

INTERCOMM

COBOL PROGRAMMERS GUIDE

LICENSE: INTERCOMM TELEPROCESSING MONITOR

Copyright (c) 2005, 2022, Tetragon LLC

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Use or redistribution in any form, including derivative works, must be for non-commercial purposes only.
2. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
3. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

COBOL Programmers Guide

Publishing History

<u>Publication</u>	<u>Date</u>	<u>Remarks</u>
First Edition	September 1973	This manual corresponds to Intercomm Release 6.0.
Second Edition	December 1982	This manual corresponds to Intercomm Release 9.0.
Third Edition	April 1994	Updates and revisions for Releases 9 and 10.

NOTES:

The following enhancements are for Release 10 only:

- 3-byte MSGHBMN number
- INTSORT (in-core table sort) service routine
- Dynamically loaded programs above the 16M line
- DWSCHK, INDUMP and SAM parameters - SYCTTBL macro
- REJECT (of msg queued for delayed subsystem) parm - SYCTTBL
- DWSSNAP Facility (online debugging/DWS snaps)
- VSAM data set access under Dynamic File Allocation (DFA)
- Subsystem message flushing (SSFL command)
- GETDATE macro
- SCTL system control/debugging command
- FTUN/SSUP, PRTY, SPAC, TCTV subsystem control commands
- Backend initiation of a transaction on an LU6.2 link
- Table Facility
- Page Facility using Table Facility (no PAGETBLE/data sets)
- VS COBOL II support

The following are desupported under Release 10:

- AMIGOS file access method
- DISAM file access method

The material in this document is proprietary and confidential. Any reproduction of this material without the written permission of Isogon Corporation is prohibited.

PREFACE

Intercomm is a state-of-the-art teleprocessing monitor system executing on the IBM System/370 and System/390 family of computers and operating under the control of IBM Operating Systems (XA and ESA). Intercomm monitors the transmission of messages to and from terminals, concurrent message processing, centralized access to I/O files, and the routine utility operations of editing input messages and formatting output messages, as required.

The COBOL Programmers Guide explains the organization of Intercomm from the application programmer's point of view and illustrates the procedures for creating COBOL application programs and integrating them into the Intercomm environment.

Syntax used in describing the coding of JCL or application program statements is:

- () A pair of braces indicates the presence of a choice: code elements contained within the braces represent alternatives, one of which must be chosen. The braces are not to be coded.
- [] A pair of brackets indicates an optional parameter which may be omitted depending on access requirements as described in the accompanying text. The brackets are not to be coded.
- A parameter consisting partially or solely of lower case letters represents the generic (Intercomm) name of the value. The programmer must substitute the actual name used for defining the data area within the specific program.

As a prerequisite to this manual, it is assumed that the user is familiar with the Intercomm Concepts and Facilities Manual. The following manuals describe in further detail facilities referenced in this manual:

- Message Mapping Utilities
- Utilities Users Guide
- Store/Fetch Facility
- Dynamic Data Queuing Facility
- Page Facility and Table Facility
- Operating Reference Manual: "Message Management"
"File Management"

Note: the term COBOL refers to both OS/VS and ANS COBOL and to VS COBOL II programs. The term OS/ANS COBOL refers to all supported COBOL compilers except VS COBOL II. A mixed environment is supported except as noted in Chapter 3. A distinction is made if necessary.

INTERCOMM PUBLICATIONS

GENERAL INFORMATION MANUALS

Concepts and Facilities

Planning Guide

APPLICATION PROGRAMMERS MANUALS

Assembler Language Programmers Guide

COBOL Programmers Guide

PL/1 Programmers Guide

SYSTEM PROGRAMMERS MANUALS

Basic System Macros

BTAM Terminal Support Guide

Installation Guide

Messages and Codes

Operating Reference Manual

System Control Commands

CUSTOMER INFORMATION MANUALS

Customer Education Course Catalog

Technical Information Bulletins

User Contributed Program Description

FEATURE IMPLEMENTATION MANUALS

Autogen Facility

ASMF Users Guide

DBMS Users Guide

Data Entry Installation Guide

Data Entry Terminal Operators Guide

Dynamic Data Queuing Facility

Dynamic File Allocation

Extended Security System

File Recovery Users Guide

Generalized Front End Facility

Message Mapping Utilities

Model System Generator

Multiregion Support Facility

Page Facility

Store/Fetch Facility

SNA Terminal Support Guide

Table Facility

TCAM Support Users Guide

Utilities Users Guide

EXTERNAL FEATURES MANUALS

SNA LU6.2 Support Guide

TABLE OF CONTENTS

		<u>Page</u>
Chapter 1	INTRODUCTORY CONCEPTS OF ON-LINE SYSTEMS	1
1.1	Introduction	1
1.2	The On-Line System Environment	1
1.3	Batch Environment vs. On-Line Environment	3
1.4	Single-Thread vs. Multithread Processing	4
1.5	Program Functions in the On-Line Environment	6
1.5.1	Monitor Control Functions	7
1.5.2	Application Processing Functions	7
Chapter 2	MESSAGE PROCESSING AND CONTROL UNDER INTERCOMM	9
2.1	The Intercomm Environment	9
2.2	System Components	11
2.2.1	Front End	11
2.2.2	Back End	11
2.2.3	LU6.2 Link	11
2.3	System Programs	12
2.4	Subsystems	16
2.4.1	Reentrant vs Nonreentrant Subsystems	16
2.5	Intercomm Tables	17
2.6	Interfacing with the Intercomm Monitor	18
2.7	Intercomm Message Header	19
2.7.1	MSGHQPR and MSGHVMI Fields	22
2.8	Intercomm Message Flow Using Message Mapping	22
2.9	Intercomm Message Flow Using Edit and Output	24
2.10	The Intercomm System Log	26
2.11	Additional Application Processing Facilities	30
Chapter 3	CODING AN INTERCOMM SUBSYSTEM IN COBOL	31
3.1	Program Structure	31
3.2	Message Processing Concepts	35
3.3	Subsystem Coding	38
3.3.1	Message Switching Between Subsystems	41
3.4	Reentrant Coding Conventions	45
3.4.1	XA/ESA Extended Storage Loading Requirements	45
3.4.2	Dynamic Working Storage (DWS) Protection Option...	46
3.4.3	VS COBOL II Program Conversion and Support	46.1
3.5	Restarted Messages	46.4
3.6	DWSSNAP Facility	46.4
Chapter 4	USING THE MESSAGE MAPPING UTILITIES	47
4.1	Concepts	47
4.2	Processing	47
Chapter 5	USING THE EDIT UTILITY	49
5.1	Concepts	49
5.2	Processing Results	49
Chapter 6	USING THE FILE HANDLER	51
6.1	General Concepts	51
6.1.1	Subsystem Processing	52

	<u>Page</u>
6.2	Calling Service Routines 54
6.2.1	Automatic Error Checking 55
6.3	Select, Release Functions 56
6.3.1	Closing A File 57
6.4	Exclusive Control for Non-VSAM Files 57
6.4.1	Release Exclusive Control--RELEX 59
6.5	Sequential Access Method (SAM) Processing 60
6.5.1	File Handler Service Routines 60
6.5.2	Undefined Record Format and Record Length 61
6.5.3	Variable-Length Record Format and Record Length... 61
6.6	Indexed Sequential Access Method (ISAM) Processing . 62
6.6.1	File Handler Service Routines 62
6.7	Direct Access Method (BDAM) Processing 64
6.7.1	File Handler Service Routines 64
6.8	Virtual Storage Access Method (VSAM) Processing 67
6.8.1	File Handler Service Routines 67
6.8.2	VSAM Processing Options 69
6.8.3	FHCW Reason Codes for VSAM 70
6.8.4	Exclusive Control for VSAM Files 70
6.8.5	Loading an ESDS Data Set 70
6.8.6	Alternate Path Processing of Keyed VSAM Files 71
6.9	ISAM/VSAM Compatibility Under Intercomm 74
Chapter 7	USING THE OUTPUT UTILITY 75
7.1	Concepts 75
7.2	Processing 75
Chapter 8	CONVERSATIONAL SUBSYSTEMS 79
8.1	General Concepts 79
8.1.1	Conversational Applications 79
8.1.2	Conversational Transactions 79
8.1.3	Retention of Information 80
8.2	Implementing Conversational Subsystems 81
8.3	Saving Information in USERSPA 82
8.4	Saving Information with Store/Fetch 84
8.5	Saving Information on a Dynamic Data Queue 86
8.6	Saving Information via the CONVERSE Service Routine 88
8.6.1	Subsystem Design Using CONVERSE 90
8.7	Design Considerations in Conversational Processing 93
8.7.1	Control of the Input to Conversations 93
8.7.2	Assigning a Verb to a Terminal 93
Chapter 9	USING INTERCOMM SERVICE ROUTINES AND FACILITIES 95
9.1	Reentrant COBOL Interface Routine (COBREENT) 95
9.2	COBOL Intersubsystem Queuing (COBPUT) 97
9.3	Input Message Switching (MSGCOL) 99
9.4	Free Dynamic (Message Area) Storage (COBSTORF) 100
9.4.1	Intersubsystem Message Queuing Via Message Collection (MSGCOL) 100

	<u>Page</u>
9.5	Send Message to Front End (FESEND) 102
9.6	User Log Entries (LOGPUT) 104
9.7	Calling User Subroutines from Reentrant COBOL Subsystems 105
9.7.1	Defining User Subroutines to Intercomm 105
9.7.2	Interfacing to User-Coded Assembler Language Subroutines 105
9.7.3	Interfacing to User-Coded COBOL Subroutines 105
9.8	Front End Control Messages 106
9.8.1	Front End Data Queuing 107
9.8.2	Front End Feedback Messages 107
9.8.3	Front End Queue Release 108
9.9	In-core Table Sort Facility (INTSORT) 109
9.10	Other Intercomm Service Facilities 110
9.10.1	Features Accessible via Assembler Macros 110.1
Chapter 10	SAMPLE PROCESSING PROGRAMS 111
Chapter 11	SUBSYSTEM TESTING 129
11.1	Introduction 129
11.2	Debugging Application Program Problems 129
11.3	Testing a Subsystem with the Front End Simulator ... 130
Chapter 12	SUBSYSTEM TESTING IN TEST MODE 167
12.1	Introduction 167
12.2	Testing a Subsystem in Test Mode 167
Chapter 13	VS COBOL II TESTING 197
Appendix A	COBOL JCL PROCEDURES 225
A.1	OS/VS and ANS COBOL Compile and Link JCL 225
A.2	VS COBOL II Compile and Link JCL 226
Appendix B	SOURCE STATEMENT LIBRARY COPY MEMBERS 229
Appendix C	INTERCOMM TABLE SUMMARY 239
Appendix D	CALLING USER SUBROUTINES FROM REENTRANT SUBSYSTEMS 243
Appendix E	NONREENTRANT SUBSYSTEMS 247
E.1	Introduction 247
E.2	Considerations for Using CONVERSE 247
INDEX 253



LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
1	On-line Transaction Processing in a Multiprogramming Environment	2
2	Differences Between Batch and On-line Environments	3
3	Multithreading in an On-line Environment	5
4	The Intercomm Environment	9
5	Intercomm Control Sequence	10
6	Intercomm System Components	13
7	Intercomm Message Header Fields	20
8	Intercomm Message Flow Using Message Mapping	23
9	Intercomm Message Flow Using Edit and Output	25
10	Sequence of Log Entries	27
11	INTERLOG Entries	28
12	Reentrant COBOL Subsystem Structure	32
13	Reentrant Application Program Environment	33
14	Intercomm System Return Codes	34
15	Subsystem Logic Using Message Mapping Utilities	36
16	Subsystem Logic Using Edit and Output Utilities	37
17	Echo Message Example; Reentrant COBOL	42
18	Message Processing Using MMU	48
19	Edit Utility Processing of Fields Omitted or in Error .	50
20	Functions of File Handler Service Routines	51
21	DD Statement Parameters for the File Handler	52
22	Defining File Handler Control Areas	53

<u>Figure</u>		<u>Page</u>
23	File Handler Service Routine Parameters	54
24	Outline of File Handler Return Codes	55
25	File Handler SELECT/RELEASE Return Codes	56
26	Exclusive Control Processing	58
27	File Handler Release Exclusive Control (RELEX) Return Codes	59
28	File Handler Sequential Access Method Return Codes	60
29	File Handler ISAM Return Codes	63
30	File Handler BDAM Option Codes	65
31	File Handler BDAM Return Codes	66
32	File Handler VSAM Call Summary	72
33	File Handler VSAM Return and Feedback Codes	73
34	Message Header Specifications for the Output Utility ..	77
35	Typical Conversational Transactions	80
36	Input Message Data Retention During a Conversation	80
37	User and Terminal Table Space in the USERSPA	82
38	Sample USERSPA Declaration Within a Subsystem	83
39	Conversational Processing Using Store/Fetch	85
40	Conversational Processing Using Dynamic Data Queuing ..	87
41	Conversational Subsystem Logic Using Converse'	89
42	CONVERSE Return Codes	91
43	COBREENT Routine Pointers (REENTSBS)	96
44	COBPUT Return Codes	98
45	Recovery From COBPUT Errors	98
46	Message Collection Return Codes	99
47	FESENDC Return Codes	103

<u>Figure</u>		<u>Page</u>
48	Sample Reentrant Subsystem (IBM ANS COBOL).....	112
49	Sample COBOL Subroutine	126
50	Table Updates to Implement Simulation Mode Testing	132
51	MMU Maps Used by Sample Subsystem	133
52	Input Test Messages Generated via CREATSIM	134
53	Linkedit and Execution JCL for Simulation Mode	136
54	SIM3270 Printout from Simulation Mode Execution	139
55	Simulation Mode Execution Log Printout	161
56	Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL) ...	169
57	Table Updates to Implement Test Mode Testing	180
58	Utilities Table Coding for Test Mode Subsystem	181
59	Test Mode Message Card Formats	183
60	Sample Input Test Messages for Test Mode	184
61	Linkedit and Execution JCL for Test Mode	185
62	Sample Test Mode Execution Snaps	188
63	Test Mode Execution Log Printout	191
64	Sample VS COBOL II Subsystem	198
65	Table Updates for Simulation Mode Testing	214
66	Simulation Mode Linkedit and Execution JCL	215
67	Release 10 Simulation Mode Log Print	218
A-1	Intercomm-supplied COBOL JCL Procedures	225
A-2	VS COBOL II JCL to Compile a Program	226
A-3	VS COBOL II JCL to Compile and Link a Program	227

<u>Figure</u>		<u>Page</u>
B-1	ICOMHEXC COPY Member	230
B-2	New Release 10 ICOMINMG COPY Member	231
B-3	Basic Release 10 ICOMINMG COPY Member	232
B-4	Release 9 ICOMINMG COPY Member	233
B-5	New Release 10 ICOMDWS COPY Member	234
B-6	Basic Release 10 ICOMDWS COPY Member	235
B-7	Release 9 ICOMDWS COPY Member	236
B-8	ICOMSBS COPY Member	237
C-1	Table Names and Associated Macro Instructions	239
C-2	Components and Associated Table Names	241
D-1	COBOL Subsystem/Subroutine Nested Calls	244
D-2	COBREENT Save Areas	244
D-3	Reentrant Subsystem Sample Coding	245
D-4	Reentrant Subroutine Sample Coding	246
E-1	Nonreentrant COBOL Subsystem Structure	248
E-2	Echo Message Example; Nonreentrant COBOL	249
E-3	Nonreentrant Application Program Environment	252

Chapter 1

INTRODUCTORY CONCEPTS OF ON-LINE SYSTEMS

1.1 INTRODUCTION

The objective of most on-line systems is to reduce the time factor from source of input data to the results of data processing. Typical on-line systems applications in the business environment are:

- Data Collection

Transactions may be edited partially on receipt, batch totals may be transmitted and verified, but the bulk of processing of the collected data takes place in the batch mode off-line.

- Inquiry/Update Systems

Transactions are processed immediately to retrieve and/or update information in an on-line data base.

- Message Switching

Transactions consist of administrative data to be rerouted to other terminals in the system.

On-line systems are characterized by a mode of operation which is nonscheduled and transaction-oriented. An operator at a terminal remote from the data processing center enters a transaction (unit of work) by transmitting a message over communication facilities. Each individual transaction is processed immediately, as opposed to batch systems, where transactions are accumulated for processing on a periodic basis (monthly, daily, etc.).

Online systems are designed to satisfy a response time requirement which is the elapsed time between a request for processing by an input message from a terminal to receipt of an acknowledgement, or response to that input message (completion of a transaction).

1.2 THE ON-LINE SYSTEM ENVIRONMENT

Typical on-line message processing application programs operate on one transaction at a time as they come in from terminals. Application programs are usually designed to process only one type of transaction, and the whole environment can be said to be transaction oriented. Input messages can be processed as received, in any order, and the files to be referenced should not be read from beginning to end for each transaction. Instead, the records in files are accessed directly, either through a specific key or some form of cross-reference look-up.

A few applications might require some sequential or list processing of a file, and while this is possible, message processing times for such applications would tend to be high.

Figure 1 shows a computer system schematic depicting a memory layout with an on-line system such as Intercomm, operating in a region or address space as a job under an operating system such as IBM's MVS. The on-line system has its own Transaction Monitor which schedules the activation of transaction processing according to the varying demands in message traffic.

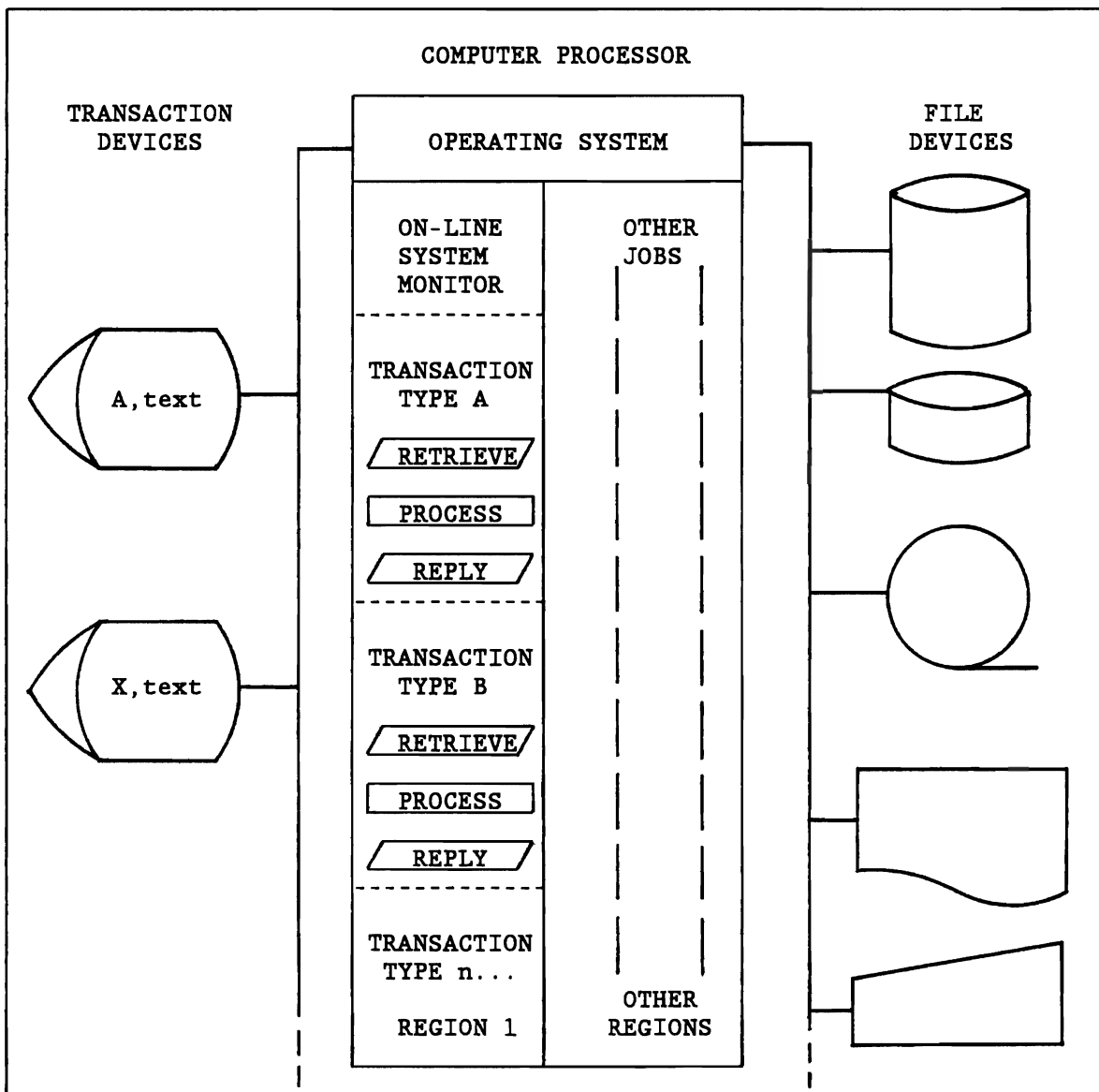


Figure 1. On-line Transaction Processing in a Multiprogramming Environment

The transaction processing programs do not conduct input or output operations with the terminals. This function is provided by the on-line system, which reads input messages from terminals and saves them (queues them) until the appropriate processing program can be activated (scheduled). The message is then retrieved from the queue and passed directly to the processing program by the Monitor. The processing program then requests the Monitor to queue its output response message, and the Monitor handles the terminal output function.

