10-17

Insert duplicate, 5-20

Insert operations, 10-15

Inserted occurrences, 7-36

Inserted segment occurrences, 7-22, 7-24, 7-35, 7-38

Inserting records
 Insert action, 10-17

Insertion, 7-25, 7-35

Integer pattern, 5-18

Intensity, 5-2, 5-3, 5-6

Interaction, 4-5, 4-8, 4-14, 4-16, 4-22, 4-28, 6-1, 6-7, 6-10, 6-12, 6-14, 6-15, 6-22, 7-2, 7-14, 7-18, 8-5, 10-18, 10-19, 12-1, 12-3

Intermediate Structure, 7-14

Intermediate structure, 5-23, 6-1, 6-13, 6-15, 7-14, 7-23, 7-27, 7-31, 7-37, 7-38, 7-39, 7-40, 7-47, 7-49, 7-52, 7-53, 8-7

Intermediate structure looping trace, 12-1

Invalid, 7-36, 9-3, 9-9, 9-10, 11-2, 12-3

Invalid character, 9-10

ISAM data bases, 1-15, 3-1, 10-1, 10-13

IT statement, 1-14

Item data base, 2-1, 2-2

Item tracking, 1-14

K

Key, 10-2

Key conversions, 9-4, 9-8

Key field, 4-7, 4-9, 4-12, 9-4, 9-8, 10-3, 10-4, 10-5, 10-7, 10-8, 10-9, 10-10, 10-14, 10-15, 10-16, 10-17

Key fields, 1-15, 3-1, 10-18

Key Sequenced Data Set
 KSDS, 10-15

Keyboard locks, 5-6

KSDS, 10-15

L

L0 statement, 10-2, 10-3, 10-4, 10-5, 10-7, 10-8, 10-10, 10-15, 10-17

Label, 7-6, 7-8

Labeled statement, 7-8

LCHILD statement, 10-8

LE operator, 7-4

LEAVE command, 7-6

Length of virtual screen fields, 5-4, 5-12, 7-42

LET command, 7-6, 7-9, 9-4

Level numbers, 10-2, 10-3, 10-5, 10-14

LIBOPNA, 1-10, 11-5

Libraries, 1-14

Library, 1-3, 1-4, 1-9, 1-13, 1-16, 5-2, 10-2, 10-7, 11-1, 11-4, 11-5

Library Open, 1-10, 1-17, 11-5

Library retrieval and maintenance, 11-6

Library Utilities, 11-6

Library utilities, 1-9, 1-14

Limit number of output pages
 OUTLIM, 11-2

Link edit, 1-4, 1-5, 1-6, 1-7, 1-8

Link Pack Area
 LPA, 1-6

Linkage editor, 1-5

Linking applications, 4-26, 5-2

Linking transactions
 Linking applications, 4-26

LIST expressions, 7-7

Literal trace, 12-3

Literals on the screen
 Screen literals, 5-2

LN operator, 5-4

Load a pseudo file, 7-19

Load library, 1-4, 1-5, 1-6, 1-7, 1-8, 1-16

Load module, 1-5

Load module name, 1-14

Load module size, 4-26

Logic level, 4-23, 4-25, 5-16, 5-17, 7-7

Logical child, 10-5

Logical expressions, 7-7, 7-9, 7-27

Logical operations, 9-1, 9-3

Logical operators, 7-4, 7-7, 9-10

Logical paging

 Operator logical paging, 5-2

Loop, 7-26, 7-43

Loop boundaries, 7-45

Looping, 6-13, 6-18, 6-19, 7-13, 7-14, 7-28, 7-35, 7-38, 7-43, 8-5, 8-6, 8-7, 9-8, 12-3