1.3 BATCH ENVIRONMENT VS. ON-LINE ENVIRONMENT

The classical batch processing system flow of input/process/output can be expanded to include message queuing and retrieving in the on-line environment. However, the typical on-line application program need only be concerned with actual transaction processing, because the on-line system does the rest. Figure 2 summarizes some of the differences between batch and on-line environments.

Batch	Online
Scheduled input	Unscheduled input
Single-application job	Multiple-application job
Delayed processing of transactions in batches by type	Immediate processing of individual transactions by type
Transaction input, processing, and output controlled by processing program logic	Terminal input/output events are asynchronous to the processing program

Figure 2. Differences Between Batch and On-line Environments

1.4 SINGLE-THREAD VS. MULTITHREAD PROCESSING

In the on-line environment, the logical path of a program in execution is called a thread. A single-thread system processes one message at a time. However, in a multiple application environment, message volume is such that all message traffic could not be adequately serviced in a single-thread mode. Large queues (waiting lines) tend to develop because messages arrive faster than they can be processed. To alleviate this problem and improve system throughput, the delay time in the processing of one message waiting for an I/O operation may be used for simultaneously processing another message. In this way, several message processing logic paths, or threads, may be active at once. This is referred to as multithreading.

Multithreading is coordinated by the Transaction Monitor, and, depending on message traffic, can occur between two or more programs or within a single program.

To illustrate this, let us assume that we have two transaction processing programs, A and B, and that three messages have arrived for processing; two A-type transactions and one B-type transaction. Programs A and B both require access to records in a file, affording an opportunity for some processing overlap or multithreading. Multithreading would occur between programs A and B if while program A is waiting for file retrieval, program B is activated by the Monitor to carry out its message processing. However, if program A were reentrant, that is, written in such a way that it could handle more than one thread at a time, then multithreading could also occur within program A. This means that while reentrant program A is waiting for a file retrieval for the processing of one message, it may be activated again to carry out the parallel processing of a second, or nth, message. Figure 3 illustrates these concepts.

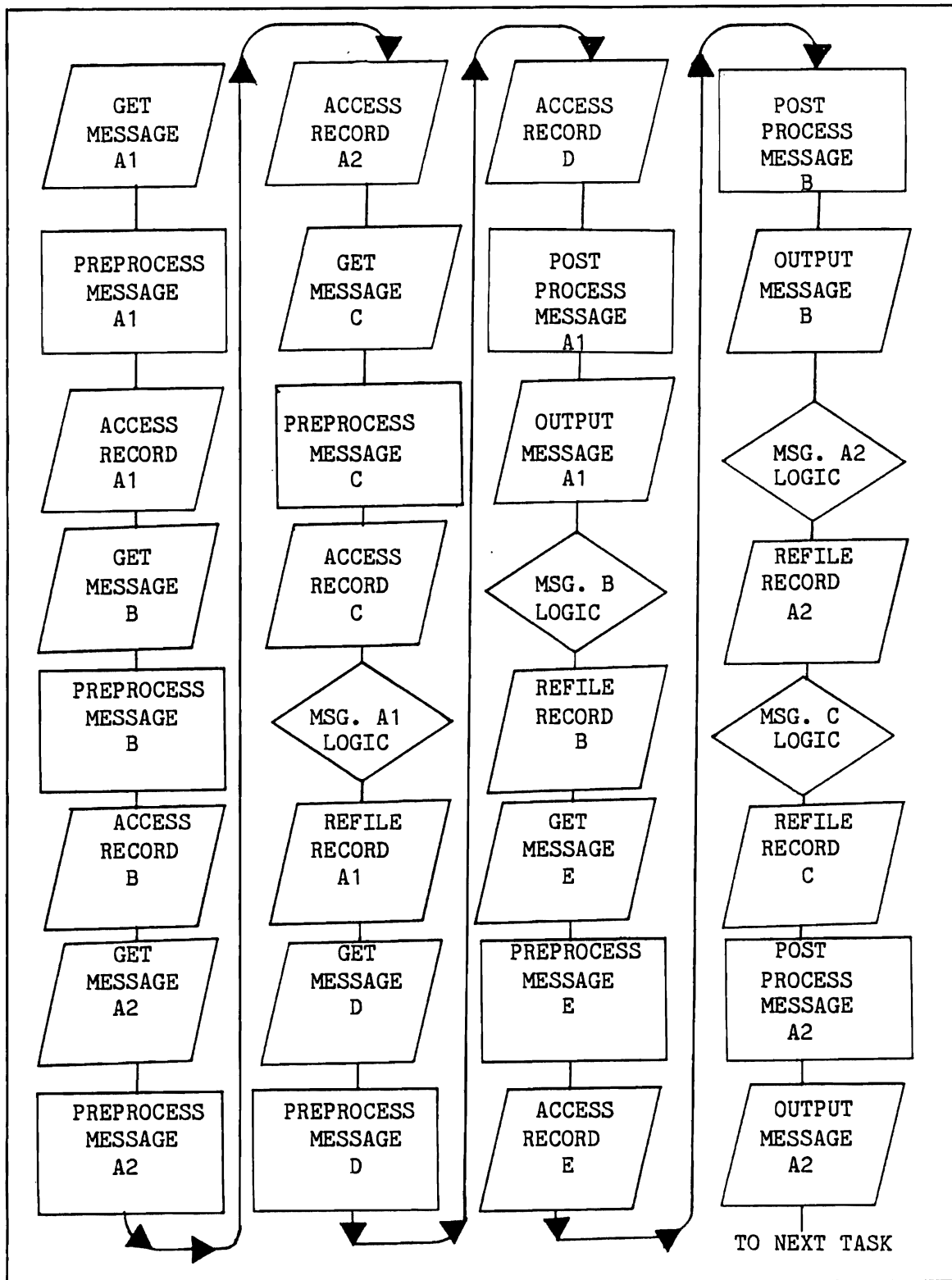


Figure 3. Multithreading in an On-line Environment

1.5 PROGRAM FUNCTIONS IN THE ON-LINE ENVIRONMENT

An on-line system consists of programs to serve four different functions:

- Line Control and Terminal Control
 - Servicing input requests from the various terminal types including transmission error recovery
 - Directing output to the various terminal types including transmission error recovery
 - Intercepting and storing messages to non-operational devices, and retrieval of messages when devices become operational
 - Translation of messages to and from terminal transmission code and EBCDIC code for processing
- Message Processing Control
 - Queuing new input messages until the associated message processing program is scheduled for execution
 - Scheduling message processing programs to obtain best system throughput for message traffic
 - Controlling multithread operation for concurrent processing of several messages
 - Centralizing data file accesses to eliminate redundant operations and provide exclusive control over records during file updates
- Systems Operation Control
 - Security checking functions to restrict certain transactions to specific operators and/or terminals, and to prevent access to unauthorized functions/files.
 - Logging (journaling) of all message traffic
 - Checkpointing, Message Restart, File Recovery and Backout-On-The-Fly (dynamic file backout) facilities
 - Cancellation of message processing programs when a program check or program loop occurs
 - Collect and display system statistics
 - Display and modify system status

- Message (Transaction) Processing
 - Editing text data from terminal input, including format conversion and content editing of individual fields
 - Retrieval and updating of data from on-line files or data bases
 - Preparation of response (output) messages to terminals
 - Queuing of response messages for output to terminals

1.5.1 Monitor Control Functions

The Intercomm System provides complete facilities for:

- Line control and terminal control
- Message processing control
- Systems operation control

1.5.2 Application Processing Functions

Transaction processing logic lies within the coding domain of the application programmer. Intercomm provides the following message and file handling support:

- Format conversion and editing of input fields
- Centralized control of data files
- Format conversion and placement of constant and variable information in response messages and terminal displays
- Queuing of messages (for the same or another terminal, or another application)

The installation-dependent application logic functions then need include only the following:

- Content editing of individual input message fields
- Retrieval and updating of data from on-line files
- Selection of individual fields for the output message(s)



Chapter 2

MESSAGE PROCESSING AND CONTROL UNDER INTERCOMM

2.1 THE INTERCOMM ENVIRONMENT

Intercomm operates under MVS as a job in a region or address space. The job is loaded at the beginning of on-line operations and continues to operate until the terminal network is closed down. Intercomm contains many system programs and application subsystems. Intercomm system programs include the Monitor and other subprograms to handle such things as terminal and peripheral I/O operations. Subsystems are message processing application programs activated by the Monitor. The term "subsystem" includes both application-oriented message processing programs written by users and Intercomm system command processing and utility programs. The Intercomm region contains the execution module itself plus dynamically allocated storage or work space, as illustrated in Figure 4.

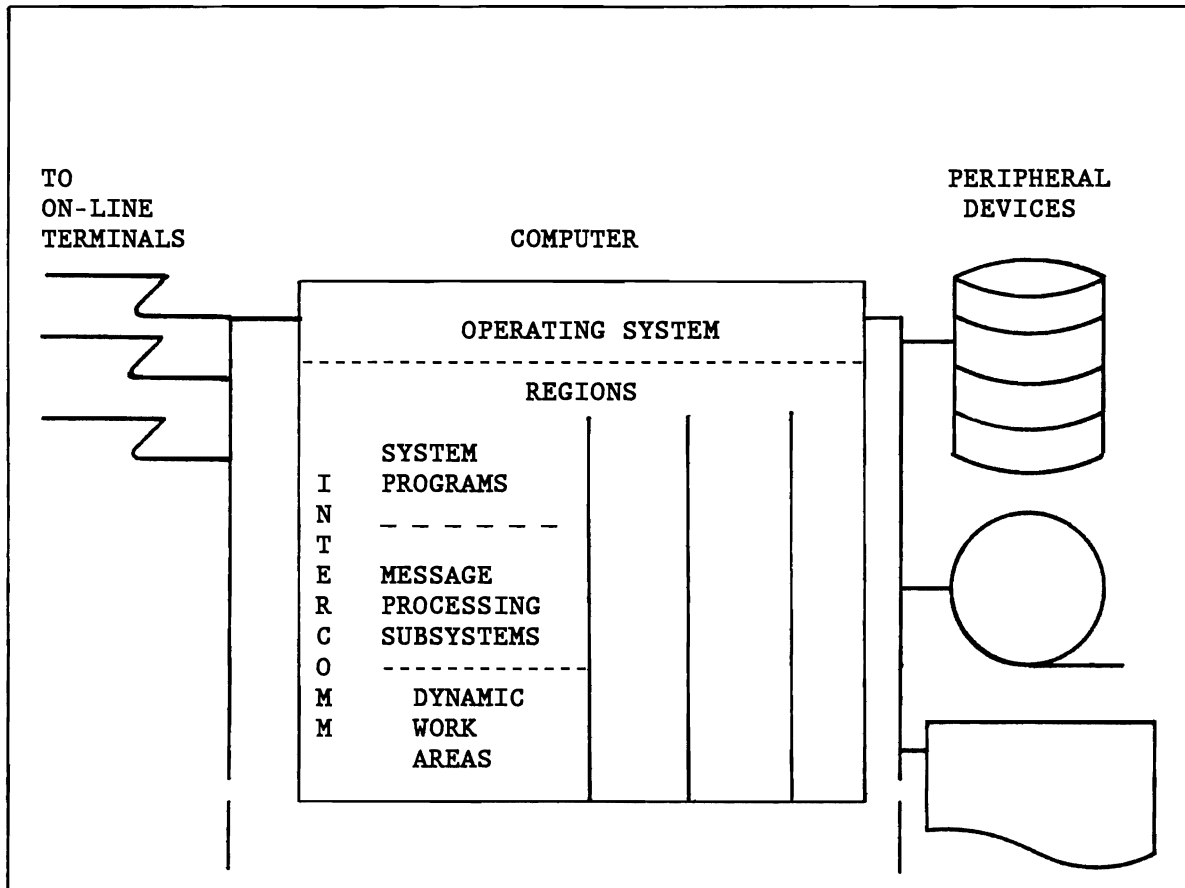


Figure 4. The Intercomm Environment

The system programs are time- or event-driven; the subsystems are message-driven. The Intercomm Monitor calls system programs to handle events and exceptional conditions as they occur, for example, terminal and peripheral I/O interrupts, time-dependent processing, excessive message traffic, and system operator commands.

A subsystem, on the other hand, is called by the system monitor when there are messages queued for it, and it has been scheduled for execution. Subsystems, while executing, can call user subroutines or call system programs to perform services, such as accessing data files and queuing messages for output or additional processing by other subsystems. Figure 5 shows that called system programs and user subroutines will always return to the calling subsystem (or subroutine), just as the subsystem itself, executing as a subroutine of Intercomm, must always return to the system monitor that originally activated it.

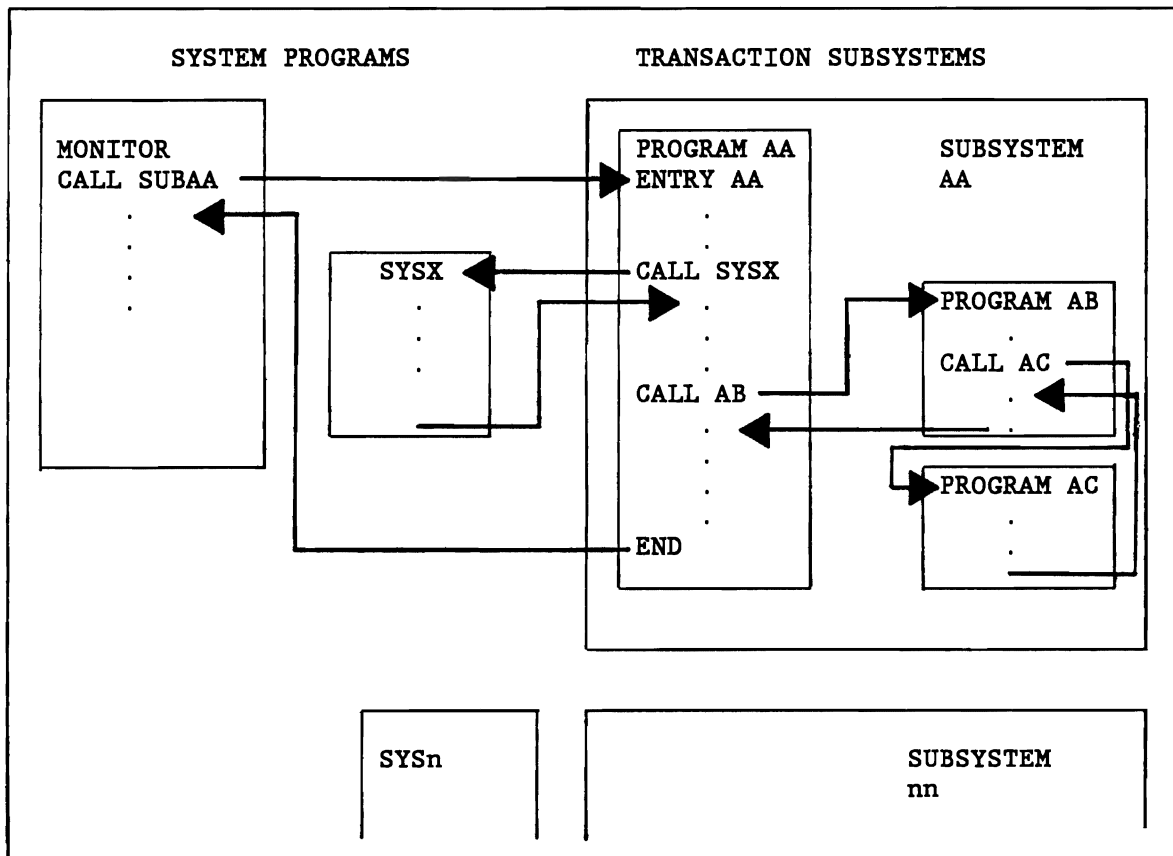


Figure 5. Intercomm Control Sequence

2.2 SYSTEM COMPONENTS

On-line system component programs are often categorized as resident or nonresident, system or user, but typical on-line terminology also distinguishes between Front End and Back End system components.

2.2.1 Front End

The Front End communicates with and monitors all terminals in the network. It receives and sends messages, checks validity, performs security checking if specified, and accomplishes appropriate code translation. The Front End communicates with the Intercomm message processing Back End via input message queuing and output message dequeuing routines. Although Intercomm has its own VTAM Front End, it can also interface with other software Front Ends such as TCAM and BTAM. The term 'terminal' refers to all supported hardware devices (such as the IBM 3270 family, PCs and workstations with 3270 emulation, etc.) and software (LU6.2 link) which can transmit input messages and/or receive output messages.

2.2.2 Back End

The Back End accomplishes all message processing control, system operation control, and processing of individual messages. It is, essentially, the "director" of the entire on-line system operation.

The Front End and the Monitor portion of the Back End are always resident, whereas message processing subsystems can be any combination of resident and loadable. (See Figure 6.) The decision to make a message processing subsystem permanently resident, or loadable, is based upon the trade-offs between response time, frequency of use, and total system core storage requirements.

2.2.3 LU6.2 Link

Support for LU6.2 sessions is an external feature to Intercomm which is available in two modes:

- Basic: (Releases 9 and 10) an upgrade to the VTAM Front End to support secondary LU6.2 sessions with IBM's CICS to receive input messages, queue them for subsystem processing, and route the responses back to CICS.
- Enhanced: (Release 10 only) an add-on to basic support plus an upgrade to the Back End to permit both receipt and initiation of transactions on LU6.2 sessions with other VTAM applications (Intercomm, CICS, etc.). Subsystems may invoke an LU6.2 transaction via the INITLU6 service routine. See SNA LU6.2 Support Guide.

2.3 SYSTEM PROGRAMS

Intercomm system programs are written in Assembler language and include the Monitor, File Handler, high-level language interface routines to maintain reentrancy, and message processing service routines.

The Monitor interfaces with the Front End via message queues and controls the processing of messages by subsystems. It is essentially a traffic director, analyzing message traffic and scheduling subsystems based upon traffic volume and priority criteria. The Monitor has four key components:

- The TP queuing interface, which communicates with the Front End to dequeue input messages or to queue output messages created by subsystems.
- The Subsystem Controller, which schedules, loads and activates the application subsystems, and performs clean up processing when the subsystem returns.
- The Dispatcher, which controls the execution of all events in the system to accomplish multithreading.
- The Resource Manager, which allocates/deallocates and controls dynamic resources (such as core storage) used by system and application programs.

The File Handler is the central Intercomm routine where all peripheral I/O service for data files is controlled. The File Handler issues OPENS, CLOSEs, GETs, PUTs, READs, and WRITEs via the operating system data management facility. Subsystems merely call an appropriate File Handler routine. Therefore, all access methods supported by Intercomm are available to any subsystem program, regardless of the programming language used. The File Handler maintains a single set of control blocks for each file defined to it via standard Job Control Language Data Definition statements, and all programs share this one set of control blocks. Intercomm can control overlapping of peripheral I/O processing, as well as provide standardized error analysis. A file is usually opened only once during an on-line session: at system startup (optional), or if not, then at the time the first I/O is requested. Since files can be accessed concurrently by different subsystems, an exclusive control feature is provided to eliminate difficulties arising when two or more subsystems (or subsystem threads) attempt to update the same record at the same time.

Language interface routines are described in Chapter 3.

The basic message processing service routines are:

- FESEND--which passes an output message to the Front End for transmission to a terminal.
- LOGPUT--which copies a message onto the system log whenever called by a system program or user subsystem.
- MESSAGE COLLECTION--which handles the queuing and dequeuing of all messages destined for subsystems.

Intercomm provides service routines to convert terminal-dependent input messages to a terminal-independent form for application processing. This transformation includes removal of terminal-dependent control characters and conversion of numeric data fields to fixed decimal or binary form, if required. Similarly, for output messages, service routines provide transformation from terminal-independent results of application subsystem processing to terminal-dependent messages for transmission. This includes insertion of terminal-dependent control characters, conversion of numeric fields to character format, if required, and inclusion of title information, if specified. Each of these routines function via user-specified descriptions (tables) of input and output message formats. These service routines are:

- Message Mapping Utilities

This is a set of service routines called by an application program to perform the device-dependent transformations specified by the user for both input and output messages. Validity checking, conversion, justification and padding/truncation of data fields is also performed. This utility also executes output message disposition (queuing/spooling), if requested.

- Edit Utility

This is a service routine called by the Monitor to process input messages, performing device-dependent transformations, and field validity checking, conversion and padding according to user-specified editing characteristics.

- Output Utility

This is a service routine executing as a subsystem to process output messages by performing device-dependent transformations, and then pass the messages to the Front End.

For detailed documentation of these facilities, see Message Mapping Utilities and the Utilities Users Guide.

Other service routines of the Intercomm system for processing requests associated with special subsystem design requirements are:

- Store/Fetch

This facility allows a subsystem to save and retrieve a temporary or permanent data string identified by a user-defined key. One or more subsystems can access each stored data string. (See Store/Fetch Facility.)

- Dynamic Data Queuing (DDQ)

This facility allows a subsystem to save and retrieve a set of related data strings (a data queue) identified by a user-defined name. One or more subsystems can access each DDQ which may be transient or permanent. A DDQ may also be used for collecting messages destined for another subsystem, a printer, or a batch program. (See Dynamic Data Queuing.)

- Table Facility

This facility provides for creating a temporary table with a unique user name in core storage above the 16M line. Table entry data strings may then be added, updated or retrieved by the same or another subsystem, and may have keys for sorting and retrieval as a user option. (See Table Facility.)

- CRT Page Facility

This facility allows a subsystem to write a set of output messages to a CRT terminal-oriented Page Data Set (Release 9) or to a table via the Table Facility (Release 10). The first message of a set is also sent to the Front End automatically. The terminal operator may then enter commands processed by the Paging subsystem to retrieve and browse through the pages of an output message set. (See Page Facility.)

- Data Base Management System Support (DBMS)

This facility consists of service routines for each supported DBMS (IDMS, ADABAS, TOTAL, DL/I) which allows access to the DBMS from Intercomm. (See Data Base Management System Users Guide.)

- Dynamic File Allocation (DFA)

This facility allows a subsystem to create (allocate) and/or access a sequential data set, or to access a VSAM data set, specifying its DSNAME as part of subsystem logic, rather than with execution JCL. (See Dynamic File Allocation.)

- Signed-on Operator-Id Checking

When executing under the security control of the Intercomm Extended Security System (ESS), a subsystem may call a service routine (SECUSER) to determine the user-ID of the operator at the terminal from which the transaction to be processed was entered. (See Extended Security System.)

- LU6.2 Support (See Section 2.2.3 and SNA LU6.2 Support Guide.)

2.4 SUBSYSTEMS

Intercomm-supplied subsystems are written in reentrant Assembler Language, and include the Output Utility, the Change/Display Utility, the Page Browsing Subsystem and many command processing subsystems.

The Output Utility allows a programmer to specify predefined report and display formats so that simply constructed output messages from a subsystem can be expanded, columnized, headed and subheaded, and displayed upon different types of devices without concern to the subsystem creating the message. Output Utility display formats can be changed without program modifications.

The Change/Display Utility allows simple inquiry and file maintenance via predefined keyword input messages from terminals causing access to data files defined by tables. The Display Utility is used in conjunction with the Output Utility to produce varied report or display formats.

The Page Facility processes commands from CRT-type terminals to browse through a series of output display screens created by the PAGE system program. Subsystems make use of this feature by calling the PAGE interface program during message processing. The terminal operator interacts with the Page Facility directly.

Command processing subsystems process Intercomm standard messages to accomplish the start/stop of system functions, message switching between terminals, displaying and changing the status of system control parameters, display of statistics, etc. The commands and text syntax are described in System Control Commands.

User-supplied subsystems accomplish application-dependent message processing. Each may call any Intercomm service routine or user-supplied subroutine, and may be written in COBOL, Assembler or PL/1.

2.4.1 Reentrant vs Nonreentrant Subsystems

In an interactive on-line environment, the probability is very high for more than one terminal operator to enter concurrent requests to be processed by the same subsystem. To accomplish the multithreading of concurrent requests, application subsystems should be coded as reentrant, that is, variable data is defined and processed in a dynamic working storage area (DWS) obtained for the exclusive use of one processing thread. Since COBOL does not allow the facility for dynamically obtaining a working storage area (no equivalent to Assembler GETMAIN/FREEMAIN processing), Intercomm provides an interface whereby COBOL subsystems may be coded as psuedo-reentrant so that multi-threading may be accomplished. A special interface to accomplish multi-threading in reentrant VS COBOL II programs is also provided. These interfaces and program coding requirements are described in Chapter 3.

2.5 INTERCOMM TABLES

Intercomm is a generalized on-line system monitor, requiring information about specific operating characteristics of a particular installation. This information is supplied in the form of tables generated with Intercomm macro instructions. Application programmers are usually not involved in defining the Intercomm tables, except for table specifications which pertain to their own applications. The basic tables controlling message processing are as follows:

- Front End Verb Table (BTVRBTB)

A table listing all valid transaction identifiers (verbs), and relating them to the subsystem required for message processing. There is one entry per verb, defined via a BTVRBTB macro.

- Front End Network Table

Tables describing the terminal network (relating individual devices to five-character station identifications), device hardware and operating characteristics, and output message queuing specifications.

- Back End Station Table (PMISTATB) and Device Table (PMIDEVTB)

Tables describing terminal identifications and device-dependent characteristics to the Message Mapping Utilities and/or the Edit and Output Utilities.

- System Parameter List (SPA)

A table describing system-wide operating characteristics. This table may be extended to include installation-defined table entries, accessible to all user subsystems and subroutines (see Chapter 8). This table is generated via the SPALIST macro.

- Data Set Control Table (DSCT)

A table generated by the File Handler describing on-line data sets. Information in this table is derived from JCL and file control (FAR) parameters at execution startup time.

- Subsystem Control Table (SCT)

A table listing the program properties (reentrancy, language, entry point, etc.), message queue specifications (core and/or disk queues), and scheduling (resident or loadable, concurrent message processing limits, priority, etc.) for each subsystem. There is one entry per subsystem, defined via a SYCTTBL macro.

The above listed tables are described in detail in the Operating Reference Manual. Additional tables describe detailed functions for the system programs, service routines and utilities.

2.6 INTERFACING WITH THE INTERCOMM MONITOR

Each message processed by Intercomm consists of a 42-byte header prefix, plus application-oriented message text. The message header is prefixed to each input message by the Front End and is analyzed by the System Monitor for all message processing control. The particular fields of the header which control message routing are Receiving Subsystem Code (MSGHRSC) and Receiving Subsystem Code High-Order (MSGHRSCH). This two-byte code is initialized by the teleprocessing interface when it constructs the header from the verb supplied at the beginning of the message text. The Front End Verb Table relates user verbs to their corresponding subsystem codes via coding of BTVRBT macros (see Basic System Macros) in a user member USRBTVRB copied into the system BTVRBTB which contains the Intercomm system verbs.

All subsystems are defined to Intercomm by an entry in the Subsystem Control Table (SCT). There is one entry for each subsystem which defines the program's general characteristics, scheduling requirements and message queuing specifications. Each subsystem must be assigned a unique two-character subsystem code for message routing. Definition of Intercomm system subsystems for utility and command processing is provided in the released member INTSCT.

The Subsystem Control Table entry for each user subsystem is defined using the SYCTTBL macro which is coded in a user member USRSCTS copied into the system INTSCT at assembly time. A full description of the macro may be found in the Intercomm Basic System Macros manual.

Many installations assign the responsibility of coding the Subsystem Control Table entries for individual user subsystems to the application programmer. At other installations, the Intercomm System Support Manager performs this task. In either case, the SYCTTBL macros must be coded with care, as there is one table controlling all user and system subsystems in operation when Intercomm is executing.

The most significant SYCTTBL macro parameters for COBOL subsystems are:

- LANG=RCOB

For reentrant COBOL subsystems (LANG=COB if nonreentrant).

- S BSP=xxxxxxx or LOADNAM=xxxxxxx (for dynamic load)

Specifies the subsystem entry, that is, the PROGRAM-ID of the COBOL subsystem (S BSP), or the load module name (LOADNAM).

- GET=nnnnn

Only meaningful if LANG=RCOB is coded. It specifies the amount of dynamic working storage (initialized to low-values) to be provided via the Linkage Section on entry to a reentrant COBOL subsystem. The amount specified may be up to 64K minus 308 bytes (which is used for a link/save prefix area, and for the DWS protection option described in Chapter 3).

- FREE=nnnnn

Only meaningful if LANG=RCOB is specified. The GET parameter must be specified to use FREE. It indicates the amount of the dynamic working storage area, provided on entry (via GET) to the reentrant COBOL subsystem, which should be freed when the subsystem completes. It defaults to the value specified for GET. (See Section 9.4 for further details.)

- TCTV=nnn

Expected maximum processing time (in seconds) in a high-volume environment before the subsystem is assumed to be looping, or in an extended wait for file or data base access, and should be timed out. Considerations for this value depend on subsystem processing such as data base access, file updates, number and type of file accesses, exclusive control for file updates, number of output messages created, enqueue lock-out possibilities, etc.

- MNCL=nn

Specifies the maximum number of concurrent threads that can be executed through this specific subsystem during a high activity period (when more than one operator enters transactions routed to this subsystem).

- RESOURC=name

This parameter is used to control concurrent access to a resource (file, table, data base, etc.) across several subsystems in one Intercomm region. The name is also coded for the ID parameter of a RESOURCE macro (coded before all SYCTTBLs in the SGT) which identifies the shared resource and the maximum concurrent subsystem threads that may be activated for that resource. Note that the maximum share count coded on the RESOURCE macro overrides the combined MNCL value for all the subsystems "naming" that resource. An internal enqueue is issued (no time-out). While using this feature will affect response time during peak activity, it does not affect the TCTV for a subsystem, which goes into effect after shared control of the resource is granted.

2.7 INTERCOMM MESSAGE HEADER

The Intercomm message header is constructed by the Front End for each message when it arrives from a terminal. New messages created within the subsystem must be prefixed with the standard forty-two-byte header format, which is constructed by copying the input message header to an output message area and then altering appropriate fields. Figure 7 lists the names and formats of all the fields in the message header, and describes their contents and changeability.

Field Name	Length	Description	Alter Legend*
MSGHLEN	2	Length of message including header (binary number)	Y
MSGHQPR	1	Teleprocessing segment I/O code: 02/F2=full message; 00/F0=header segment; 01/F1=intermediate segment 03/F3=final (trailer) segment	N
MSGHRSCH	1	High-order receiving subsystem code	Y
MSGHRSC	1	Low-order receiving subsystem code	Y
MSGHSSC	1	Low-order sending subsystem code	M
MSGHMMN	3	Monitor message number assigned by Message Collection (binary)	N
MSGHDAT	6	Julian date (YY.DDD)**	N
MSGHTIM	8	Time stamp (HHMMSSSTH)	N
MSGHTID	5	Terminal identification (originating terminal on input messages, destination terminal on output) or Broadcast Group name	Y
MSGHFLGS	2	Message indicator flags (MSGHCON - Rel 9)	N
(MSGHPID)	2	Reserved area (MSGHFLGS - Rel 9)	N
MSGHBMN	3	Front End message number - Rel 10 (binary)	N
MSGHSSCH	1	High-order sending subsystem code	M
MSGHUSR	1	User/system processing code***	L
	2	Used for special processing by the Front End (MSGHBMN - Rel 9)	N
MSGHLOG	1	Log code (see Figure 11)	L
MSGHBLK/ MSGHRETN	1	Reserved area/ Subsystem return code (for log code X'FA' entries only)	N
MSGHVMI	1	Verb or message identifier interpreted by receiving subsystem as required, and by FESEND	Y

Figure 7. Intercomm Message Header Fields (Page 1 of 2)

* Alter Legend:

Y = Must be filled in for intersubsystem message switching and output messages passed to FESEND (MSGHVMI should be set to X'57' or X'67', as appropriate, for output messages passed directly to FESEND)

M = Should be filled in for user's own information (required by Intercomm for message restart/file recovery and Log Analysis)

N = Do Not Touch (must be copied from input to output message header area)

L = May be modified for user codes based on subsystem logic

** The period represents a one-byte message thread number (for resource management and/or message restart purposes).

***MSGHUSR is used by Intercomm modules as follows:

1. If the BTVERB macro for the input verb has HPRTY=YES coded; contains a C'P' to request priority queuing for the subsystem. The user may move a C'P' to this field to request priority queuing for output messages to a terminal (via FESEND) or to another subsystem (via Message Collection).
2. For an input message from a BTAM 3270 CRT which contains SBA sequences, has a C'F' in the 01 log record.
3. For output messages to a switched async device (Teletype, Dataspeed 40, and 2740); a C'B' requests disconnect after transmitting the output message.
4. For output messages to a switched Teletype or Dataspeed 40 device; a C'X' requests using the alternate call-list for the next input message (as described in the BTAM Terminal Support Guide).
5. For output messages discarded by the Front End, a C'F' indicates the message was flushed by command, a C'Z' that it was discarded by the VTAM OTQUEUE user exit (Rel 10 only).

If none of the above considerations are applicable, the subsystem may use this field for messages queued to other user subsystems, or for special logging information. The LOGPRINT utility always prints the value coded in this field (in hexadecimal).

Figure 7. Intercomm Message Header Fields (Page 2 of 2)

2.7.1 MSGHQPR and MSGHVMI Fields

In general, a COBOL application subsystem does not need to be concerned with the MSGHQPR field, unless processing long input from a Teletype or similar device where message input may be segmented. In this case, the DDQ Facility must be used to store and sequentially forward the input message segments. Otherwise, input messages from the Front End always contain a QPR of C'2'. Both MMU and the Output Utility set the QPR to X'02' for output messages unless the Output Utility finds it necessary to segment an output message, in which case a segment code is used. The various uses of the MSGHVMI field for input and output message processing may be determined from the index references to this field at the end of this manual.

2.8 INTERCOMM MESSAGE FLOW USING MESSAGE MAPPING

The interaction of Intercomm system components, tables and subsystems with the Message Mapping Utilities (MMU) is summarized in Figure 8; the path of one input message and its corresponding output message is traced, and the numbered arrows in the diagram correspond to the numbered paragraphs below.

- 1 The Front End reads an input message and prefixes a 42-byte control header containing routing information, time, date, originating terminal and message length. The message is then queued for subsystem processing by Message Collection.
- 2 The System Monitor schedules the subsystem and retrieves the message based upon the Subsystem Control Table (SCT) scheduling criteria.
- 3 The message is passed to the subsystem.
- 4 Input in terminal-dependent format is transformed to a terminal independent form by a call to a Message Mapping Utility (MMU).
- 5 The subsystem performs message processing logic, requesting I/O service functions from the File Handler or Data Base Manager interface.
- 6 The subsystem creates one or more terminal-dependent output messages by calling MMU.
- 7 The subsystem passes the message formatted by MMU to the Front End by a call to FESEND (unless MMU is asked to perform this function).
- 8 The subsystem returns control to the System Monitor, passing a return code indicating normal completion or an error condition.

In the Intercomm multithread environment, this same sequence of events is carried out concurrently for many messages.

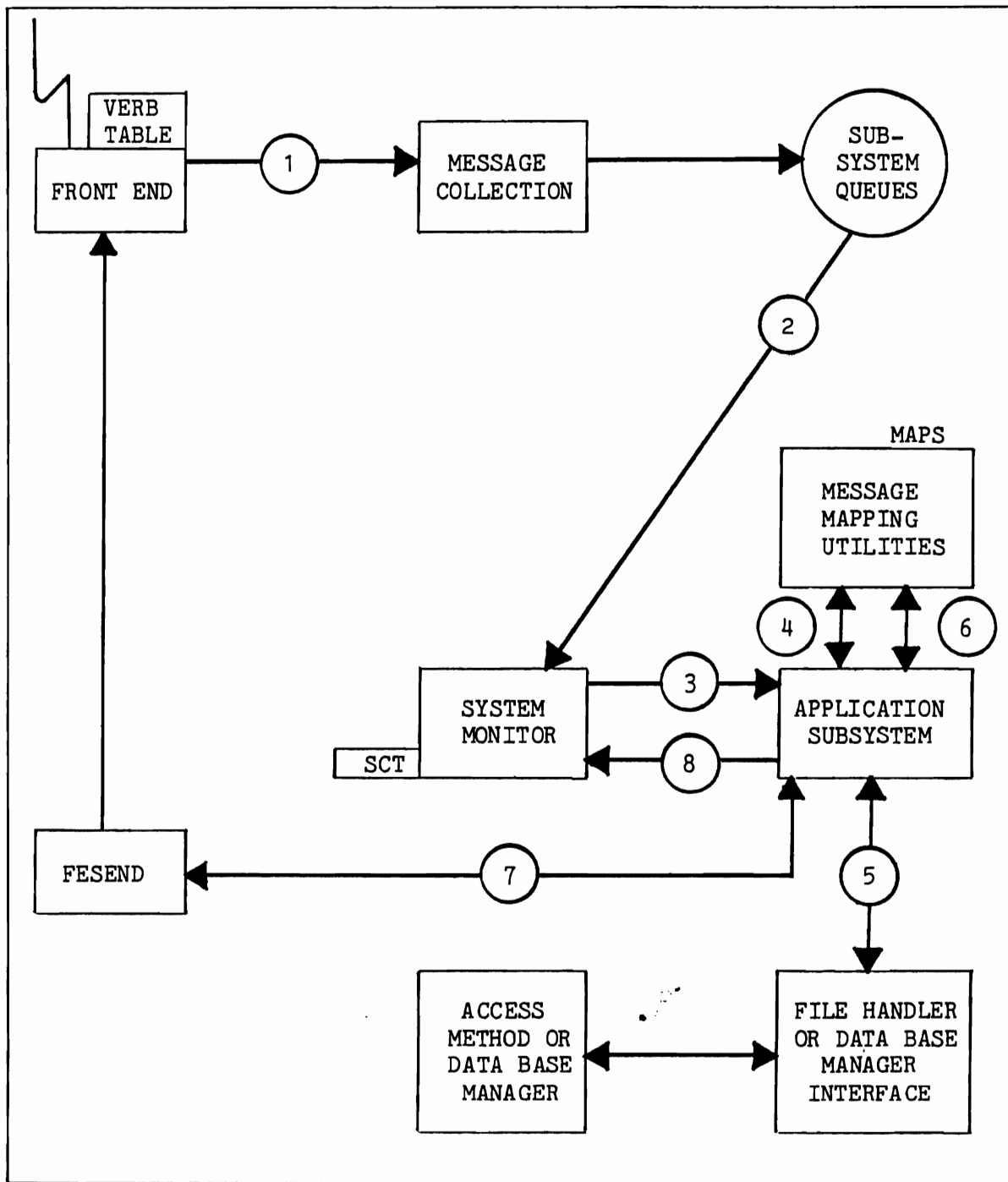


Figure 8. Intercomm Message Flow Using Message Mapping

2.9 INTERCOMM MESSAGE FLOW USING EDIT AND OUTPUT

The path of one input message and its corresponding output message is traced in Figure 9; the numbered arrows in the diagram correspond to the numbered paragraphs below.

- 1 The Front End reads an input message and prefixes a 42-byte control header containing routing information, time, date, originating terminal, and message length. The message is then queued for subsystem processing by Message Collection.
- 2 The System Monitor schedules the subsystem and retrieves the message based upon the Subsystem Control Table (SCT) scheduling criteria.
- 3 The Edit Utility is called (if required) and the input message is edited according to the Edit Control Table (ECT).
- 4 If Editing is not successful due to invalid input data, the Edit Utility optionally creates an error message for the originating terminal and queues it for the Output Utility by calling Message Collection. The subsystem is not activated.
- 5 If Editing is successful, the edited message is passed to the subsystem. If editing is not required, the unedited message is passed directly to the subsystem.
- 6 The subsystem performs message processing logic, requesting I/O service functions from the File Handler or Data Base Manager interface.
- 7 The subsystem creates one or more output messages and queues them for the Output Utility by calling Message Collection (COBPUT).
- 8 The subsystem returns control to the System Monitor, passing a return code indicating normal completion or an error condition.
- 9 The System Monitor schedules the Output Utility and passes the output message(s) to it for processing.
- 10 The Output Utility performs formatting, if specified in the message header, according to entries in the Output Format Table (OFT), finally passing the message to the Front End via a call to FESEND.
- 11 The Output Utility returns to the System Monitor.

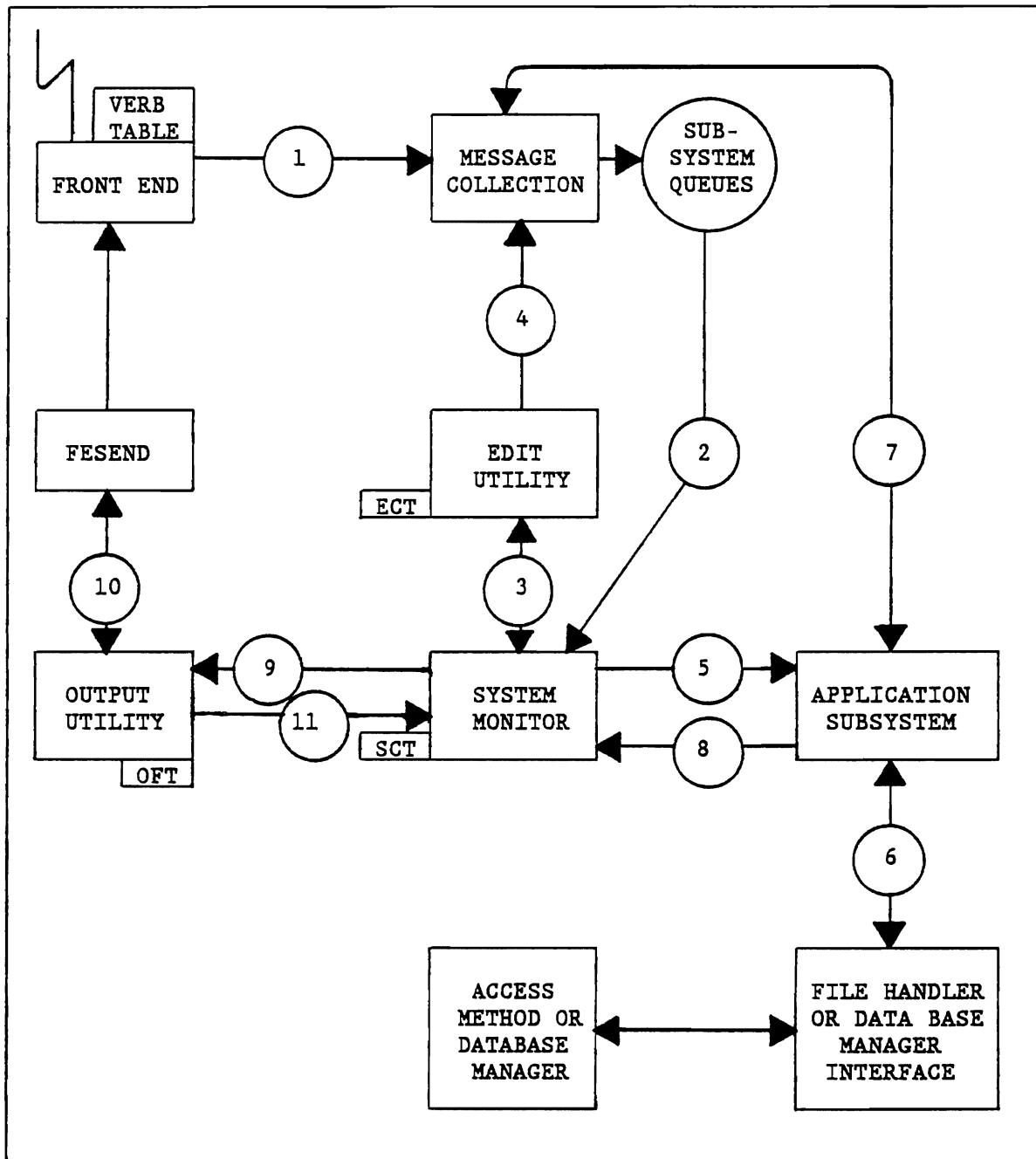


Figure 9. Intercomm Message Flow Using Edit and Output

2.10 THE INTERCOMM SYSTEM LOG

The Intercomm system log (INTERLOG) provides system journaling and maintains a historical record of all traffic within the system. Complete documentation of performance during on-line processing is thus provided, along with system control for restart/recovery.