Looping status, 7-43, 7-53

Looping trace, 12-1

Loops, 7-14, 7-45

Lower level segment, 6-18, 6-19

Lower level segment processing, 7-12

LPA, 1-6

LR statement, 3-1

LS statement, 3-1, 10-2, 10-4, 10-8

LT operator, 7-4

LTERM name, 10-10

M

M action, 4-20, 5-18, 7-36

M5INPUT, 1-13

M5LIB

 Library, 1-3

M5PARM, 1-6, 1-10, 1-17, 5-2

 CONREQ, 7-7

 Device Table, 5-2, 5-8

 LIBOPNA, 1-10, 1-17, 11-5

 TFREQ, 7-2, 9-6

M5SSOUT, 1-13

M5UTIL, 1-14

Main Storage Data Base

 MSDB, 10-10

MAP name, 5-2

MAPSET name, 5-2

Marker character, 5-9

Match action, 4-7, 4-20, 5-18, 7-36

Message, 2-8

Message Format Service

 MFS, 1-6

Message line, 5-4

Message processing, 10-12

Message Sending, 7-33

Message sending operation, 7-10

Message sending operator, 7-5

Message switch, 4-26

Messages

 Error messages, 1-5

MFS, 1-6, 1-7, 1-16, 4-28, 5-2, 5-16

MID name, 5-2

Missing, 4-11, 6-7, 6-18, 6-19, 7-22, 7-23, 7-24, 7-27, 7-35, 7-36, 7-38, 7-45, 7-47, 7-49, 7-52, 8-4, 8-14, 9-9, 11-2, 12-3

Missing character, 9-10

Missing segments, 7-36, 7-38

MMDDYY pattern, 5-18

MOD name, 5-2

Move operation, 9-1, 9-3, 9-4

MP region, 1-6

MS operator, 5-17, 5-18, 5-20, 5-22, 7-5, 7-10, 7-22, 7-33

MSDB, 10-10

MSG command, 5-23, 7-6, 7-10, 7-19, 7-22, 7-24, 7-26, 7-33

MSGLINE flag, 5-4

Multi-path nested looping, 7-52

Multiple destinations, 4-11, 4-13, 5-4

Multiple libraries, 1-14

Multiple output screens, 4-22

Multiple transactions, 4-26

Multiply, 7-9

Multiply occurring segment, 6-18, 9-8

N

N action, 4-8, 5-13, 5-18

NE operator, 7-4, 9-10

Nested Looping, 7-45

Nested subroutines, 7-28

NEXT FUNCTION branch, 7-43, 8-6

NEXT FUNCTION command, 7-16, 7-17

NEXT FUNCTION transfer, 7-43

NEXT REQUEST branch, 7-8, 7-43, 8-6

NM operator, 5-11, 5-12, 9-8

No backout, 7-33

No loop started, 12-3

Non-display field, 9-8

Non-display intensity, 5-6

Non-existent

 Missing, 9-10

Non-select

 NS operator, 7-4

Non-synchronized segment, 7-38

Non-terminal related MSDB, 10-10

Normal intensity, 5-6

NOROLLB, 7-33

NOT, 7-7

Not found, 5-19, 5-20, 5-22, 8-4, 8-14

nput/Output specifications, 4-19

NS END, 7-7, 7-27

NS operator, 7-4, 7-7, 7-13, 7-29, 7-31

Null action, 4-8, 5-13, 5-18

Null fields, 11-2

Null screen segments

 Null segment processing, 6-7

Null segment processing, 5-2, 6-7

Null segments, 5-2

Null values, 9-4, 10-18, 10-19

Number of times segment occurs, 10-14

Numeric attribute, 5-13

Numeric constant, 5-18

Numeric fields, 9-1, 9-2, 9-3, 9-4

Numeric specification, 5-6, 5-13

Numeric validation, 5-20

O

Object module, 1-3, 1-4, 1-5, 1-6, 1-7, 1-8, 1-9

OC operator, 4-26, 11-4

 CONV, 11-2

 example of, 11-3, 11-4

 OUTLIM, 5-2, 11-2, 11-3, 11-4

 RECYCLE, 11-2

 START, 4-26, 11-2, 11-3, 11-4

 XIND, 11-2, 11-3, 11-4

Online applications, 1-1

Online load library

 Load library, 1-4

Operating system, 1-14

Operator logical paging, 5-2

OR, 7-7

Order of segment occurrences, 10-2

Originating terminal, 4-23, 4-25

OS operator, 4-25, 7-3, 11-1, 11-2, 11-3, 11-5

OUTLIM, 5-2, 11-2

OUTPUT branch, 7-8, 7-26

OUTPUT CONTROL, 8-6, 8-13

Output Control, 5-2, 7-14, 10-12

Output fields

 Output screen fields, 4-18

Output processing, 6-17, 6-18, 6-20, 7-36, 9-8

Output screen, 4-2, 4-19, 4-22, 4-23, 4-25, 4-26, 6-18, 6-19, 6-20, 6-22, 7-3, 7-25, 7-26, 7-27, 11-1, 11-2, 11-3, 11-4