Message traffic is recorded at the time of entry on a subsystem queue, and at the time message processing begins and ends within each subsystem. Subsystems may make user entries on the system log by calling an Intercomm system program (LOGPUT).

An installation may suppress some or all log entries, depending on its own requirements. The system log is optionally used at Intercomm system restart time to restore message traffic within the system at the time of failure. The logging entries are blocked and written to a variable-length sequential data set which may reside on disk or tape.

Log entries are in one of two formats: HT--42-byte message header and full text, as the message arrives from a terminal and is queued for a subsystem, or queued for a terminal; or HO--header-only entries, to mark progress through the system or error conditions.

Log entries are identified by a code in the MSGHLOG field of the message header. The time and date stamps (MSGHTIM and MSGHDAT) in the message header are updated for each log entry.

Progress of a message through a specific subsystem, or through the Front End, is indicated by the same Monitor Message Number (MSGHMMN) in each log record (01-30-FA or F2-F3). Complete progress of a message, from the first processing subsystem to final transmission, is indicated by the same Front End Message Number (MSGHBMN). The log may be printed completely or selectively via the Intercomm off-line utility LOGPRINT, described in the Operating Reference Manual.

A timing analysis utility (Log Analysis), which is supplied with Intercomm, may be used off-line to produce a report of message queuing and processing time. Statistics for messages by terminal, verb, subsystem, and/or system totals are provided. See the Operating Reference Manual.

The logging entries may be input to user-written batch programs to provide performance analysis in detail, such as traffic vs. network configurations, accounting routines, etc.

Figure 10 illustrates the log entries for one input message and a corresponding output message generated via the Output Utility. Number 6 appears only if executing in Test mode, since there is no Front End.

For live or simulated mode Intercomm, two additional entries are an F2 log code (HT) when the message is queued for the Front End via FESEND (appears in place of the 40 log entry between the 30 and FA entries), and an F3 log code (HO) when the message was transmitted by the Front End. Logging of the message to be transmitted (log code F2) occurs before final Front End processing (idles insertion, New Line to SBA sequence conversion, etc.).

If Message Mapping is used and the message is passed to the Front End via FESEND (Figure 8), only the log entries numbered 1, 2, and 4 appear for each message processing thread, with the FESEND log entry (log code 40 or F2) appearing in place of log entry 3. Log entries 3, 5, and 7 represent the additional processing for a message passed to the Output Utility (receiving code U).

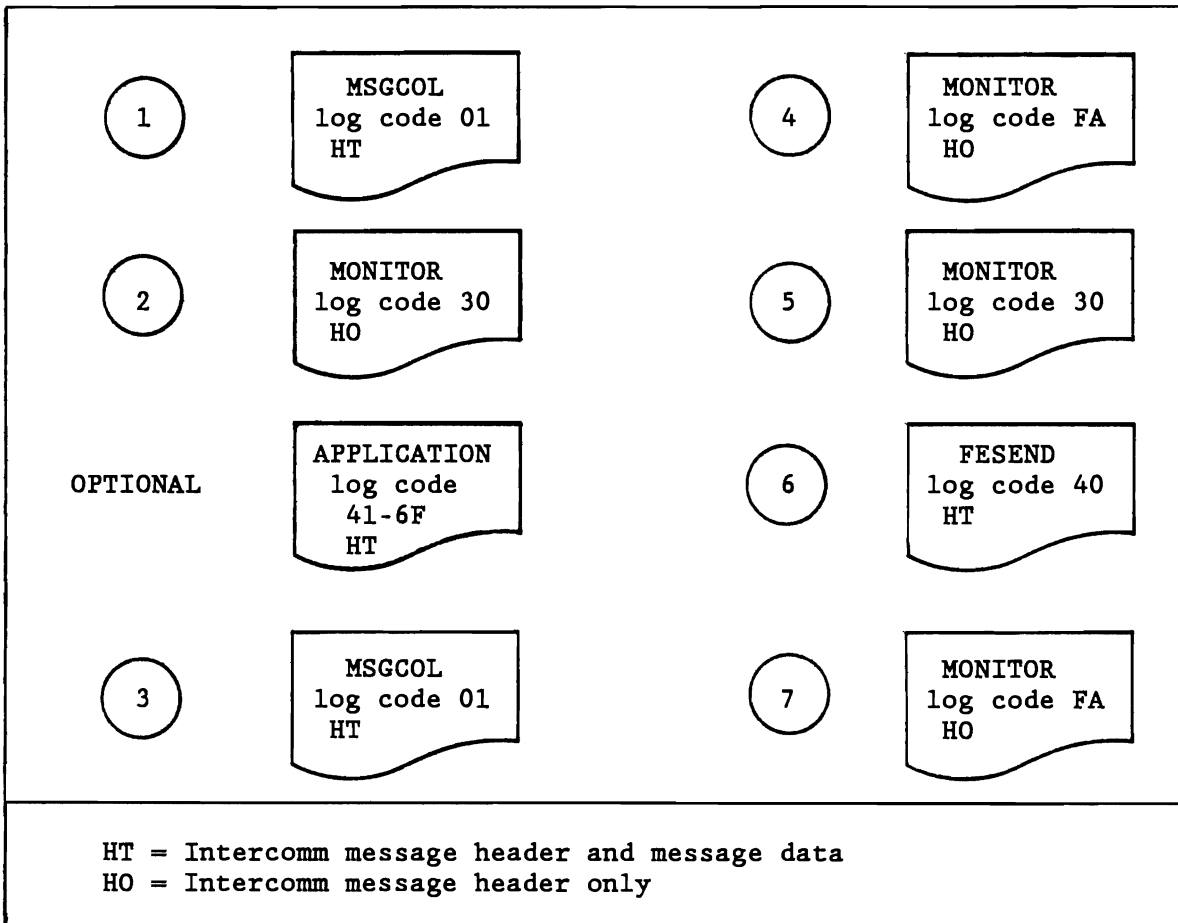


Figure 10. Sequence of Log Entries

Figure 11 describes all the Intercomm log codes. Note that user log entries may only use log codes in the range X'41' to X'6F'.

Internal Code	External Code	Format	Description	Origin	Restart Use
X'00'	00	HT	Checkpoint Record	Checkpoint	Yes
C'2'	01	HT	Message queued for subsystem by Front End or a subsystem	Message Collection	User
C'R'	02	HT	Message restarted through the system	LOGPROC	User
C'P'	03	HT	Message restarted--related to Data Base Recovery	LOGPROC	User
X'21-2F'	21-2F	HO/HT	See <u>SNA LU6.2 Support Guide</u>	Front End	No
C'T'	30	HO	Message passed to subsystem for processing	Subsystem Controller	User
C'Z'	40	HT	Message passed to Front End (test mode only)	FESEND	No
X'41'- X'6F'	41- 6F	HT	User called LOGPUT	Any Subsystem	No
X'80'- X'8E'	80- 8E	HT	File Recovery before-images	IXFLOG	User
X'8F'	8F	HO	Checkpoint Records indicator	IXFCHKPT	Yes
X'90'- X'9E'	90- 9E	HT	File Recovery after-images	IXFLOG	User
X'9F'	9F	HT	Intercomm Startup	LOGPUT	Yes
X'A0'	A0	HO	Message restart begun	LOGPROC	Yes
X'A1'	A1	HO	Message restart finished: all subsequent log entries produced by live Intercomm	LOGPROC	Yes
X'AA'	AA	HT	Intercomm Closedown	LOGPUT	No
X'CO'	C0	HT	Region started (Multiregion only) (Text=Region-id(s))	MRINTER	No
C'A'	C1	HT	Message successfully queued for Satellite Region	MRQMGR CR only	User

Internal Code: Log code in core during processing (snaps and dumps)
 External Code: Log code after translation by LOGPUT (INTERLOG printout)
 Format: HT for header and text, HO for header only
 Restart Use: Yes, No, User (specified via user-coded system macros)

Figure 11. INTERLOG Entries (Page 1 of 2)

Internal Code	External Code	Format	Description	Origin	Restart Use
C'B'	C2	HO	Message successfully passed to Satellite Region	MRQMNGR CR only	User
C'C'	C3	HO	Message lost (Region/Hold Q full) or flushed (SR/SS down)	MRQMNGR CR only	User
C'I'	C9	HT	Sign on/off processing, security violation messages	ESS	No
C'3'	FA	HO	Normal message complete	Subsystem Controller	User
C'5'	FB	HO	Unprocessed message--invalid subsystem/QPR code	Message Collection	User
C'6'	FC	HO	Unprocessed message--core and disk queue full	Message Collection	User
C'8'	FD	HO	Message cancelled--program error, time-out or I/O error; or flushed by command (Rel 10)	Subsystem Controller	User
C'9'	FE	HO	Message flushed by Retriever, used when application program does not obtain (via GETSEG) all parts of a segmented message; or message failed security check	Retriever SYCT400	No
C'1'	F1	HT	Message after verb verification	USRBTLOG (optional)	No
C'2'	F2	HT	Message queued for transmission	FSEND	User
C'3'	F3	HO	Message transmitted, discarded (MSGHUSR=Z-Rel 10), or flushed (MSGHUSR=F-Rel 10)	Front End	User
C'4'	F4	HO	3270 output message content invalid--message dropped.	BLHOT	No
C'5'- C'8'	F5- F8	HT	Transmitted DDQ msg status: see <u>SNA Term. Support Gd.</u>	Front End	No
X'FF'	FF	HT	Intercomm Restart Accounting	MSGAC	Yes

Figure 11. INTERLOG Entries (Page 2 of 2)

2.11 ADDITIONAL APPLICATION PROCESSING FACILITIES

In addition to the application programming facilities described in this and related manuals, the application designer should be aware of the following processing options available under Intercomm:

- Off-line batch region execution: the Intercomm File Handler, DFA, DDQ, Store/Fetch and MMU may be executed by an off-line program (coded as non-reentrant) to prepare a file, data strings, or messages for on-line access. See the associated manuals for linkedit considerations.
- Multiregion Facility batch region interface: when executing an on-line Multiregion system, any batch application region may pass a message or a FECMDDQ (see also Chapter 9) to an on-line subsystem or to the Front End via the Output Utility subsystem. See Multiregion Support Facility.
- Time controlled processing: instead of being triggered by an input terminal message, an application may be designed to execute at a particular time of day. See the Operating Reference Manual.
- Segmented input message processing via DDQ: segmented input messages, whether gathered by Intercomm from a remote device (CPU, etc.) or generated by an application program, are placed on a DDQ and may be serially passed to an application subsystem via a DDQ Facility interface. See Dynamic Data Queuing.
- Dynamic linkedit feature: dynamically loaded user subsystems and subroutines are linkedit to called Intercomm resident routines and COBOL support routines at startup, thus reducing the size of the load modules. The LOAD system control command is used to force a relinkedit of a new version of a dynamically loaded program placed on the load library while Intercomm is executing. See the Operating Reference Manual.
- User exits: various user exits for installation dependent processing are listed in the Operating Reference Manual.
- Binary table search: service routines for incore table searching are described in the Assembler Language Programmers Guide.
- IJKPRINT: service routine to write one or more print lines to SYSPRINT (SYSOUT data set). See the Operating Reference Manual.
- IJKDELAY: service routine to request a timed delay (averaging 100 milliseconds) of program processing, to allow other work (subsystem threads) to process. See the Operating Reference Manual.

CODING AN INTERCOMM SUBSYSTEM IN COBOL

3.1 PROGRAM STRUCTURE

An application subsystem executing under Intercomm control is activated to process one message. The following examples typify the concerns of message processing logic:

1. Interpretation of message text to reroute administrative data to another terminal.
2. Editing of message text, creation of a record on a sequential data set for later off-line processing and preparation of an acknowledgement message to the originating terminal.
3. Editing and analysis of message text to determine file retrieval and/or update criteria, data file access, preparation of a response message for the operator at the originating terminal.
4. Analysis of an application-oriented control message and appropriate action, such as checking batch totals from example 2, above, or acting on a special request to close a file or perform some other control function.

All subsystems are called by Intercomm and execute as subroutines with standard parameters passed on entry to the program. These parameters must be defined in the Linkage Section of the COBOL subsystem in the following order:

1. The input message to be processed (42-byte header plus message text) of maximum length 4096 bytes.
2. The System Parameter Area table (a 500-byte internal table plus appended user fields, if any), of maximum length 4096 bytes. Only the user fields may be modified, if desired.
3. The Subsystem Control Table entry for the called subsystem (a 100-byte table entry). This may not be modified.
4. A fullword computational field (PIC S9(8)) into which the subsystem must place an appropriate Intercomm return code before returning control to Intercomm.
5. The dynamic working storage area acquired by Intercomm for this reentrant subsystem to use (for all non-constant user and Intercomm-required fields) while processing a particular message thread. The size of the area obtained is specified by the subsystem's Subsystem Control Table entry (GET parameter). For nonreentrant COBOL and for FORTRAN subsystems, see Appendix E.

Figures 12 and 13 illustrate a reentrant COBOL subsystem with the Linkage Section initialized with the parameters described above for the Intercomm operating environment. A precise definition in the Linkage Section of the System Parameter Area (SPA) and Subsystem Control Table entry (SCT) is only required if these table areas are referenced by the subsystem during processing. Otherwise, an elementary 01 (PIC X) to be used as a parameter save space for the Procedure Division USING clause is sufficient. Note that the DWS area passed to the subsystem is that following a 256-byte Link/Save prefix used exclusively by the Intercomm interface routines.

```

ID DIVISION.
PROGRAM-ID. EXAMPLE1.
REMARKS. THIS IS A REENTRANT INTERCOMM COBOL SUBSYSTEM PROGRAM.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 CONSTANT-ITEMS          PICTURE X(8) VALUE 'CONSTANT'.
.
.   THESE ARE NEVER-CHANGING LITERAL VALUES...
.

LINKAGE SECTION.
01 INPUT-MESSAGE-AREA          PICTURE X(4096).
01 SPA                          PICTURE X(500).
01 SCT                          PICTURE X(100).
01 INTERCOMM-RETURN-CODE      PICTURE S9(8) COMPUTATIONAL.
01 DYNAMIC-WORK-SPACE.
02 OUTPUT-MESSAGE-AREA        PICTURE X(2000).
02 FILE-RECORD-AREA.
.
02 INDEPENDENT-ITEMS PICTURE S9(7)V99 COMPUTATIONAL-3.
.
.   ALL MODIFIABLE STORAGE MUST BE DEFINED HERE...
.
02 RETURN-VALUE PIC 99.
.
.   OTHER AREAS...
.

PROCEDURE DIVISION USING INPUT-MESSAGE-AREA, SPA, SCT,
INTERCOMM-RETURN-CODE, DYNAMIC-WORK-SPACE.
.
.                               PROGRAM PROCESSING LOGIC...
.
.   ...GO TO INTERCOMM.
.

INTERCOMM.
MOVE RETURN-VALUE TO INTERCOMM-RETURN-CODE.
GOBACK.

```

Figure 12. Reentrant COBOL Subsystem Structure

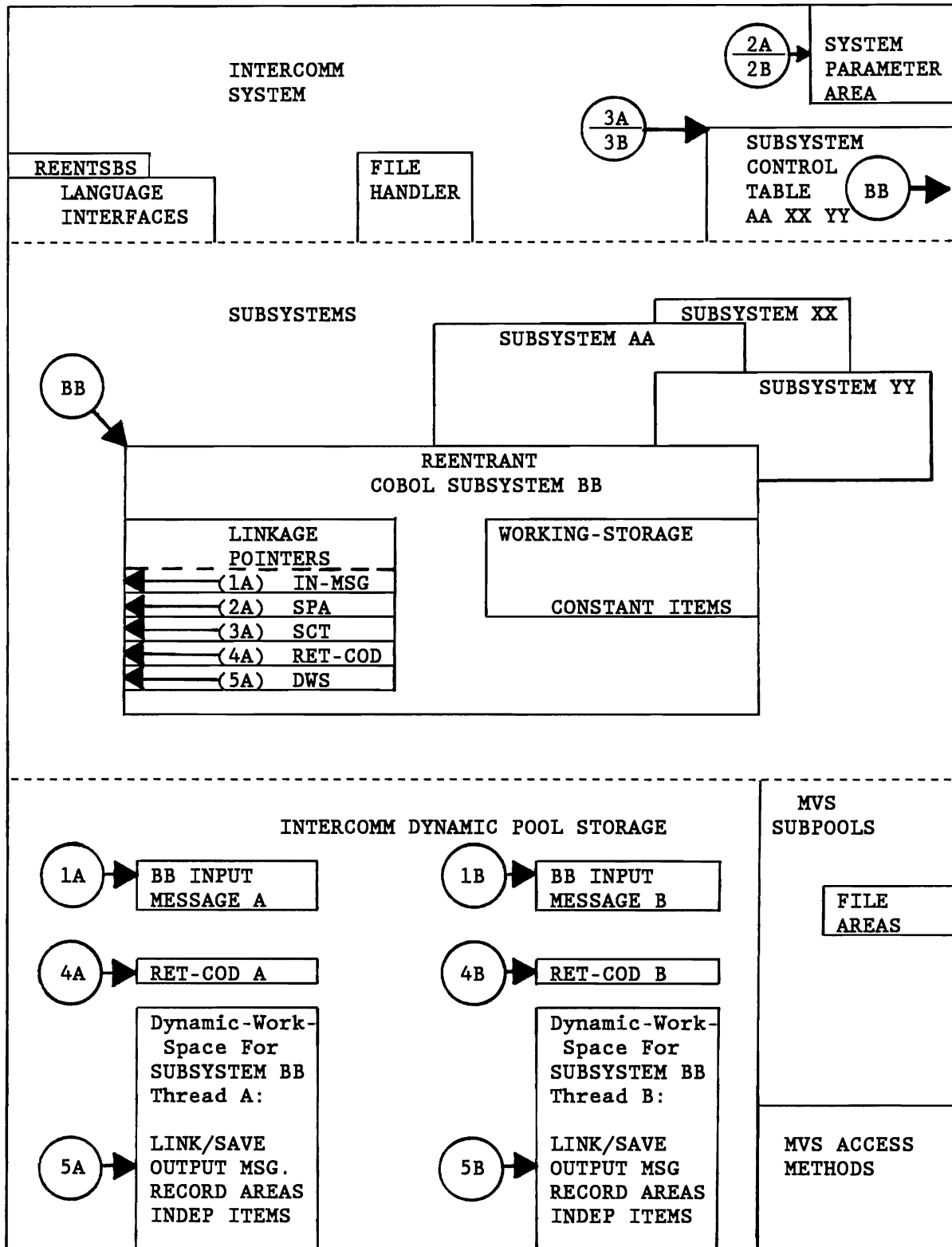


Figure 13. Reentrant Application Program Environment.

After a subsystem completes processing and returns control to the Subsystem Controller (see Chapter 2), the Intercomm return code is checked to determine whether the message should be cancelled due to an error. Then the return code is placed in the externally saved input message header in MSGHRETN (MSGHCON+1 for Rel 9), and the header is logged with an appropriate log code (see Chapter 2). Figure 14 describes Intercomm return codes. If the subsystem (or a called subroutine) program checks, or the return code is 8 or 12, USRCANC returns an appropriate error message to the terminal operator. USRCANC is a user exit provided by Intercomm under the name PMICANC, and is described in the Operating Reference Manual.

Return Code	Meaning	Subsystem Controller Error Action
0	Successful completion	None
4	Applies to Assembler Language subsystems only	
8	Unrecoverable error condition (no core, MAPEND error, etc.)	Message canceled, CALL to USRCANC
12	I/O error	Message canceled, CALL to USRCANC
16	(Reserved for internal use)	---
20-60	User codes to identify unusual condition	None
64	File or DBMS Update Subsystem, no message restart required*	None
68	File or DBMS Inquiry Subsystem, message restart required*	None
72-254	Same as 20-60	None
255	Reserved for MROTPUT	None
900**	Successful completion	None
912	Force Backout-on-the-Fly*	File updates or additions backed out
*See <u>File Recovery Users Guide</u> or <u>Data Base Management System Users Guide</u>		
**Used only when a called Assembler Language subroutine (MSGCOL/FESEND) has requested or freed the <u>input</u> message. If MAPIN has been called and has freed the input message, a return code of 0 must be used.		

Figure 14. Intercomm System Return Codes

3.2 MESSAGE PROCESSING CONCEPTS

The application program receiving the message may analyze the Verb Message Identifier (MSGHVMI) in the header and/or message text fields to further control message processing logic. The meaning of different VMI values is dependent on the design requirements of the program receiving the message. For example, the Front End sets the VMI to X'00' to indicate to the Subsystem Controller that editing by the Edit Utility is required, based on the specification in the Front End Verb Table for a given verb (BTVERB macro, EDIT parameter). The PREPROG interface routine then analyzes the VMI to determine if the Edit Utility should be called prior to passing the message to the subsystem (if editing is successful). A VMI value of X'FF' (high-values) indicates that no processing is required by, or was performed by, the Edit Utility. Any other value in the VMI indicates that the Edit Utility has already processed the message or that a user subsystem has placed a code in the field before switching (queuing) the message to the currently processing subsystem.

An application subsystem creates an output message by building a 42-byte header and appropriate message text. This new message is either passed to the Front End via FESEND for transmission to the terminal, or is queued for later processing by the Output Utility or some other subsystem by calling the Intercomm system program COBPUT. The subsystem destined to receive this new message is determined by the receiving subsystem code fields (MSGHRSC, MSGHRSCH) in the message header. The receiving subsystem may then analyze the VMI, as appropriate. The Output Utility, for example, analyzes the VMI to determine whether or not prespecified output message formatting is to be performed. If the output message is passed directly to FESEND, MSGHRSCH and MSGHRSC should be set to binary zeros (low-values).

Subsystem logic for input message text analysis and output message text creation varies, depending on whether Message Mapping or the Edit and Output Utilities are used. Figures 15 and 16 illustrate subsystem processing logic for these two cases.

It is very important to note that the input message area (Intercomm header and message text) may only be examined (treated as a read-only area) by the application program. It may also be copied to an output message area (header only, or header and text) where it may be added to or changed, depending on program logic. Never add data to the input message text area.

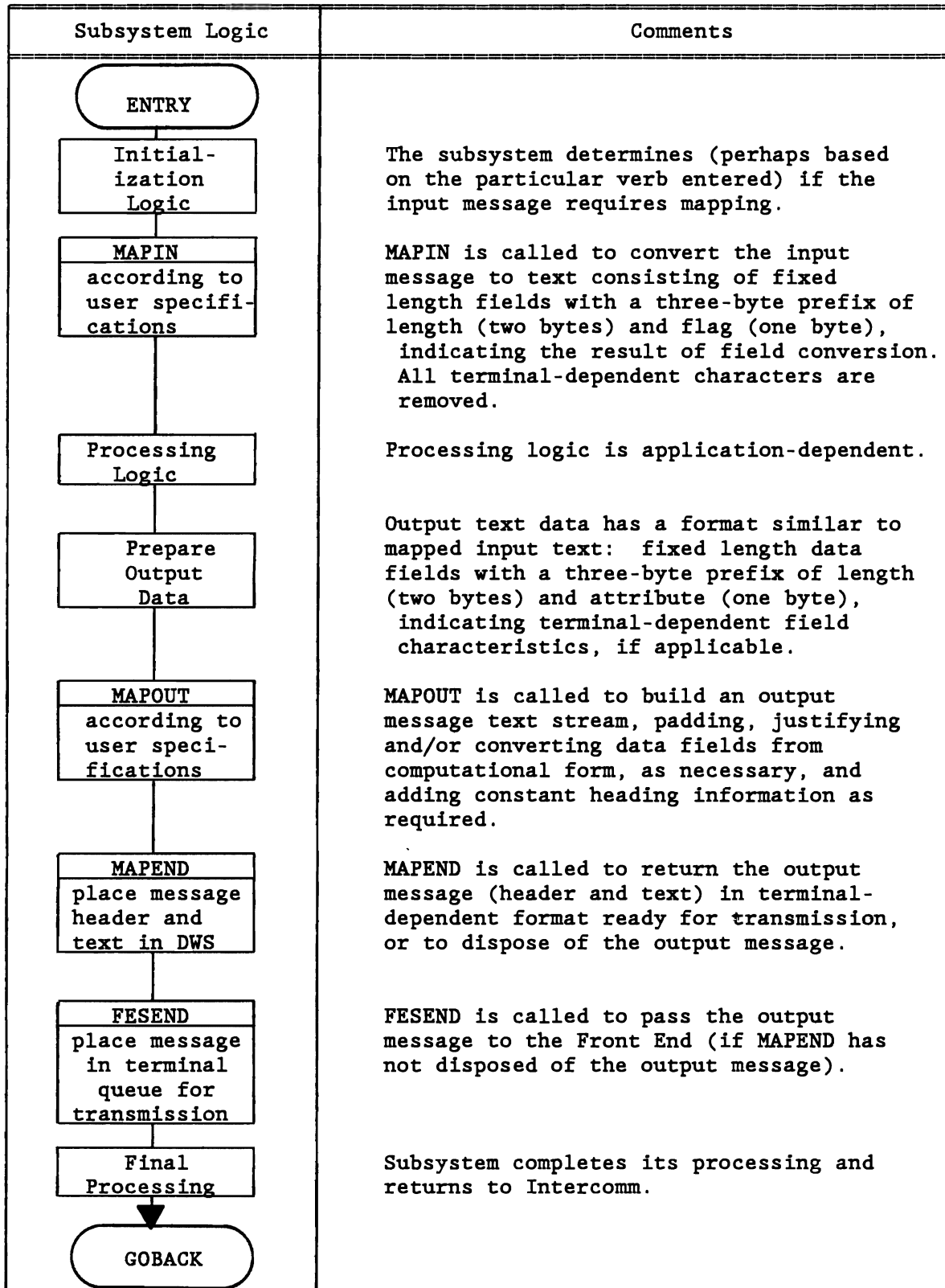


Figure 15. Subsystem Logic Using Message Mapping Utilities

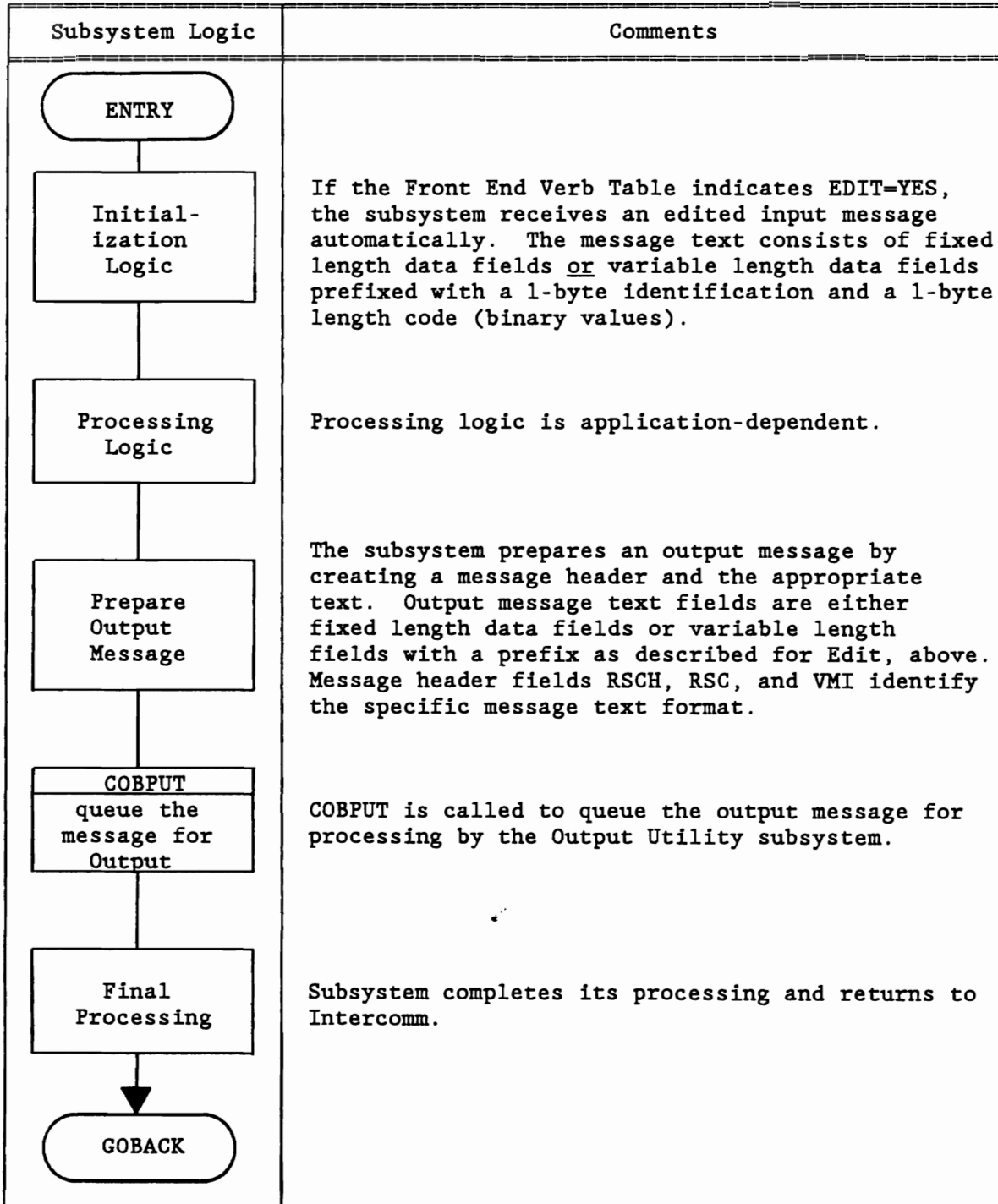


Figure 16. Subsystem Logic Using Edit and Output Utilities
(Page 1 of 2)

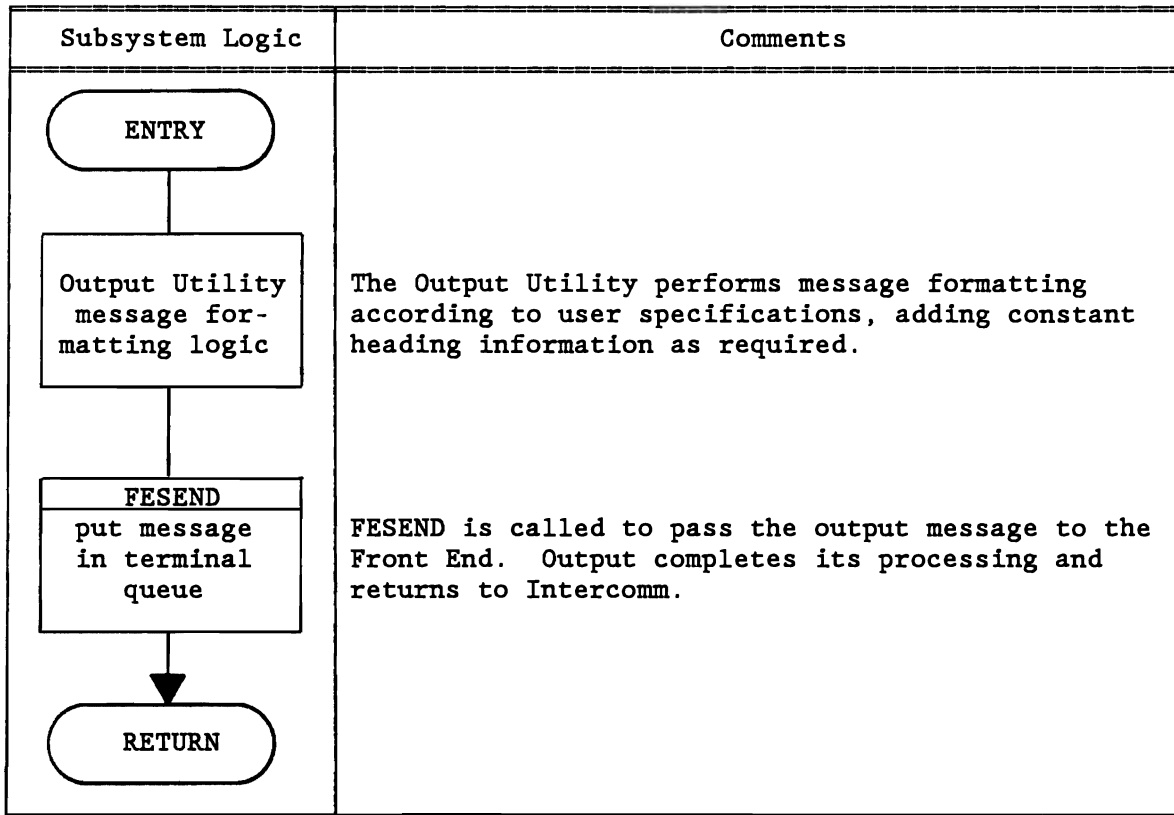


Figure 16. Subsystem Logic Using Edit and Output Utilities
(Page 2 of 2)

3.3 SUBSYSTEM CODING

The language interface routines are:

- PREPROG--which interfaces the Subsystem Controller to the COBOL subsystem by initializing the reentrant (if VS COBOL II) or pseudo-reentrant (if OS/VS or ANS COBOL) environment for each subsystem processing thread. If the VMI of the input message is X'00', the Edit Utility is called to edit the message. If successful, the subsystem is activated. If unsuccessful, EDIT returns an appropriate error message to the input terminal and PREPROG returns to the Subsystem Controller (subsystem not activated). If the subsystem is loaded above the 16M line, it will receive control in 31-Amode.
- COBREENT--which maintains linkages and save areas (and performs Amode switching) for COBOL subsystem interface to Intercomm service routines and for user subroutines, thus preserving the multithreaded reentrant environment while providing standard CALL interfaces to the routines. For VS COBOL II, COBREENT saves and restores the thread's run unit environment.

- COBPUT--which is called via COBREENT to copy a message from the dynamic working storage of a COBOL program into the Intercomm-managed dynamic pool storage area before passing it to Message Collection to be queued for another subsystem.
- REENTSBS--table of Intercomm service routine and user-coded subroutine entry points, names and related characteristics.

All COBOL subsystems and subroutines interface to Intercomm service routines and user subroutines using standard CALL 'literal' statements. Dynamic call is not supported. One routine only is called: COBREENT. The first passed parameter is the name of a code defining the actual routine to which interface is desired, subsequent parameters are those required by the called routine, and should be in Dynamic Working Storage if the subsystem (subroutine) can be loaded above the 16M line (must be a 24-bit address). Coding format:

```
CALL 'COBREENT' USING routine-code, parm1[, parm2,...].
```

Subsequent chapters of this manual and of related message processing facility manuals contain detailed descriptions of applicable routine-codes and the required parameters. The Intercomm source text member ICOMSBS, listed in Appendix B, provides the definition of the halfword routine-code constants (PIC 9(4) COMP) used for calling most of the Intercomm service routines via COBREENT. To ensure that the correct code value is used, ICOMSBS should be COPYed into the Working-Storage Section of each COBOL program as follows:

```
01 COBREENT-CODES COPY ICOMSBS.
```

Code names correspond to the entry point name defined in REENTSBS, and the code itself is an index value (offset) into the REENTSBS table (see Chapter 9).

Figure 17 illustrates the basic coding required to implement an Intercomm subsystem and the definition of an input message and creation of an output message via an application to "echo" the text of an incoming message back to the originating terminal. The Message Mapping Utilities, the Edit Utility and the formatting capabilities of the Output Utility are not used.

1. The message header is created by copying the input message header to the output message header area and adjusting the following fields:
 - MSGHSSCH, MSGHSSC--Sending Subsystem Code
Move in the original receiving subsystem code values, MSGHRSCH (to MSGHSSCH) and MSGHRSC (to MSGHSSC), to identify the current subsystem as the sending subsystem.

- MSGHRSCH, MSGHRSC--Receiving Subsystem Code
Move in a predefined code to indicate further processing (the next subsystem) for this message (for FESEND, use low-values).
 - MSGHVMI--Verb/Message Identifier
Move in a predefined code for subsystem processing, or to indicate to FESEND that the output message is not fully formatted, use X'57'. If an output message is formatted by MMU, do not touch this field.
 - MSGHLEN--Message Length
Modified to include header and text length of output message.
 - MSGHTID--Receiving Terminal Name
If the originating terminal is to receive the response message, do not change. Otherwise, specify the receiving terminal name for the output message(s).
2. The new message text is created by copying the input message text to the output text area, and then appending the author's name and a message ending character (X'26' or X'37').
 3. Queuing of the output message for the terminal is accomplished via the service routine FESEND (FESENDC).
 4. The return code from the queuing routine must be analyzed to assure that the new message was actually queued, and recovery action taken if not.
 5. The last logical activity in the subsystem is to move a value to the Intercomm return code field and GOBACK to the Subsystem Controller.

The program identification and entry point name must correspond to the subsystem entry point described in the Subsystem Control Table.

The input-message entry parameter defined in the Linkage Section has been further detailed to reference the 42-byte input message header and the input message text as separate entities. See Chapter 2 for a description of individual fields in the message header.

To assist the programmer in defining the message header, there are two source text members, ICOMINMG and ICOMDWS, listed in Appendix B; they may be COPYed into the appropriate portions of the Linkage Section. Additionally, the COPY member ICOMHEXC (for the Working-Storage Section) provides common hexadecimal codes.

The entry parameters for the System Parameter Area (SPA) and Subsystem Control Table (SCT) entry for the subsystem are not detailed as there is no need to reference any of their individual fields.

The entry parameter for the Intercomm return code is used to indicate the result of message processing to the Subsystem Controller.

The output message format appears as the first definitions in the Dynamic-Work-Space entry in the Linkage Section, which corresponds to the fifth entry parameter.

Constants are defined as usual in the Working-Storage Section. Independent items, that is, areas of storage modified during program execution, must be included in the Linkage Section as part of the Dynamic-Work-Space definition. Such items also include storage areas required for Intercomm service routines and passed to those routines as parameters, whether or not the subsystem references or modifies those areas. Additionally, variable areas passed as parameters to user subroutines must also be defined in the Dynamic-Work-Space. Unmodified constant values (map names, file DD names, etc.) may be defined in Working-Storage even though passed as parameter values to called routines, except if an OS/VS or ANS COBOL program is loaded above the 16M line (see Section 3.4.1). For VS COBOL II, variable items may be in Working-Storage (see Section 3.4.3).

3.3.1 Message Switching Between Subsystems

Any Intercomm subsystem may send a message to any other Intercomm subsystem. If a message is sent to some other subsystem, it is called "message switching." An application subsystem can switch a message to the Output Utility, which is another subsystem. The Change/Display Utility switches messages to the Output Utility. An application subsystem may switch (or requeue) a message to itself in the event that reprocessing or deferred processing of the message is required. An application subsystem may exceed an installation's core limitations and be broken into several subsystems. One subsystem may receive a message input from a terminal, perform partial processing and develop intermediate results in the form of a message sent to a second subsystem. The second subsystem processes the intermediate results as an input message and may complete the message processing or develop additional intermediate results in the form of messages sent or switched to any other subsystem or subsystems. Any one of these subsystems might also switch messages to the Output Utility.

Message switching between subsystems is accomplished by moving the input message header to an output message area, changing the receiving subsystem codes in the output header, adding (or copying) message text, and then calling COBPUT. The Verb/Message Identifier (MSGHVMI) may be initialized for interpretation by the receiving subsystem. A VMI equal to X'00' indicates that the Edit Utility is to be called by PREPROG prior to activating the subsystem.