- Output screen content, 4-28
- Output screen data, 1-11
- Output screen destination, 4-25, 11-1
- Output screen fields, 4-2, 4-5, 4-16, 4-18, 4-19, 4-20, 5-6, 5-9
- Output screen formatting, 7-26
- Output screen name, 11-1
- Output screen pages, 11-2
- Output screen processing, 6-19, 7-14
- Output Screen Requests/Procedures, 7-3
- Output screen requests/procedures, 11-1, 11-5
- Output screen to transaction, 4-26
- Output screen, selection of
 - Screen control, 4-22
- Output Select operator
 - OS operator, 11-1
- OUTPUT TRANSFER, 7-16, 7-17
- OUTPUT transfer, 7-8
- Output transfer, 7-26
- Overdefining a screen field
 - V/S/A entry, 5-4
- Overflow, 9-5
- Overflow character, 9-10

P

- P action, 4-8
- P qualifier, 5-18
- Packed decimal field, 9-1, 9-2, 9-4, 9-5
- PAGENO flag, 11-2
- Paging commands, 5-2
- Parentheses, 7-7, 7-9
- Partial field flags, 7-10, 7-11
- Partial field length, 7-11
- Partial field Start, 7-11
- Partial fielding, 5-16, 7-10, 7-11, 9-9
- Passing control between applications, 4-23, 4-28
- Passing control between transactions, 4-23, 4-28
- Passing data between interactions, 4-6, 4-16
- Path, 4-18, 6-7, 7-45, 7-49
- Path calls, 3-1
- Patterns, 5-18
- PCB, 3-2, 4-2, 10-3, 10-8, 10-12, 10-19
- PCB mask, 10-19
- PCB mask fields, 1-11, 1-13
- PCB masks, 1-15, 3-2, 7-39, 7-53
- PCT, 1-9
- PF function, 7-10
- Physical updates, 1-11, 4-4
- PL/I language, 7-9
- Plan, 1-4, 1-5, 1-7, 1-8, 1-9
- PLI statement, 7-9
- PNUMBER flag, 7-11
- Positive sign, 9-5
- PPT, 1-8, 1-16
- PR statement, 7-1, 7-6, 7-7, 7-8, 7-9, 7-10
- Preload executor, 1-6
- Primary library, 1-14
- Procedural updates, 7-23, 7-24, 7-53
- Procedural updating, 7-40
- Procedural validation, 7-10
- Procedure commands, 7-3, 7-5
- Procedure components, 7-15
- Procedure loops, 7-8
- Procedure sharing, 11-1, 11-5
- Procedure Trace, 12-1
- Procedure trace, 7-53
- Procedure tracing, 12-3
- Procedure type, 7-1, 7-18, 7-29, 7-53
- Procedures, 1-10, 1-11, 1-17, 5-23, 7-1, 9-10, 11-1, 11-4, 11-5, 12-1, 12-2

Processing lower level segments, 7-12
Processing multiple segment occurrences, 7-43
Processing multiple segments
 Looping, 6-13
Processing recurring segments, 7-12
Processing SL statement, 5-1
PROCSEQ parameter, 10-8
Program Control Table
 PCT, 1-9
Program function keys, 2-4, 4-4, 4-10, 4-13, 4-14, 4-23
Program Processing Table
 PPT, 1-8
Program Specification Block
 PSB, 1-6
Program-to-program message switch, 4-26
Programmed symbols, 5-7
Propagation of segment actions, 4-13
Protected, 5-6
PSB, 1-6, 1-7, 1-8, 1-9, 1-13
PSB name, 1-14
Pseudo fields, 7-2
Pseudo file fields, 1-11, 1-13
Pseudo files, 1-15, 3-2, 4-2, 4-8, 6-19, 10-18
PSTART flag, 7-11
PT statement, 1-14

Q

Q action, 4-8, 4-11, 4-12, 4-13
Qualifiers, 4-6, 5-11, 5-18, 6-20, 7-6, 8-2, 8-4, 8-5, 8-10, 9-6, 9-8, 10-19