To switch messages between terminals, the destination terminal identifier (MSGHTID), and the VMI, would also have to be changed before calling COBPUT or FESEND.

```

PP 5740-CB1 RELEASE 2.4                                IBM US/VS COBOL

      1                                1J.22.55          MAY 26,1944

00001 000010 ID DIVISION.
00002 000020 PROGRAM-ID. ECHOMSG.
00003 000030 REMARKS. THIS REENTRANT SUBSYSTEM ECHOS AN INPUT MESSAGE
00004 000040     CONTAINING UP TO 500 CHARACTERS OF TEXT BACK TO THE
00005 000050     ORIGINATING TERMINAL.
00006 000060     IT COPIES THE INPUT TO THE OUTPUT MESSAGE AREA,
00007 000070     MODIFIES THE MESSAGE HEADER, APPENDS THE AUTHOR'S NAME,
00008 000080     AND MESSAGE ENDING CHARACTER, BEFORE CALLING FESENDC TO
00009 000090     QUEUE THE MESSAGE FOR THE INPUT TERMINAL.
00010 000100 ENVIRONMENT DIVISION.
00011 000110 DATA DIVISION.
00012 000120 WORKING-STORAGE SECTION.
00013 000130 01 HEX-CODES COPY ICMHEXC.
00014 C      01 HEX-CODES.
00015 C          05 HEX-00     PIC X VALUE ' '.
00016 C          05 CODE-00   REDEFINES HEX-00   PIC X.
00017 C          05 HEX-15     PIC X VALUE ' '.
00018 C          05 CODE-21   REDEFINES HEX-15   PIC X.
00019 C          05 HEX-37     PIC X VALUE ' '.
00020 C          05 CODE-55   REDEFINES HEX-37   PIC X.
00021 C          05 HEX-50     PIC X VALUE 'E'.
00022 C          05 CODE-80   REDEFINES HEX-50   PIC X.
00023 C          05 HEX-51     PIC X VALUE ' '.
00024 C          05 CODE-81   REDEFINES HEX-51   PIC X.
00025 C          05 HEX-52     PIC X VALUE ' '.
00026 C          05 CODE-82   REDEFINES HEX-52   PIC X.
00027 C          05 HEX-53     PIC X VALUE ' '.
00028 C          05 CODE-83   REDEFINES HEX-53   PIC X.
00029 C          05 HEX-54     PIC X VALUE ' '.
00030 C          05 CODE-84   REDEFINES HEX-54   PIC X.
00031 C          05 HEX-55     PIC X VALUE ' '.
00032 C          05 CODE-85   REDEFINES HEX-55   PIC X.
00033 C          05 HEX-56     PIC X VALUE ' '.
00034 C          05 CODE-86   REDEFINES HEX-56   PIC X.
00035 C          05 HEX-57     PIC X VALUE ' '.
00036 C          05 CODE-87   REDEFINES HEX-57   PIC X.
00037 C          05 HEX-72     PIC X VALUE ' '.
00038 C          05 CODE-114  REDEFINES HEX-72   PIC X.
00039 C          05 HEX-FF     PIC X VALUE ' '.
00040 C          05 CODE-255  REDEFINES HEX-FF   PIC X.

00042 000140 01 REENTSBS-CODES COPY ICUMSBS.
00043 C      01 REENTSBS-CODES.
00044 C          * THESE CODES REPRESENT OFFSETS FOR ROUTINE ADDRESSES IN THE
00045 C          * TABLE NAMED REENTSBS. ONLY THE MOST COMMONLY USED VALUES
00046 C          * ARE INCLUDED HERE; THE USERS MANUAL HAS A COMPLETE LIST.
00047 C          * IF OFFSET ODD, THEN TRUE OFFSET=(OFFSET+1)
00048 C          05 INTSORTC   PIC 99 COMP VALUE 99.

```

Figure 17. Echo Message Example; Reentrant COBOL
(Page 1 of 6)

2	ECHOMSG	15.22.55	MAY 26,1944
00049 C	05	DWS-SNAP	PIC 99 COMP VALUE 95.
00050 C	05	MAPFREE	PIC 99 COMP VALUE 91.
00051 C	05	FECMRLSE	PIC 99 COMP VALUE 87.
00052 C	05	FESEND	PIC 99 COMP VALUE 83.
00053 C	05	FESENOC	PIC 99 COMP VALUE 79.
00054 C	05	DYN-ALLOCATE	PIC 99 COMP VALUE 75.
00055 C	05	DYN-ACCESS	PIC 99 COMP VALUE 71.
00056 C	05	MAPURGE	PIC 99 COMP VALUE 67.
00057 C	05	MAPCLR	PIC 99 COMP VALUE 63.
00058 C	05	MAPEND	PIC 99 COMP VALUE 59.
00059 C	05	MAPOUT	PIC 99 COMP VALUE 55.
00060 C	05	MAPIN	PIC 99 COMP VALUE 51.
00061 C	05	INTUNSTO	PIC 99 COMP VALUE 47.
00062 C	05	INTSTORF	PIC 99 COMP VALUE 43.
00063 C	05	INTFETCH	PIC 99 COMP VALUE 39.
00064 C	05	FECMFD8K	PIC 99 COMP VALUE 35.
00065 C	05	FECMDDQ	PIC 99 COMP VALUE 31.
00066 C	05	DQ-WRITEX	PIC 99 COMP VALUE 27.
00067 C	05	DQ-READX	PIC 99 COMP VALUE 23.
00068 C	05	DQ-WRITE	PIC 99 COMP VALUE 19.
00069 C	05	DQ-READ	PIC 99 COMP VALUE 15.
00070 C	05	DQ-CLOSE	PIC 99 COMP VALUE 11.
00071 C	05	DQ-OPEN	PIC 99 COMP VALUE 07.
00072 C	05	DQ-BUILD	PIC 99 COMP VALUE 03.
00073 C	05	FH-SELECT	PIC 99 COMP VALUE 4.
00074 C	05	FH-RELEASE	PIC 99 COMP VALUE 8.
00075 C	05	FH-READ	PIC 99 COMP VALUE 12.
00076 C	05	FH-WRITE	PIC 99 COMP VALUE 16.
00077 C	05	FH-GET	PIC 99 COMP VALUE 20.
00078 C	05	FH-PUT	PIC 99 COMP VALUE 24.
00079 C	05	FH-RELEX	PIC 99 COMP VALUE 28.
00080 C	05	FH-FEQY	PIC 99 COMP VALUE 32.
00081 C	05	COBPUT	PIC 99 COMP VALUE 68.
00082 C	05	MSGCOL	PIC 99 COMP VALUE 72.
00083 C	05	COBSTORF	PIC 99 COMP VALUE 76.
00084 C	05	CONVERSE	PIC 99 COMP VALUE 80.
00085 C	05	DBINT	PIC 99 COMP VALUE 84.
00086 C	05	LOGPUT	PIC 99 COMP VALUE 88.
00087 C	05	PAGE-FILE	PIC 99 COMP VALUE 92.
00088 C	05	FH-GETV	PIC 99 COMP VALUE 96.
00089 C	05	FH-PUTV	PIC 999 COMP VALUE 100.
00090 C	*	CODES 104 AND UP INDICATE USER ADDITIONS TO THE TABLE	
00092	000150 01	AUTHORS-NAME.	
00093	000160	04	OUT-NAME PIC X(10) VALUE ' T.ELGUERA'.
00094	000170	04	OUT-MSG REDEFINES OUT-NAME.
00095	000180	06	NAME-CHAR PIC X OCCURS 10 TIMES.

Figure 17. Echo Message Example; Reentrant COBOL
(Page 2 of 6)

```

3          ECHOMSG          15.22.55          MAY 26,1994

00097      000190 LINKAGE SECTION.
00098      000200 01 INPUT-MESSAGE COPY ICOMINMG.
00099 C          01 INPUT-MESSAGE.
00100 C          04 MESSG-HDR.
00101 C          06 MSGH-LENGTH          PIC S9999  COMP.
00102 C          06 MSGH-QPR            PIC X.
00103 C          06 MSGH-RSCH          PIC X.
00104 C          06 MSGH-RSC          PIC X.
00105 C          06 MSGH-SSC          PIC X.
00106 C          06 MSGH-MMN          PIC XXX.
00107 C          06 MSGH-DATE.
00108 C          08 MSGH-YR            PIC 99.
00109 C          08 MSGH-PERIOD        PIC X.
00110 C          08 MSGH-JULIAN-DAY    PIC 999.
00111 C          06 MSGH-TIME.
00112 C          08 MSGH-HH            PIC 99.
00113 C          08 MSGH-MM            PIC 99.
00114 C          08 MSGH-SS            PIC 99.
00115 C          08 MSGH-TM            PIC 99.
00116 C          06 MSGH-TID.
00117 C          08 MSGH-T11           PIC X.
00118 C          08 MSGH-T12-3         PIC XX.
00119 C          08 MSGH-T14-5         PIC 99.
00120 C          06 MSGH-FLGS          PIC X(2).
00121 C          06 MSGH-PID           PIC X(5).
00122 C          06 MSGH-PIDX REDEFINES MSGH-PID.
00123 C          08 FILLER             PIC X(2).
00124 C          08 MSGH-BMN           PIC X(3).
00125 C          06 MSGH-SSCH          PIC X.
00126 C          06 MSGH-ADDR          PIC X(3).
00127 C          06 MSGH-ADRX REDEFINES MSGH-ADDR.
00128 C          08 MSGH-USR           PIC X.
00129 C          08 FILLER             PIC X(2).
00130 C          06 MSGH-LOG           PIC X.
00131 C          06 MSGH-BLK           PIC X.
00132 C          06 MSGH-VMI           PIC X.

00134      000210          04 INPUT-MESSAGE-TEXT          PIC X          OCCURS 500 TIMES.
00135      000220 01 SYSTEM-PARAMETER-TABLE          PIC X.
00136      000230 01 SUBSYSTEM-CONTROL-TABLE          PIC X.
00137      000240 01 INTERCOMM-RET-CODE          PIC S9(7) COMPUTATIONAL.
00138      000250 01 DYNAMIC-WORK-SPACE          COPY ICOMDWS.
00139 C          01 DYNAMIC-WORK-SPACE.
00140 C          02 OUTPUT-MESSAGE.
00141 C          04 OMESSG-HDR.
00142 C          06 OMSGH-LENGTH          PIC S9999  COMP.
00143 C          06 OMSGH-QPR            PIC X.
00144 C          06 OMSGH-RSCH          PIC X.
00145 C          06 OMSGH-RSC          PIC X.
00146 C          06 OMSGH-SSC          PIC X.
00147 C          06 OMSGH-MMN          PIC XXX.
00148 C          06 OMSGH-DATE.

```

Figure 17. Echo Message Example; Reentrant COBOL
(Page 3 of 6)

```

4          ECHOMSG          15.22.55          MAY 26,1994

00149 C          08  UMSGH-YR          PIC 99.
00150 C          08  UMSGH-PERIOD      PIC X.
00151 C          08  UMSGH-JULIAN-DAY  PIC 999.
00152 C          06  UMSGH-TIME.
00153 C          08  UMSGH-MM          PIC 99.
00154 C          08  UMSGH-MM          PIC 99.
00155 C          08  UMSGH-SS          PIC 99.
00156 C          08  UMSGH-TH          PIC 99.
00157 C          06  UMSGH-TID.
00158 C          08  UMSGH-TI1         PIC X.
00159 C          08  UMSGH-TI2-3       PIC XX.
00160 C          08  UMSGH-TI4-5       PIC 99.
00161 C          06  UMSGH-FLGS        PIC X(2).
00162 C          06  UMSGH-PID         PIC X(5).
00163 C          06  UMSGH-PIDX REDEFINES UMSGH-PID.
00164 C          08  FILLER            PIC X(2).
00165 C          08  UMSGH-BMN         PIC X(3).
00166 C          06  UMSGH-SSCH        PIC X.
00167 C          06  UMSGH-ADDR        PIC X(3).
00168 C          06  UMSGH-ADRX REDEFINES UMSGH-ADDR.
00169 C          08  UMSGH-USR        PIC X.
00170 C          08  FILLER            PIC X(2).
00171 C          06  UMSGH-LOG         PIC X.
00172 C          06  UMSGH-BLK        PIC X.
00173 C          06  UMSGH-VMI        PIC X.

00175 000260     04  OUTPUT-MESSAGE-TEXT PIC X OCCURS 510 TIMES.
00176 000270     02  FESENDC-RETURN-CODE PIC 99.
00177 000280     88  QUEUED VALUE ZERO.
00178 000290     02  I PIC S9(4) COMPUTATIONAL.
00179 000300     02  J PIC S9(3) COMPUTATIONAL.

```

Figure 17. Echo Message Example; Reentrant COBOL
(Page 4 of 6)

```

5          ECHUMSG          15.22.55          MAY 26,1994

00181    000310  PROCEDURE DIVISION USING
00182    000320                                INPUT-MESSAGE
00183    000330                                SYSTEM-PARAMETER-TABLE
00184    000340                                SUBSYSTEM-CONTROL-TABLE
00185    000350                                INTERCOMM-RET-CODE
00186    000360                                DYNAMIC-WORK-SPACE.
00187    000370      MOVE MESSG-HDR TO UMESSG-HDR.
00188    000380      MOVE OMSGH-RSCH TO OMSGH-SSCH.
00189    000390      MOVE OMSGH-RSC TO OMSGH-SSC.
00190    000400      MOVE LOW-VALUES TO OMSGH-RSCH.
00191    000410      MOVE LOW-VALUES TO OMSGH-RSC.
00192    000420      MOVE HEX-57 TO OMSGH-VMI.
00193    000430      PERFORM MOVE-A-CHARACTER VARYING I FROM +1 BY +1
00194    000440                                UNTIL I IS EQUAL TO MSGH-LENGTH - 42.
00195    000450      PERFORM NAME-MOVE VARYING J FROM +1 BY +1 UNTIL J > +10.
00196    000460      MOVE HEX-37 TO OUTPUT-MESSAGE-TEXT (I).
00197    000470      COMPUTE OMSGH-LENGTH = I + 42.
00198    000480      CALL 'COBREENT' USING
00199    000490                                FESENDC
00200    000500                                OUTPUT-MESSAGE
00201    000510                                FESENDC-RETURN-CODE.
00202    000520      IF NOT QUEUED
00203    000530                                MOVE FESENDC-RETURN-CODE TO INTERCOMM-RET-CODE
00204    000540      ELSE
00205    000550                                MOVE ZEROS TO INTERCOMM-RET-CODE.
00206    000560      GOBACK.
00207    000570  SUBROUTINE SECTION.
00208    000580  MOVE-A-CHARACTER.
00209    000590      MOVE INPUT-MESSAGE-TEXT (I) TO OUTPUT-MESSAGE-TEXT (I).
00210    000600  NAME-MOVE.
00211    000610      MOVE NAME-CHAR (J) TO OUTPUT-MESSAGE-TEXT (I).
00212    000620      COMPUTE I = I + 1.

```

Figure 17. Echo Message Example; Reentrant COBOL
(Page 5 of 6)

3.4 REENTRANT CODING CONVENTIONS

When coding a reentrant COBOL subsystem (or subroutine), care must be taken to observe the following conventions:

1. Never use the ALTER verb.
2. Never modify the WORKING-STORAGE SECTION.
3. Define in the Linkage Section all areas modifiable during program execution as items subordinate to the fifth 01 (Dynamic-Work-Space) entry.
4. Call all system service routines, data base service routines, and user subroutines through the COBREENT interface program in order to maintain reentrancy and the multithreading environment.
5. A call to COBREENT within a PERFORM range, although permissible, must be treated as a branch (GO TO) out of the PERFORM range. This restriction is easily met if all PERFORM ranges are accessed only by PERFORM statements. Never GO TO a performed paragraph.
6. Verify that the amount of dynamic storage defined for the subsystem in the Subsystem Control Table is an exact multiple of 8 and is the same or greater than the number of characters shown in the DMAP and defined for the fifth 01 (DWS) entry in the Linkage Section (see Figure 17).
7. FDs cannot be used, nor any file access verbs (OPEN, READ, etc.).
8. Do not use the reserved COBOL data name RETURN-CODE for any Intercomm return codes. (That data name refers to the contents of general register 15, which is not used for this purpose by the interface programs.)
9. Do not forget to code GOBACK to exit to Intercomm; otherwise a User 519 abend will result.
10. Ensure numeric fields are not zero before executing a DIVIDE or COMPUTE.

3.4.1 XA/ESA Extended Storage Loading Requirements (Release 10 only)

COBOL subsystems and subroutines using Intercomm reentrant coding conventions are eligible for loading above the 16M line if these recommendations are followed:

- The module should be linkedited with the AMODE=31,RMODE=ANY and REUS (RENT if VS COBOL II) parameters (see Appendix A)

- For subsystems, the LOADNAM, LANG=RCOB, BLDL=YES (default), and REUSE=YES (default) parameters are required on the SYCTTBL macro (a loaded subsystem remains in extended storage except when necessary to delete it after a program check, time-out, or by user system control command request)
- For subroutines, the LNAME, TYPE=COBOL, BLDL=YES (default) and USAGE=REENT parameters are required on the SUBMODS macro defining the subroutine to Intercomm (see also Section 9.7)
- Ensure that the Intercomm interface routines PREPROG, COBREENT and DYNLOAD (for loaded subroutines) were reassembled under XA or ESA (with the XA global on in the Intercomm global table SETGLOBE if at SM level 2240 or lower)
- All parameters (except the ICOMSBS code) passed via calls to COBREENT must be in 24-Amode storage (DWS). Constants (file names, map names, etc.) must be moved to the DWS before the call, except if VS COBOL II (see below).

3.4.2 Dynamic Working Storage (DWS) Protection Option

Destruction of Intercomm storage pool areas can result if the Dynamic Working Storage (DWS) acquired for a COBOL subsystem or subroutine is too small. This user option causes Intercomm to allocate extra space at the end of the DWS for each reentrant program. When the subsystem or subroutine calls COBREENT, this space is checked to see if it has been modified. If so, then the DWS is too small and the thread is terminated with a program check (Snap 126). An error message is sent to the terminal operator, and the Intercomm control terminal.

This protection option applies only to reentrant COBOL subsystems with an equal value of GET and FREE specified on the SYCTTBL macro instruction, and to all reentrant COBOL subroutines with the GET parameter specified on the SUBMODS macro. This option cannot detect the possibility of storage destruction beyond the extra DWS area. Because of the processing overhead required for this feature, it should be used only until subsystems are thoroughly tested.

The DWS protection option is requested system-wide via the DWSCHK parameter on the SPALIST macro at system generation time. The option may also be dynamically controlled system-wide by the Intercomm STRT and STOP system control commands which control activation and deactivation of various system control and debugging features. In addition under Release 10, for individual subsystems, the option may be requested via the DWSCHK parameter on the SYCTTBL macro describing the subsystem. If YES is coded, and the option is active system-wide, DWS checking will be performed. If NO is coded, then it will not be performed even though active for the system. Conversely, if not active for the system, SYCTTBL coding is ignored.

3.4.3 VS COBOL II Program Conversion and Support (Release 10 only)

- Compiler options required for Intercomm programs are:

RENT, RES and NODYNAM
DATA(24)
NOTEST and NOFDUMP
LIB

The following compiler options are recommended:

APOST (single quote for literals and CALLs)
TRUNC(BIN)
COMPILE (if not the default)
NONAME and NOTERM
SSRANGE (if supported for Intercomm COBOL II installation)
LIST and NOOFFSET

All other compiler options are dependent on site standards or programmer specification, and are explained in detail in IBM's VS COBOL II Application Programming Guide For MVS. See the sample Compile JCL in Appendix A for compiler parms specification.

Note that use of the OPTIMIZE option may enhance performance, but will increase the size of the COBOL load module as all the PERFORMed paragraph code is generated inline in the object (assembler) code immediately following the PERFORM statement. If a paragraph is PERFORMed multiple times, then multiple copies of the paragraph code are generated in the object code. While this may not affect seldom executed loadable programs (or programs loaded above the 16M line), it will greatly increase the size of the Intercomm load module for resident subsystems and subroutines.

- Coding Requirements and Options:

GOBACK (not STOP RUN) must be used to return to Intercomm.

Dynamic Working Storage (GET parameter on SYCTTBL or SUBMODS) must be a minimum of 8 bytes. Under VS COBOL II, variable (modifiable) fields may be in the 'WORKING-STORAGE SECTION'. They (and VALUE fields) do not have to be copied from Working-Storage to the DWS area, even if the program is loaded above the 16M line. Due to the DATA(24) compiler option requirement, no coding changes are needed to have a reentrant program dynamically loaded above the 16M line. If loadable above the line, storage will be saved in the Intercomm Address Space, and the only requirement is the added AMODE=31,RMODE=ANY linkedit parameters (see sample Compile and Link JCL in Appendix A). VS COBOL II subroutines must also have a DWS of at least 8 bytes, and should define all variable fields in that DWS if the subroutine may be called more than once within one message processing thread (run-unit). If subroutines define variable fields in the 'WORKING-STORAGE SECTION', they need code to clear those fields (to low-values) on each entry to the subroutine if it may be called more than once within a single subsystem processing thread (run unit).

'COBREENT' must be called to interface to all Intercomm service routines and user subroutines. The REENTSBS code is still required as the first parameter in the passed parameter list.

CALL identifier (dynamic call) is not supported.

Nested COPY (below the 01 level) may be used: change the \$\$COPY for MMU symbolic maps to a standard COPY statement. The COPRE pre-compile step is no longer needed for Intercomm COPY.

Non-reentrant (do not use a DWS and do not call COBREENT) programs may not be converted to VS COBOL II (unless first recorded). Note that single-threading (serial execution) may be forced via coding MNCL=1 on the SYCTTBL macro.

VS COBOL II user subroutines may not be called by non-COBOL II subsystems (or subroutines), even if called via COBREENT by a reentrant OS/ANS COBOL program. That is, convert subsystems to VS COBOL II before converting any reentrant subroutines called by those subsystems.

Reduce the number of WORKING-STORAGE fields which have VALUE definitions by converting them to literals in the program code (for MOVE statements, for example). The literal table is not copied out to dynamic storage acquired by VS COBOL II. Conversion does not apply to values in COPY members such as ICOMSBS and COBLOGCH.

If DWS fields are moved to WORKING-STORAGE, note that every 01 level definition is forced to the next doubleword boundary, even if the previous field is less than 8 bytes. Therefore, group areas together by field size and alignment under 01 level group-names, such as FILE-AREAS, MMU-AREAS, DATA-BASE-AREAS, etc. See sample program in Chapter 13. Note that fields subordinate to 01 level names may be passed as parameters on subroutine calls.

77 level fields always start on a doubleword boundary, even if the previous field is less than 8 bytes, or not a multiple of 8 in length. Therefore, define such fields which are not 8-byte multiples in length under an appropriate 01 level group (see above).

Prefix programs/stubs/roots may not be linked with VS COBOL II programs as the first routine to receive control. See Operating Reference Manual for user exits (PREPROGI/E) to use for modifying the subsystem parameter list/areas.

SORT, MERGE and File I/O verbs are not supported.

READY TRACE, RESET TRACE and SERVICE RELOAD statements must be deleted (not supported by VS COBOL II).

- Only the IBM VS COBOL II (Release 3.0 and higher) compiler is supported. Installation and linkedit of the VS COBOL II environment is described in other Intercomm documentation. See also Chapter 13.

- Subsystems may be resident, in an EXGRP or OVERLAY A, or dynamically loadable (above the 16M line if possible - COBREENT provides mode conversion). They may not be in OVERLAY B, C, or D (linked with MONOVLY). Subsystems must be coded and linked as reentrant (RENT parm used on linkedit with NOCALL and LET) and if loadable, NOCALL suppresses linking in called routines (COBREENT and IGZEBST, which are in the Intercomm linkedit). The external references to entry points in the Intercomm load module will be resolved by dynamic linkedit. No programs may directly call any user subroutines which call other routines. All user subroutines which are not single-threaded (self-contained) and/or are not in the Intercomm load module must be called via COBREENT. If the subsystem is loaded above the 16M line, then all user subroutines not linked with the subsystem must be called via COBREENT (for mode conversion). On the SYCTTBL macro, LANG=RCOB and REUSE=YES are required.
- Subroutines called via COBREENT must be coded and defined as reentrant, and may be resident or loadable (above or below the 16M line) and must be defined in the REENTSBS table. See Chapter 9 on defining a DWS area for a COBOL subroutine. On the SUBMODS macro, TYPE=COBOL and USAGE=REENT are required.
- All Intercomm Facilities and features available at the user site may be accessed as currently documented, except:

CONVERSE Facility not supported.
DWS checking not applicable to fields moved to Working-Storage.
DWSSNAP will not snap fields/areas defined in Working-Storage,
only those in the DWS area.
- Snap facilities have been enhanced to snap VS COBOL II applicable storage for subsystem threads in indicative dumps, and to provide interface debugging snaps at critical times (see Messages and Codes). Note that in 118 (timeout) and 126 (program check) snaps, the save area chain contains an extra save area because the TGT save area is not removed from the chain as for OS/ANS COBOL, when a reentrant COBOL program calls COBREENT. Under the MVS SAVE AREA TRACE in the snap, there may be two consecutive listings for calls to COBREENT: the first is for the TGT save area, and the second is for the copy of the TGT save area in the DWS prefix. Only the second appears in OS/ANS COBOL snaps.
- A new snap 123 (see Messages and Codes) has been added which is produced (with Intercomm message MP003I) when a recoverable U10nn abend is caused by a VS COBOL II run time subroutine (such as an SSRANGE checking, invalid sign, truncation, or recursive call abend). The thread is cancelled and the subsystem is flagged inactive (NO SCHED). That is, no new messages are processed until the subsystem is corrected and reloaded via the LOAD command, or the FTUN/SSUP commands are used to reactivate the subsystem (set SCHED to YES). An indicative dump (similar to a snap 126) is produced if indicative dumps are active for the region and the subsystem. See IBM's VS COBOL II Application Programming Debugging for abend and error message (IGZ0nnI) explanation.

3.5 RESTARTED MESSAGES

After an Intercomm system failure (abend or operator cancel) or an operating system failure (requiring a re-IPL of the CPU), Intercomm may be brought up in Restart Mode which permits reprocessing of messages in progress at the time of failure. Additionally, previously cancelled messages (see Figure 14), and unprocessed messages (received and queued, but not started) will be requeued for processing after system startup completes. This is accomplished by retrieving the original input messages from the log created in the previous Intercomm execution as described in the Operating Reference Manual, and may be coordinated with file or database record backout as described in the File Recovery Users Guide and DEMS Users Guide.

Restarting of messages for a particular subsystem is controlled by the RESTART parameter of the SYCTTBL macro defining the subsystem in the SCT. A restarted input message (in progress at failure time) contains a log code of C'R' or C'P' (if data base update may be executed by the subsystem). All other input messages contain a log code of C'2' (see Figure 11). A subsystem may need a different processing path for a restarted message and should be careful about creating an output response message which might confuse a terminal operator.

3.6 DWSSNAP FACILITY (Release 10 only)

The DWSSNAP Facility allows a COBOL subsystem to snap data areas from its own DWS; a COBOL subroutine can snap areas from its own DWS and/or areas from the calling subsystem's DWS (data areas passed as parameters to the subroutine via Linkage). The output of the DWSSNAP request may be sent to SNAPDD (unlimited output) with snap ID=087 or may be returned to the inputting terminal (limit is one screen of output per snap, all subsequent pages of output are lost), or may be routed to another terminal, usually a printer (maximum output of 20 pages).

Parameter	Contents
SNCWname	The Snap Control Word, initialized to: 'ØSØØ' (SNAP Option) for output to the system SNAPDD data set; 'ØDØØ' (DISPLAY Option) for output back to the inputting terminal, 'ØPØØ' (PRINTER Option) for output to terminal named in next parm.
term-id	The Intercomm terminal name where output is to be routed. Only coded if PRINTER option used.
parm-address-start	A data name in the subsystem's/subroutine's DWS which represents the start of the area to be snapped.
parm-address-end	A data name in the subsystem's/subroutine's DWS which represents the end (must be a higher address than start) of the area to be snapped.

Coding format:

```
CALL 'COBREENT' USING DWS-SNAP, SNCWname[, term-id]
      [, parm-address-start[, parm-address-end]].
```

The CALL to DWSSNAP can have up to 5 address pairs specified. However, no addresses need be coded if a snap of the entire DWS is desired. For example:

```
CALL 'COBREENT' USING DWS-SNAP, SNCWname.
```

will cause the entire DWS to be snapped.

```
CALL 'COBREENT' USING DWS-SNAP, SNCWname, parm-address-start.
```

will cause a snap of DWS from parm-address-start to the end of DWS.

When using the DWSSNAP Facility to receive output at the inputting terminal, data areas to be snapped (all inclusive) cannot exceed 300 bytes (only one page of output will be sent to the inputting terminal; all additional output will be ignored/lost) when one pair of addresses is specified. If multiple address pairs are specified then the number of bytes that can be snapped is 300 minus 48 (times the number of address pairs desired). The storage snapped will be displayed at the terminal just as it would appear in a formatted dump; hexadecimal digits (to the left) and the alphanumeric equivalent (to the right).

When calling DWSSNAP from a COBOL subroutine, the addresses passed as parms must be within the subroutine's DWS or that of the main COBOL subsystem's DWS. To pass addresses in the DWS of the subsystem from a subroutine, they must be part of the Linkage Section of the subroutine. For example:

LINKAGE SECTION.

```
01 DWS.
   02 SNCW                PIC X(4).
   .
01 RECORD-AREA.
   04 RECORD              PIC X(166).
   04 RECORD-END         PIC X.
```

PROCEDURE DIVISION USING DWS, RECORD-AREA.

```
MOVE 'BDW' TO SNCW.
CALL 'COBREENT' USING DWS-SNAP, SNCW, RECORD, RECORD-END, DWS.
```

will cause a snap, to the inputting terminal, of the 166-byte Record-Area passed to the subroutine by the subsystem via Linkage and the entire DWS of the subroutine, provided the output does not exceed one screen (everything in excess of one screen will be lost). RECORD-END is a dummy delimiter for displaying the passed record area.



Chapter 4

USING THE MESSAGE MAPPING UTILITIES

4.1 CONCEPTS

The Message Mapping Utilities (MMU) provide an interface between the application subsystem and terminal-dependent message processing logic for both input and output messages. MMU is invoked by calls to Intercomm service routines which perform mapping functions based upon user-specified tables (MAPs). Mapping includes justification, padding, and conversion of character data to/from arithmetic format.

4.2 PROCESSING

MMU input mapping produces fixed length data fields prefixed by a two-byte length and one-byte flag (indicates errors or omissions) unless the data fields are defined in a structured (named) segment (contiguous group of fields). In this case the three-byte prefix occurs for the entire segment, not for the individual fields.

MMU output mapping operates upon data in the same format, but the flag byte becomes the field (or segment) attribute character. The mapped input text area and the unmapped output text area are called symbolic maps and are defined by special MMU \$\$COPY statements in the application program's Dynamic-Work-Space for OS/ANS COBOL. Under VS COBOL II, use the standard COPY statement to copy symbolic maps into either the Working-Storage Section or into the DWS area. The application program references data fields and the associated prefix by symbolic name. For example, a customer name field (CUSTMER) of twenty-five characters would appear in an MMU symbolic definition as follows:

06 CUSTMERL	PIC 9(4) COMP.	(length)
06 CUSTMERT	PIC X.	(flag/attribute)
06 CUSTMER	PIC X(25).	(data)

Output message disposition is determined by options passed to MMU: the formatted message(s) may be returned to the subsystem; passed to FESEND for terminal queuing; passed to the Page Facility for CRT page browsing; or spooled to a DDQ for subsequent transmission as a series of report pages for a printer.

A summary of message processing logic using MMU is shown in Figure 18. For a complete description of Message Mapping and its use by application subsystems, refer to the Intercomm Message Mapping Utilities.

Chapter 5

USING THE EDIT UTILITY

5.1 CONCEPTS

The Edit Utility may be used for input messages instead of MMU. It provides an interface to facilitate application program logic for message editing. When editing has been requested for a verb (via Front End Verb Table specification), the Intercomm PREPROG interface program calls the Edit Utility to produce edited message text from data fields entered by the terminal operator.

The edited message becomes the input message passed to the subsystem. The Edit Control routine strips the following field definition characters during the course of editing:

- The system separator character, as defined in the System Parameter List (SPA)
- 3270 CRT SBA sequences
- Dataspeed 40/1 and 2 terminal TAB characters
- New Line characters
- Carriage Return or combined Carriage Return/Line Feed
- End of Text, End of Message, End of Block, or End of Transmission characters.

All other device control characters not translated or otherwise suppressed by the Front End translation table for a particular device will be treated as text within a field.

Editing is controlled by the Edit Control Table (ECT - system table PMIVERBS), which contains all information about each message necessary to perform editing. An edit proceeds field by field based upon the user-specified ECT. Data fields may be edited by Intercomm or user-coded Edit Subroutines. For a complete description of the Edit Utility, its components and processing logic, refer to the Intercomm Utilities Users Guide. The sample program in Chapter 12 illustrates edited message processing.

5.2 PROCESSING RESULTS

The result of processing by EDIT is a message with a standard forty-two-byte message header and data fields in one of the following basic formats:

- Fixed Format

Each edited field is of fixed length in a predefined sequence as follows:

HEADER	DATA 1	DATA 2	-----	DATA N
--------	-----------	-----------	-------	-----------

- Variable Format

Each edited field may vary in length and position in the edited result. Each edited field is prefixed with a one-byte identification code, one-byte length, and possibly a one-byte occurrence number for fields defined as repetitive in the ECT:

HEADER	I	L	DATA X	I	L	DATA Y	-----	I	L	DATA Z
--------	---	---	-----------	---	---	-----------	-------	---	---	-----------

The Edit Utility considers a message successfully edited if there are no required fields (as specified by the Edit Control Table) in error or omitted. In the case of unsuccessful editing, Edit sends an error message to the originating terminal for each required field omitted or in error. If none of the required fields is omitted or in error, it remains the responsibility of the application program to analyze the edited result and perform recovery logic for any non-required fields in error. Figure 19 summarizes results of Edit processing for fields in error.

Field Type	Fixed Format	Variable Format
Non-Required Field Omitted	Field appears in edited result, filled with pad character associated with Edit Subroutine, that is, spaces for alphanumeric field, zero for numeric field, or user-assigned.	Field does not appear in edited result.
Non-Required Field in Error	Field appears in edited result filled with high-values (X'FF').	Field does not appear in edited result.
Required Field in Error or Omitted	Message rejected by EDIT.	Message rejected by EDIT.

Figure 19. Edit Utility Processing of Fields Omitted or in Error

Chapter 6

USING THE FILE HANDLER

6.1 GENERAL CONCEPTS

The Intercomm File Handler provides centralized control over all data file access in the on-line system. Requests for data file access are made in message processing subsystems by calling a File Handler service routine.

The correspondence between the normal COBOL file access functions and the Intercomm File Handler service routines is shown in Figure 20.

Function	COBOL Verbs	Service Routine
Prepare a file for access	OPEN	SELECT
Access logical records sequentially (QSAM,QISAM)	READ,WRITE GET,PUT	GET,PUT GET,PUT
Access logical records randomly (BISAM,BDAM)	READ,WRITE REWRITE	READ,WRITE WRITE
Access physical blocks (BSAM,BDAM)	READ,WRITE	READ,WRITE
Access VSAM files	READ,START WRITE,REWRITE	GETV PUTV
Conclude file access	CLOSE	RELEASE

Figure 20. Functions of File Handler Service Routines

A data file on-line is identified to the File Handler by the existence of a data definition (DD) statement in the execution JCL. Files must be existing (DISP=OLD or SHR) except for sequential output data sets (DISP=NEW or MOD).

DD statement requirements are illustrated in Figure 21. Additional requirements for VSAM are described in that section. Special processing definitions for particular files are defined to Intercomm at system startup by FAR (File Attribute Record) parameters. These include READONLY (prohibit output), OPEN (at startup), file duplexing, etc., and are described in the Operating Reference Manual. Additional parameters for file recovery (in case of program or system failure) are described in the File Recovery Users Guide.

//ddname*	DD	DSNAME=**	
//		,DISP=**	
//		,DCB=(DSORG=**	
//		,OPTCD=**	For BSAM,BDAM,BISAM only.
//		,RECFM=	Must be specified by existing
//		,BLKSIZE=	data set label or explicitly
//		,LRECL=	in DD statement.
//		,NCP=	
//		,LIMCT=	
//		etc.)	

*Name used to identify file in calls to SELECT.
**Marks those parameters which must be explicitly specified on the DD statement for each data set.

Figure 21. DD Statement Parameters for the File Handler.

In centralizing data file accesses, the File Handler provides one central set of control blocks for each file, thus reducing core requirements in individual message processing subsystems. There are no File Description entries in a COBOL-coded Intercomm program.

Furthermore, all the facilities of the following Operating System Data Management functions are accessible to any subsystem: BDAM, BSAM, QSAM, BISAM, QISAM and VSAM.

The File Handler also supports the following ISAM replacement access method available from another vendor: IAM.

Data Base interfaces supported under Intercomm (IDMS, ADABAS, TOTAL, DL/I) are described in the DBMS Users Guide and the respective vendors' manuals.

6.1.1 Subsystem Processing

In the on-line environment, several subsystems in concurrent execution may require access to the same data file. Rather than each subsystem issuing an OPEN and corresponding CLOSE for accessing a particular file, the File Handler will open a file the first time it is accessed (unless already opened at startup) and the file remains open for the duration of the on-line job in execution. A SELECT request simply establishes internal control blocks and the corresponding RELEASE request merely disconnects those internal control blocks. In each subsystem, following a SELECT for a particular file, access functions (READ, WRITE, GET, PUT, GETV, PUTV) may be called as many times as may be necessary for message processing logic. RELEASE must be called for each selected file prior to the return to the System Monitor.

Each subsystem must provide space for two File Handler control areas. The information in these areas is unique for each message thread, so they must be defined in the Dynamic-Work-Space of reentrant programs, that is, defined in the Linkage Section as a subordinate item to the fifth entry parameter. To assure that they are fullword aligned, they should be defined following an eight-digit computational item, such as 02 FILLER PICTURE S9(8) COMP SYNC. Figure 22 shows how these control areas may be defined so as to force the proper alignment.

```

02 FORCE-ALIGN PIC S9(8) COMP SYNC.
02 FHCW REDEFINES FORCE-ALIGN.
  04 FH-RET1 PIC X.
    88 IOK VALUE 0.
    88 IOERROR VALUE 1.
    88 NOT-FOUND VALUE 2.
    88 EOF VALUE 2.
    88 XTO VALUE 3.
    88 NO-DD VALUE 9.
  04 FH-REQ1 PIC X.
  04 FH-REQ2 PIC X.
  04 FH-RET2 PIC X.
02 EXTDSCT1 PIC X(48).
02 EXTDSCT2 PIC X(48).

```

Figure 22. Defining File Handler Control Areas

For each call to a File Handler service routine, the File Handler is passed the addresses of the two control areas. The first is an aligned 48-character area, called an External DSCT (EXTDSCT), which the File Handler uses to save control information for the subsystem processing thread, from the time that a given file is first SELECTed until it is finally RELEASEd. A unique EXTDSCT must be defined for each file concurrently accessed within the same processing thread. The other control field, called the File Handler Control Word (FHCW), is an aligned four-character field used for communication between the File Handler and the calling subsystem. Prior to each call to a service routine, the subsystem must clear the FHCW with spaces or initialize it with a predefined request code as described for each routine. A code of space (blank) is indicated in the detailed access descriptions by the lower case letter $\$$. An example of such a request would be to establish Exclusive Control during a call to READ with intent to update. The File Handler will return a completion code in this word, after servicing a request, to communicate the status of the operation back to the subsystem.

6.2 CALLING SERVICE ROUTINES

A COBOL subsystem may call the File Handler service routines through the Intercomm interface module COBREENT, and provide a routine-code name corresponding to the desired routine name, as described in the Intercomm COPY member ICOMSBS. The COBREENT prototype coding format is described in Chapter 3.

The parameters for the File Handler service routines are described in Figure 23. The specific parameters passed to a given service routine depend on file requirements and the processing options of the particular service routine called. If the calling subsystem (or subroutine) might be loaded above the 16M line, then all parameters (except the ICOMSBS code) must be in the 24-Amode DWS (may be in Working-Storage for VS COBOL II).

Parameter	Content
EXTDSCTname	A 48-character fullword-aligned area supplied by the subsystem for the File Handler's use for each file SELECTed (see Figure 22)
FHCWname	The File Handler Control Word, in which the File Handler returns a completion code to the subsystem (see Figure 22)
ddname	An eight-character constant initialized with the name of the DD statement describing the data set to Intercomm (move to the DWS for calls from 31-Amode OS/ANS COBOL programs)
Record-area	The area for data read from, or to be written to, the file
Key	The key for file access (ISAM, Keyed BDAM, VSAM-KSDS)
VSAM RBA	Four-byte Relative Byte Address number (ESDS)
VSAM RRN	Four-byte Relative Record Number (RRDS)
Block-ID	Applies only to BDAM files: <ul style="list-style-type: none"> • three-byte relative block number (RBN) • three-byte relative track and record number (TTR) • eight-byte actual address (MBBCCHHR)

Figure 23. File Handler Service Routine Parameters

The File Handler IAM support uses the Intercomm ISAM support routines.

On return from a File Handler service routine, the leftmost position of the FHCW area will contain a character code indicating the result of the operation, as shown in Figure 24. Additionally, for VSAM files, the rightmost position of the FHCW will contain a VSAM reason code.

Code	Meaning
0	Normal completion
1	Hardware I/O error
2	Unusual condition (EOF, invalid key, etc.)
3	Exclusive control time-out occurred
4-8	Not used
9	Invalid request (no DD statement, invalid parameter sequence, attempt to output to an input only file, etc.)

Figure 24. Outline of File Handler Return Codes

The application subsystem logic must then analyze this return code and take appropriate error recovery action. An error message might be created and queued for output to the terminal. Otherwise, the subsystem can return to the Subsystem Controller with a return code of 12, indicating that the Subsystem Controller should call the USRCANC routine which in turn will send an error message to the terminal.

6.2.1 Automatic Error Checking

If the application subsystem logic is such that special error recovery processing is not required, the File Handler will perform error checking itself and data will be returned to the subsystem only if the return code is zero. Otherwise, the File Handler will force a program check, which causes cancelling of the input message and return to the Subsystem Controller, which calls the USRCANC routine. To request this function, place a character 'C' in the first byte of the FHCW prior to calling a File Handler service routine.

6.3 SELECT, RELEASE FUNCTIONS

SELECT must be called to initialize the subsystem's EXTDSCT prior to any data access function performed by the File Handler. Prior to the call to SELECT, the subsystem's EXTDSCT must be initialized to low values.

RELEASE must be called to notify the File Handler that its pointers to the subsystem's EXTDSCT should be cleared and that all data access to a particular file within one subsystem thread is complete. There must be a RELEASE corresponding to each SELECT of a file. Multiple SELECTs of the same file using the same EXTDSCT are not permitted without intervening RELEASEs, within the same processing thread. After each RELEASE, the EXTDSCT should be cleared to low-values before being reused.

Coding format:

```
CALL 'COBREENT' USING FH-SELECT, EXTDSCTname, FHCWname, ddname.
```

```
CALL 'COBREENT' USING FH-RELEASE, EXTDSCTname, FHCWname.
```

Note: the ddname must be in the DWS if the calling program can be loaded above the 16M line (except if VS COBOL II).

Figure 25 describes the return codes for SELECT and RELEASE.