R

R action, 4-8
R operation, 9-4

R operator, 7-4
Range browsing, 8-1, 8-3, 8-7, 8-10, 8-14
RBA, 10-16, 10-17
RC statement, 1-10, 1-14, 10-12, 11-5
RCT, 1-9
Re-display input screen, 2-9, 4-4, 4-5, 4-7, 5-13, 5-14, 5-16, 5-19, 5-20, 7-10, 7-23, 7-24, 7-26, 7-33, 9-9
Re-initialize temporary fields, 7-14, 7-31
Re-synchronization, 8-7
Re-validation, 9-9
RECORD FORMAT, 10-10
Record Format, 10-11
Record format, 10-2, 10-4, 10-15, 10-16, 10-17
Record key, 10-8, 10-9, 10-10, 10-15, 10-16, 10-17
Record length, 3-1
Record length field, 10-15, 10-17
Record size, 10-2, 10-15, 10-17
Recoverable files, 4-4
Recurring screen segments, 6-1
Recurring segments, 6-1
Recursive invocation of subroutine, 7-29
RECYCLE, 11-2
Redefining a screen field
 V/S/A entry, 5-4
Referencing a DSDB, 8-1, 8-6, 8-7
Relational data bases
 , 1-15
Relational expressions, 7-7
Relational operators, 7-4, 7-7, 9-10
Relational rules, 3-1
Relative Byte Address
 RBA, 10-16
Relative Record Data Set
 RRDS, 10-17
Relative Record Number
 RRN, 10-17

Removal of missing segment occurrences, 7-38
Repeat action, 4-8, 4-11, 4-12, 4-13
Repetitive information, 5-23
Replace action, 4-8
Replace if non-blank action, 4-8
Replace operator, 7-4
Replace with blank or zero action, 4-8
Replacement operation, 9-1, 9-3, 9-4
Request components, 7-12
Request groups, 1-9, 11-6
Request loop, 7-8, 7-14
Request operators, 7-3, 7-4, 7-5
Request pages, 7-1
Request sharing, 11-1, 11-5
Request trace
 Procedure trace, 7-53
Request type, 7-1, 7-18, 7-29, 7-53
Requests, 1-9, 1-11, 1-17, 5-22, 9-10, 11-1, 11-4, 11-5, 12-1, 12-2
Resource Control Table
 RCT, 1-9
Result fields, 9-9
Retrieval, 4-4, 6-15, 8-1
Retrieve, 4-7
Retrieve source statements, 11-6
RETURN, 8-6
RETURN branch, 7-8, 7-29
RETURN command, 7-6, 7-8, 7-16, 7-17, 7-28
RETURNCD flag, 7-9
Reverse video, 5-7
RF statement, 1-14, 6-20, 7-6, 8-1, 8-2, 8-3, 8-9, 10-15, 10-17, 10-18, 10-19
Right-justify screen fields, 5-13
ROLB call, 4-4
ROLLBACK, 4-4
Root key, 8-1, 8-2, 8-4, 8-7

Root segment, 6-1, 10-2, 10-7, 10-8, 10-16, 10-17
RP statement, 1-13, 1-14
RRDS, 10-17
RRN, 10-17
Run control, 1-10, 1-14, 1-17
Run control group, 1-14, 11-5
Run files, 1-14
Run parameters, 1-13, 1-14
Run types, 1-9, 1-14, 1-17

S

S action, 4-8
S qualifier, 5-11, 7-19, 7-41, 9-8
S0 statement, 4-2, 4-6, 4-8, 4-10, 5-4, 5-5, 5-6, 5-13, 5-16, 5-20, 6-15, 9-6, 9-8
Same screen field, 5-4
Sample application, 2-1
Saved pseudo files, 10-18, 11-2
Saved working storage, 10-19
SC statement, 4-23, 6-20, 6-22, 9-7, 11-1, 11-4
Scattered keys, 10-7
Scientific notation
 Floating point field, 9-1
Scratch Pad Area
 SPA, 10-19
Screen content, 4-2
SCREEN CONTROL, 8-6
Screen control, 1-4, 1-10, 1-17, 4-22, 4-23, 4-24, 4-25, 4-26, 5-2, 6-13, 6-20, 6-22, 7-3, 7-14, 11-1, 11-2, 11-3, 11-4, 11-5
SCREEN CONTROL branch, 7-43
SCREEN CONTROL transfer, 7-8, 7-43
Screen control transfer, 7-16, 7-17
Screen Definition, 7-1
Screen definition, 1-4, 1-7, 1-8, 1-10, 1-16, 4-2, 4-5, 4-9, 4-14, 4-27, 4-28, 4-30, 5-23, 6-7, 6-13, 6-14, 6-18,

6-22, 7-41, 8-13, 10-4, 11-5

Screen field assignment, 5-4

Screen field attributes, 5-6

Screen field de-editing, 5-13

Screen field default values, 5-12

Screen field editing, 5-9

Screen field error, 5-19

Screen field input required, 5-13

Screen field length, 5-12

Screen field names, 5-11, 9-8

Screen fields, 4-4, 4-7, 4-9, 4-11, 4-13, 4-14, 4-16, 4-18, 4-23, 5-2, 5-4, 5-7, 5-9, 5-12, 5-18, 5-19, 5-22, 5-23, 6-7, 6-14, 7-25, 7-41, 7-42, 9-4, 9-8

Screen fields in error, 7-33

Screen flow summary
 Design Manual, 1-4

Screen format, 6-1

Screen format library, 1-6, 1-7

Screen formatting utilities, 1-4

Screen Identifier, 5-2

Screen layouts, 1-4, 4-6, 5-2, 5-3, 5-7, 5-23

Screen literals, 5-2, 5-3, 5-7, 6-7

Screen name, 5-2

Screen Name Entry (ER), 7-3

Screen name entry (ER), 7-29, 11-1, 11-4, 11-5

Screen procedures
 Procedures, 5-23, 7-1

Screen processing definition, 4-20
 Screen definition, 4-6
 screen definition, 1-4

Screen request
 Requests, 5-23, 7-1

Screen Segments, 6-23

Screen segments, 5-2, 5-23, 6-1, 6-2, 6-5, 6-14, 6-15, 6-19, 6-22, 7-35, 7-36, 7-39, 9-8, 12-1

Screen structure
 Structured screen, 6-7

Screen Sync Type, 6-20

Screen synchronization, 2-2, 7-35, 7-38, 7-39, 7-42, 8-5, 8-9, 10-4, 12-1

Screen synchronized, 4-8, 6-7, 6-12, 6-13, 6-14, 6-18, 6-20, 6-22, 7-40, 8-5, 10-4, 10-16, 10-17, 10-18

Screen Synchronized Data Base
 SSDB, 1-11

Screen-to-screen flow
 screen control, 1-4

SCREEN/CONTROL branch, 7-8

SD statement, 4-27, 5-1, 5-2, 5-11, 6-7, 8-6, 8-13, 10-12

SE operator, 5-22, 7-4, 7-10, 7-26, 7-33

Search fields, 1-15, 3-1, 4-8, 4-9, 4-12, 9-4, 9-8, 10-3, 10-4, 10-7

Secondary indexing, 10-8

SEGM statement, 10-4, 10-5, 10-8

Segment accessibility rules, 6-18

Segment Action, 12-1

Segment action trace, 12-1

Segment actions, 1-11, 4-7, 4-8, 4-11, 4-12, 4-13, 4-18, 4-30, 5-13, 5-18, 6-7, 6-10, 6-12, 6-13, 6-15, 6-18, 6-19, 6-23, 7-22, 7-27, 7-35, 9-8, 10-4, 10-7, 10-16, 10-17, 10-18

Segment availability, 7-35, 7-38, 7-53

Segment key, 10-2, 10-3, 10-4, 10-5, 10-7, 10-8, 10-9, 10-10, 10-14, 10-16, 10-17
 Key field, 4-7

Segment length, 1-15, 3-1, 10-5

Segment names, 3-1, 10-2, 10-3, 10-4, 10-8

Segment not found, 5-19, 5-20, 5-22

Segment numbers, 5-23, 10-2, 10-3, 10-4, 10-14

Segment occurrences, 6-12, 6-13, 6-18, 10-2

Segment order, 10-2, 10-19

Segment Search Argument
 SSA, 3-1, 10-7

Segment sizes, 1-15

Segment types, 10-2

Segments, 3-1, 4-4, 5-23, 6-1, 6-2, 6-7, 6-10, 6-12, 6-14,

6-18, 7-35, 10-2, 10-3, 10-4, 10-5, 10-14
SELECT ALL, 6-13, 6-19, 6-20, 6-22
Select segment occurrences for output screen, 7-53
Selecting the output screen
 Screen control, 4-22
Selection, 7-3
Selection for output, 7-25
SENSEG statements, 10-8
Separating screen fields, 7-42
Sequential dependent, 10-11
SET command, 5-23, 7-6, 7-10, 7-26, 7-33
Set Error, 7-33
Set Error operator
 SE operator, 5-22
Set error operator, 7-4
SG operator, 6-13, 6-18, 6-19, 6-20, 6-22
Sharing procedures, 11-1, 11-5
Sharing requests, 11-1, 11-5
Sign, 9-2, 9-3, 9-5, 9-8
Signon page, 5-2
Simple loops, 7-45
Single path loop, 7-45
Single path nested looping, 7-45
SL statement, 5-3, 5-7, 5-8, 5-23
Sn statement, 4-10, 4-14, 5-9, 5-12, 5-16, 5-20, 5-22, 5-23, 9-6, 9-8
Source fields, 4-2, 4-5, 4-16, 4-19, 4-30, 5-4, 5-9, 7-14, 9-8
Source listing, 1-4
Source statement retrieval, 1-9, 11-6
Source statement retrieval run, 1-13, 1-14
Source/Destination field
 Source fields or destination fields, 4-6
SPA, 10-18, 10-19, 11-2
Split screen segments, 6-3
SQL, 1-3, 1-4, 1-7, 1-8, 1-9, 1-15
SQL calls, 10-12, 12-2
SS statement, 5-23
SSA, 3-1, 10-4, 10-7
SSDB, 1-11, 1-13, 4-2, 4-8, 6-1, 6-18, 6-19, 6-20, 6-22, 6-24, 7-14, 7-22, 7-23, 7-38, 7-39, 7-53, 8-1, 8-5, 10-16, 10-17, 10-18, 12-2
SSDB and DSDB, 8-5
SSDB fields, 7-19, 7-24, 7-28
SSDB I/O call trace, 12-2
SSR
 Source Statement retrieval run, 1-13
START, 11-2
Starting loops, 7-45
Statement label, 7-6, 7-8
Statement types, 1-14, 11-5
Status of loops, 7-43, 7-53
Sterling MARK V Library
 Library, 1-3
Storage Type, 10-18, 10-19
Storage type, 10-18
Structure, 10-8
Structured file, 3-1
Structured records, 10-2, 10-3, 10-13
Structured screens, 5-2, 5-23, 6-1, 7-27
Structured work area
 pseudo files, 1-16
Structured working storage, 10-18
SUB branch, 7-8, 7-29
Subordinate segment processing, 7-12
SUBROUTINES, 7-8
Subroutines, 7-3, 7-14, 7-26, 7-28, 7-31, 12-3
Subtract, 7-9
Subtract action, 4-8
Suppress missing, invalid, and overflow indicators, 11-2
Sync qualifier, 8-2, 8-4, 8-5, 8-10

Synchronizing, 8-5
Synchronizing field, 10-16, 10-17
Synchronizing field 1, 8-3, 8-12
Synchronizing field 2, 8-3
Synchronizing fields, 7-35, 7-39, 7-53, 8-1, 8-2, 8-3, 8-5, 8-7, 8-10, 8-13, 8-14, 9-4, 9-8
Syncpoint rollback, 4-4
Synonym, 10-2
System delimiter, 5-9
System fields
 Flag fields, 1-13

T

T qualifier, 9-6
Table, 3-1
TCONTROL flag, 12-2
Teleprocessing monitor, 1-14
Temporary field definitions, 7-2
Temporary fields, 1-11, 1-13, 3-2, 4-2, 4-23, 7-2, 7-14, 7-39, 7-53, 8-1, 9-6, 9-7, 9-8, 9-9, 10-19
Temporary storage, 10-19
Terminal group, 5-2
Terminal ID, 4-25, 11-3
Terminal related MSDB, 10-10
Terminal type, 5-2, 5-8
Terminate an application, 4-25
Terminology, 22-1
Testing, 1-4, 1-13
Testing for existence, 7-23
Testing for invalid, 7-36
Testing for missing, 7-36
TF statement, 7-1, 7-2, 9-6, 10-19
TFREQ, 7-2, 9-6
TP calls, 12-1
TRACE command, 7-6, 12-3

Trace commands, 7-53, 9-6, 12-1
Trace control, 12-1, 12-2
Trace flags, 12-2, 12-3
Trace messages, 12-2
Trace operators, 7-5, 7-53, 9-6, 12-1, 12-3
Trace output, 12-3
Tracing, 1-4, 1-13, 4-8, 4-12, 4-30, 6-18, 6-19, 6-22, 7-19, 7-43, 7-53, 8-1, 10-3, 10-12, 12-1
Tracing message, 7-35
Tracing options, 1-13, 1-14, 12-1, 12-2, 12-3
TRANCODE flag, 4-26, 4-28, 5-2, 11-2, 11-3
TRANSFER-TO locations, 7-16, 7-17
TRANSACT, 1-7
Transaction code, 4-28, 5-2, 11-3
Transaction initial entry, 2-5
Transaction initiation, 11-2
Transaction linking
 Application linking, 4-26
Transaction name, 1-14
 Transaction code, 4-28
Transaction Work Area
 TWA, 10-19
Transactions, passing control, 4-23, 4-28
TRANSFER command, 7-6, 7-8
Transfer to location, 7-8
TRANSFER TO NEXT FUNCTION, 7-16, 7-17
Transfer to NEXT FUNCTION, 7-43
TRANSFER TO OUTPUT, 7-8, 7-16, 7-17
Transfer to output, 7-26
TRANSFER TO SCREEN control, 7-16, 7-17
Transfer to SCREEN CONTROL, 7-43
Transfer to screen control, 7-8
TWA, 10-19
Type, 7-1, 7-2
Type 1, 7-3, 7-18, 7-19, 7-20, 7-41, 10-18, 11-4, 12-1, 12-3

Type 2, 7-3, 7-18, 7-22, 7-23, 7-36, 7-38, 7-40, 11-4
Type M, 7-3, 7-18, 7-24, 7-25, 7-38, 7-40, 11-4, 12-2
Type N (or Blank), 7-2
Type O, 7-3, 7-14, 7-18, 7-25, 7-38, 7-45, 11-1
Type S, 7-3, 7-28

U

U action, 4-7, 7-36, 7-40
UC statement, 1-14
Unavailable
 Missing, 7-35, 9-10
Underlined, 5-7
Unordered, 10-2
Unstructured screens, 6-1
Unsynchronized occurrences, 6-18
Unsynchronized segment, 7-38
Updatable segments, 7-40
Update action, 4-7, 7-36, 7-40
Update operations, 10-15, 10-17
Update procedurally, 7-23
Update SSDB, 7-23
Update/Insert action, 7-40
Update/insert action, 4-7, 10-17
Updated segment occurrences, 7-36
Updated segments, 7-24
Updates, 4-4
 in-memory, 1-11, 4-4, 6-15, 6-17, 7-24, 7-25
 physical, 1-11, 4-4, 6-17, 6-18, 6-19
 procedural, 7-24
Updating segments, 7-40
Utility control, 1-14
Utility run, 1-14

V

V action, 4-7, 7-40, 10-17

V/S/A entry, 5-4, 6-5, 9-8
Validate data base segment, 7-22
Validate input screen fields, 7-19, 7-22
Validate updates, 7-24
Validating
 Validation, 5-13
Validating input screen fields
 Validation, 5-16
Validation, 4-4, 4-6, 4-11, 4-30, 5-13, 5-18, 5-19, 5-20, 5-23, 6-7, 6-10, 7-3, 7-10, 7-19, 7-22, 7-33
Validation field, 5-18
Validation operators, 5-13, 5-16, 5-17, 5-18
Validity, 9-10
Variable length records, 10-15, 10-17
Variably occurring segments, 10-14, 10-15, 10-17, 10-18
Verify updated segments, 7-24
View, 3-1, 10-2
Virtual input screen fields, 7-41
Virtual key fields, 10-4, 10-8, 10-9, 10-10, 10-16, 10-17
Virtual output screen fields, 7-41
Virtual screen field, 5-4, 5-12, 5-22, 6-7, 7-41, 9-8
Virtual/Same/After entry
 V/S/A entry, 5-4
VSAM alternate indexing, 10-17
VSAM cluster, 10-15, 10-17
VSAM data bases, 1-15, 3-1, 10-15, 10-17
VSAM files, 9-4
VSAM I/O calls, 12-2

W

W qualifier, 10-19
Working storage, 1-15, 3-2, 4-2, 7-2, 7-53, 10-19
Working storage fields, 1-11, 1-13, 7-39, 9-8, 9-9

X

X action, 4-8, 4-11, 4-19, 4-20, 5-13, 6-15, 6-18, 7-27, 7-35, 7-38

XATTR specification, 5-7

XDFLD name, 10-7, 10-8

XDFLD statement, 10-8, 10-9

XIND, 11-2

XR operator, 7-3, 11-4, 11-5

Z

Zoned decimal field, 9-1, 9-2, 9-4