Return Codes (First Byte of FHCW)	SELECT	RELEASE
0	A reusable file (disk input) ready for access; sequential access begins at first record.	Successful release
1	A nonreusable file (SYSOUT, disk output (DISP=NEW/MOD or DISP=SHR/OLD and FAR WRITEOVER parm specified, or a data set on tape) ready for access, begins after last record previously accessed. Or empty/reused VSAM ESDS file ready for output only.	Not applicable
9	No ddname found in File Handler internal control table. (No DD statement in JCL or the file has been "locked" by the FILE control command.)	File not selected.

Figure 25. File Handler SELECT/RELEASE Return Codes

6.3.1 Closing a File

Occasionally, it is necessary to close a file, perhaps because it is to be updated by a batch job. A special form of RELEASE requests the File Handler to close a file. However, unless some external control is taken to assure that no other programs have selected the file, a close request could cause other transactions for the file to fail. Also, if new transactions are attempting to access the closed file, the File Handler will open it again and unpredictable results may occur. Intercomm provides the FILE system control command for systemwide file access control.

To close a file from an application program:

- If the file has been previously selected: first release the EXTDSCT by calling RELEASE referencing the EXTDSCTname used when the file was selected (as described above), then
- Move a character C to the second byte of the FHCW ('~~BCB~~') and call RELEASE supplying the ddname of the file to be closed; use the following coding format:

```
CALL 'COBREENT' USING FH-RELEASE, ddname, FHCWname.
```

Note: the ddname must be in the DWS if the calling program can be loaded above the 16M line (except if VS COBOL II).

6.4 EXCLUSIVE CONTROL FOR NON-VSAM FILES

In a multithread environment with only inquiry applications, the fact that several message processing programs may concurrently retrieve data from the same file or files presents no operational problems. However, when more than one message processing program attempts to update or add records to a file, data integrity problems can occur. Figure 26 illustrates the problems of concurrent updates; program B's update nullifies that of program A. Exclusive control implies that while one program is operating on a record, that is, the time between a READ and a WRITE, all other requests to read or write that particular record will be delayed. A program requesting a record held during exclusive control by another program is not notified of this delay, but rather stops execution in the File Handler until exclusive control is either removed or expires so that the File Handler can then proceed with the requested function. Exclusive control, when required, must be requested separately with each call to File Handler READ or GET functions. Exclusive control for basic access methods operates at the block or record level. Exclusive control for queued access methods operates at the data set level; thus applications should be designed to avoid GET for update whenever feasible.

To obtain exclusive control over the entire data set in a QISAM file or over a physical block in a BDAM or BISAM file, move '~~BXB~~' to the File Handler Control Word prior to calling GET or READ. Exclusive control does not apply to physical sequential (QSAM/BSAM) files.





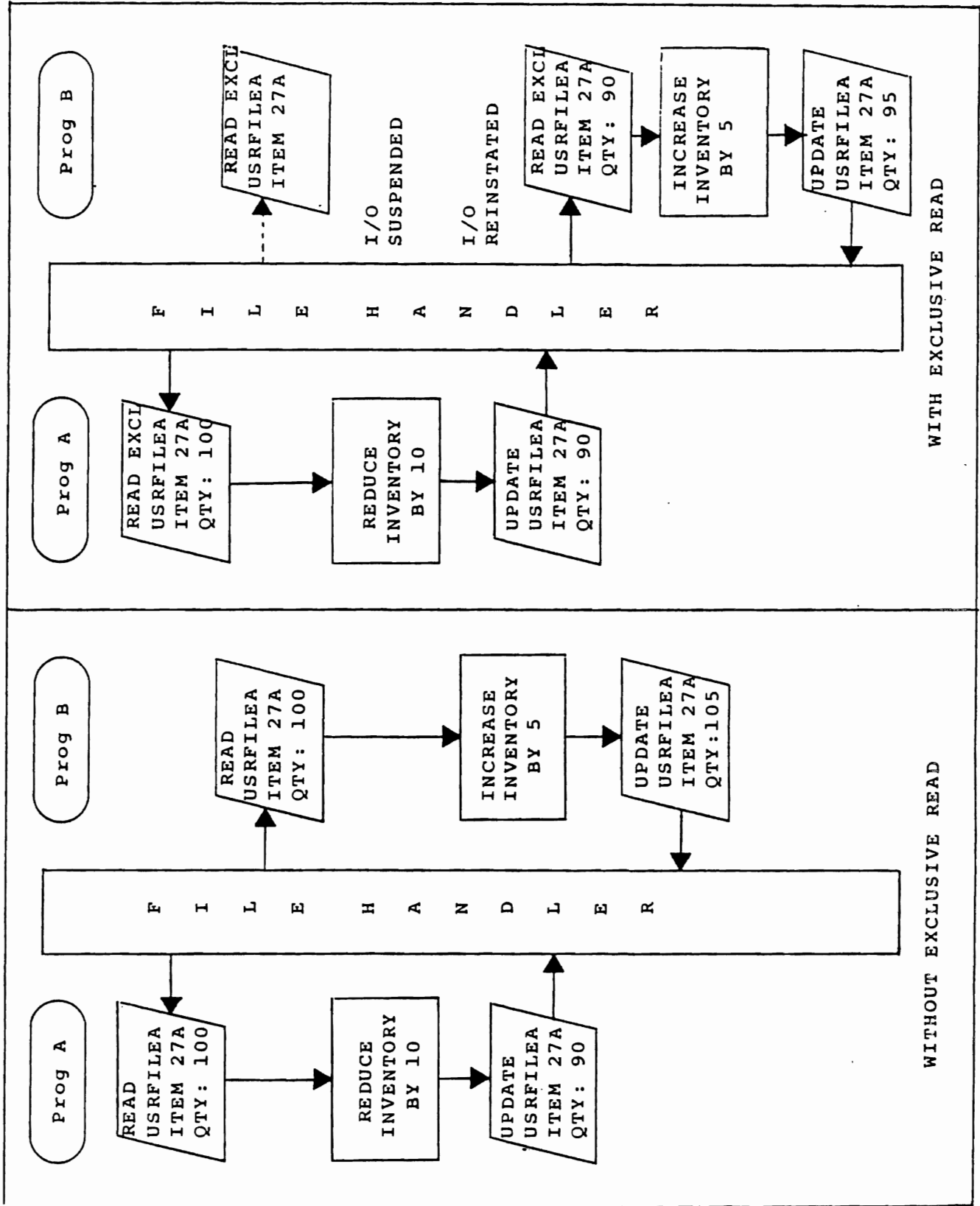


Figure 26. Exclusive Control Processing

Exclusive control will be released by:

- A call to WRITE or PUT referencing the same EXTDSCTname, that is, the update of the previously acquired record, and no key or block-id specified.
- A call to WRITE referencing the same EXTDSCTname and a key and/or block-id is specified.
- A call to READ or GET referencing the same EXTDSCTname (retrieving a new record from the file).
- A call to RELEASE referencing the same EXTDSCTname.
- An elapsed time after the call to READ with Exclusive Control greater than the exclusive control time-out value of the File Handler. This is set at two minutes for any given record and a maximum of ten minutes for consecutive exclusive accesses to a QISAM file.

NOTE: A return code of 3 after a call to WRITE or PUT to update a record held in exclusive control indicates that exclusive control timed out: the WRITE or PUT did not take place. The program should re-READ or re-GET the same record with exclusive control and WRITE or PUT again, after reprocessing the record.

- A call to RELEX, if the program logic is such that the record does not need to be updated, or additional and time-consuming activity (accessing other files) is required before resuming access to the file. Such a program could call RELEX to release exclusive control without actually RELEASEing the file until later in the program logic.

6.4.1 Release Exclusive Control--RELEX

RELEX is called to release Intercomm or VSAM exclusive control without having to read, update, time-out, or RELEASE the file.

Coding format:

CALL 'COBREENT' USING FH-RELEX, EXTDSCTname, FHCWname.

Return Code	Meaning
0	Exclusive control released
9	File not selected or invalid function

Figure 27. File Handler Release Exclusive Control (RELEX) Return Codes

6.5 SEQUENTIAL ACCESS METHOD (SAM) PROCESSING6.5.1 File Handler Service Routines--GET, PUT (QSAM); READ, WRITE (BSAM)

GET is called to access the next sequential logical record from a file. PUT is called to write the next sequential logical record to a file. READ is called to access the next sequential physical block. WRITE is called to write the next sequential physical block. If PUT or WRITE is called referencing a disk data set, the record last accessed by a GET or READ will be updated, however, the length may not be changed. GET processing is subtasked by the File Handler in order to provide multithreading facilities; for further details, see the Operating Reference Manual.

Coding format:

```
CALL 'COBREENT' USING FH-GET, EXTDSCTname, FHCWname,
record-area [,record-length].

CALL 'COBREENT' USING FH-READ, EXTDSCTname, FHCWname,
record-area [,record-length].

CALL 'COBREENT' USING FH-PUT, EXTDSCTname, FHCWname,
record-area [,record-length].

CALL 'COBREENT' USING FH-WRITE, EXTDSCTname, FHCWname,
record-area [,record-length].
```

Return Codes	GET, READ	PUT, WRITE
0	Successful	Successful
1	I/O Error	I/O Error
2	End-of-file	(Not applicable)*
9	Not selected or invalid function; that is, using an output-only file	Not selected or invalid function; that is, using a tape input file or readonly file, or file not sequential.

* For WRITE to a disk file: indicates End-of-file (write not done)

Figure 28. File Handler Sequential Access Method Return Codes

6.5.2 Undefined Record Format and Record Length

The record-length parameter is valid and required only when a file with an undefined record format (DCB=RECFM=U) is accessed. The record-length parameter points to a fullword containing the length of the output record before a PUT or WRITE operation, or to contain the length of the input record after a GET or READ operation. The second character of the File Handler Control Word must be set to U to utilize this feature. Do not code the DCB subparameter LRECL on the DD statement for the file in the Intercomm execution JCL. The BLKSIZE, RECFM and DSORG subparameters are required.

6.5.3 Variable-Length Record Format and Record Length

Variable-length records start with a Record Descriptor Word (RDW) which must be fullword aligned (PIC 9(8) COMP SYNC). The first two bytes of the word contain the record length in binary (+4 for the RDW); the second two bytes contain binary zeros (low values). The RDW is followed immediately by the record data, and must be recognized by the subsystem on input, and provided and initialized on output.

For blocked files, if GET or PUT are used, the access method will perform the blocking and deblocking. If READ or WRITE are used, the application program must perform the deblocking (READ) and blocking (WRITE). In this case, the block must start with a Block Descriptor Word (BDW) of four bytes (aligned); the first two bytes contain, in binary, the total block length (including 4 for the BDW), and the second two bytes contain binary zeros (low values). For JCL details, and FAR options for defining and accessing the file, see the Operating Reference Manual.

6.6 INDEXED SEQUENTIAL ACCESS METHOD (ISAM) PROCESSING

To use an ISAM file on-line under Intercomm, do not define three DD statements (INDEX/PRIME/OVERFLOW) for either the off-line creation of the ISAM data set, or the on-line execution DD statement. For creation, let the access method set up the index and overflow areas (use CYLOFL parameter on DD statement). For on-line execution, define only DISP=OLD and the data set name, volser and unit parameters if not catalogued, and the DCB parameter DSORG=IS. Optionally, the DCB parameter OPTCD may also be specified. See also the descriptions of FAR parameters applicable to ISAM data sets described in the Operating Reference Manual.

6.6.1 File Handler Service Routines--GET, PUT (QISAM); READ, WRITE (BISAM)

GET is called to access the next sequential record, or to reposition (if a key is specified) and access the next sequential record. READ is called to retrieve a specific record at random. PUT is called to update the last record retrieved by a call to GET. WRITE is called to update the last record retrieved by a call to READ, or to add a record to the file (if a key is specified). For update, exclusive control may be requested; otherwise use blanks in the FHCW.

Coding format:

to retrieve next sequential record:

```
CALL 'COBREENT' USING FH-GET, EXTDSCTname, FHCWname, record-area.
```

to reposition and retrieve record with key equal or high:

```
CALL 'COBREENT' USING FH-GET, EXTDSCTname, FHCWname, record-area,
key.
```

to update last GET:

```
CALL 'COBREENT' USING FH-PUT, EXTDSCTname, FHCWname, record-area.
```

to retrieve a specific record:

```
CALL 'COBREENT' USING FH-READ, EXTDSCTname, FHCWname, record-area,
key.
```

to update last READ:

```
CALL 'COBREENT' USING FH-WRITE, EXTDSCTname, FHCWname, record-area.
```

to add a specific record:

```
CALL 'COBREENT' USING FH-WRITE, EXTDSCTname, FHCWname, record-area,
      key.
```

Figure 29 describes return codes for ISAM access.

QISAM Return Codes	GET w/o Key	GET w/Key	PUT
0	Next sequential record retrieved	Record with equal or next higher key retrieved	Record from previous GET updated
1	I/O error	I/O error	I/O error
2	End of File	Key out of range	N/A
3	N/A	N/A	Exclusive Control Time-out
9	File not selected or invalid function	File not selected or invalid function	File not selected or invalid function
BISAM Return Codes	WRITE w/o Key	WRITE w/Key	READ
0	Record from previous READ updated	Record with specified key added	Record with equal key retrieved
1	I/O error	I/O error	I/O error
2	N/A	Key already exists or no room to add new record	Key does not exist
3	Exclusive Control Time-out	N/A	N/A
9	File not selected or invalid function	File not selected or invalid function	File not selected or invalid function

Figure 29. File Handler ISAM Return Codes

6.7 DIRECT ACCESS METHOD (BDAM) PROCESSING

BDAM files are accessed by block-id. The form of the block-id is defined in the OPTCD subparameter of the DCB parameter of the DD statement and the same form must be used by all programs accessing the file:

- OPTCD=RF--block-id is three-byte binary RBN (relative block number) for fixed-length files only
- OPTCD=AF--block-id is eight-byte actual MBBCCHHR
- OPTCD=F--block-id is three-byte binary TTR (relative track and record number) for fixed- or variable-length files.

The F permits feedback (of block-id) requests: the form of the block-id is that requested by the OPTCD parameter. For Keyed BDAM with extended search, insert an E immediately after the = sign (that is, code OPTCD=ERF, etc.), and specify the LIMCT subparameter on the DCB parameter of the DD statement.

6.7.1 File Handler Service Routines--READ, WRITE (BDAM)

READ is called to retrieve a physical block. WRITE is called to update a block previously read, to replace an existing block in a preformatted file, or to add a new block.

Coding format:

```
CALL 'COBREENT' USING FH-READ, EXTDSCTname, FHCWname,
                    record-area[, key], block-id.
```

```
CALL 'COBREENT' USING FH-WRITE, EXTDSCTname, FHCWname,
                    record-area[, key][, block-id].
```

Figure 30 shows FHCW options (byte 2) for standard and keyed BDAM files, and when to use key and/or block-id fields. Figure 31 describes the corresponding return codes. When reading a keyed BDAM file, the key will be read into the key field if a key parameter is passed and the key is not used as the search argument (w/o extended search). For a keyed BDAM file, replace requires a previous read; update and replace are synonymous.

Intercomm provides two utilities for off-line preformatting of fixed-length BDAM files:

- CREATEGF for BDAM files without keys
- KEYCREAT for BDAM files with keys.

These utilities are described in the Operating Reference Manual.

1. BDAM Files Without Keys

Code	Request	Macro
Ø	READ w/o exclusive control, w/block-id	READ DIF
X	READ w/exclusive control, w/block-id	READ DIX
Ø	WRITE to update last READ, w/o block-id	WRITE DI/DIX
Ø	WRITE to update/replace w/o previous READ, w/block-id	WRITE DI
A	WRITE to add a record--variable-length only (record address returned automatically in caller's block-id field)	WRITE DAF

2. BDAM Files With Keys

Code	Request	Macro
*Ø	READ data block only w/o exclusive control (w/extended search) w/key, w/block-id	READ DKF
*X	READ data only w/exclusive control (w/extended search) w/key, w/block-id	READ DKX
J	READ key and data block w/o exclusive control w/o extended search, w/block-id (w/key)	READ DIF
I	READ key and data w/exclusive control w/o extended search, w/block-id (w/key)	READ DIX
*Ø	WRITE to update data only w/o extended search w/key	WRITE DKF/DKX
I	WRITE to update key and data w/o extended search, w/key (w/block-id)	WRITE DI/DIX
*A	WRITE to add a record--next available space w/key, w/block-id (w/extended search)	WRITE DAF
*Feedback of record addresses may be requested for these options only by placing an F in byte 3 of the FHCW.		

Figure 30. File Handler BDAM Option Codes.

NOTE: The DI form of the macros (issued in the File Handler) requires that the block-id field contains the exact address of the data record in the form specified by the OPTCD subparameter on the DD statement. With the DK form, if

extended search is not specified (via E on the OPTCD subparameter), only one track is searched for a record with key matching that passed in the key field, and starting at the address specified in the block-id field. A WRITE for update of last READ does not need a block-id, as positioning is remembered internally.

1. BDAM Files Without Keys

Return Codes	READ	WRITE w/o block-id	WRITE w/block-id
0	Block retrieved	Block from previous READ updated	Specified block added/replaced
1	I/O error	I/O error	I/O error
2	Block out of range	N/A	RECFM=F... Block out of range
			RECFM=V... No space available/ block out of range
3	N/A	Exclusive Control Time-Out	N/A
9	File not selected or invalid function	File not selected or invalid function	File not selected or invalid function

2. BDAM Files With Keys

Return Codes	READ	WRITE w/o block-id	WRITE w/block-id
0	Logical record retrieved	Record from previous READ updated	Specified record added
1	I/O error	I/O error	I/O error
2	Key not found (READ w/key)	Key not found at block-id saved from previous READ (WRITE DK only)	RECFM=F... No dummy record found
			RECFM=V... No space available
3	N/A	Exclusive Control Time-Out	N/A
9	File not selected or invalid function	File not selected or invalid function	File not selected or invalid function

Figure 31. File Handler BDAM Return Codes

6.8 VIRTUAL STORAGE ACCESS METHOD (VSAM) PROCESSING

VSAM support is provided for all three file types: KSDS, ESDS, and RRDS. Subsystems designed to access VSAM files use two File Handler service routines; GETV and PUTV. SELECT and RELEASE function for VSAM as they do for OS data sets. Calls are similar to the standard File Handler format, with the File Handler Control Word (FHCW) used to specify VSAM options. DD statements for VSAM must specify AMP=(AMORG) and for fixed-length data records, 'RECFM=F' must also be specified on the AMP parameter: AMP=(AMORG,'RECFM=F'). FAR options and execution options for VSAM files such as LSR buffer pool support, empty ESDS file load or overwrite, and data set name sharing, are described in the Operating Reference Manual. Most users converting ISAM to VSAM can continue to use their current File Handler calls. Refer to "ISAM/VSAM Compatibility under Intercomm" later in this chapter for further details.

6.8.1 File Handler Service Routines--GETV, PUTV (VSAM)

A VSAM call may request either sequential or direct access and may specify access for KSDS via keys (keyed access) or for ESDS via Relative Byte Addresses (addressed access). A keyed access call for direct retrieval may provide either a generic key or a full key, and may specify a search for either an equal (generic) key or for the first greater-or-equal (generic) key.

A VSAM Relative Record Number Data Set (RRDS) may be accessed sequentially, or directly by Relative Record Number. A direct access request to a RRDS is made by supplying the Relative Record Number of the desired record instead of a key or RBA. All direct accesses to an RRDS must specify "full key, search equal." RBA access is not allowed and RRNs should not be converted to RBAs for access to an RRDS. Records may be inserted into empty slots in an RRDS but a record may not be added with a higher relative record number than the maximum allowed. This maximum is specified when the data set is defined to VSAM.

GETV calls are processed assuming that no update will be performed unless the caller so specifies. The caller may switch back and forth from direct to sequential access, provided VSAM rules are not violated, for example, keyed request against an entry-sequenced data set. The File Handler service routine GETV is called for retrieval. The File Handler service routine PUTV is called for storage or deletion.

Coding formats:

For sequential access

```
CALL 'COBREENT' USING FH-GETV, EXTDSCTname, FHCWname, record-area.
```


Coding formats (continued):

For direct access

```
CALL 'COBREENT' USING FH-GETV, EXTDSCTname, FHCWname,
      record-area, {rba}.
                   {key}
                   {rrn}
```

For update of record retrieved by preceding GETV or for sequential addition

```
CALL 'COBREENT' USING FH-PUTV, EXTDSCTname, FHCWname, record-area.
```

For direct addition of a new record

```
CALL 'COBREENT' USING FH-PUTV, EXTDSCTname, FHCWname,
      record-area, {rba}.
                   {key}
                   {rrn}
```

where:

EXTDSCTname is the standard File Handler parameter.

FHCWname is the standard File Handler parameter. Its VSAM use is to define processing options and to return completion codes to the caller (see Figures 32 and 33).

record-area is the label of the user's I/O area. For fixed length records, no length is specified and data will start in the beginning of the area. For variable length, the first four bytes of the area are used as an OS-type, fullword-aligned, variable record descriptor word (RDW), the first two bytes of which specify the appropriate length in binary (data length +4); data begins in the fifth byte. For GETV, the File Handler will return this length to the caller and for PUTV, the caller must provide the length to the File Handler.

rba is the label of an aligned fullword containing the Relative Byte Address when required for addressed access.

key is the label of a field providing a key, when required for keyed access. If a generic key is provided, then the first two bytes of this field must be the length, in binary, of the generic key which must begin in byte 3, and the field must be fullword-aligned.

rrn is the address of a fullword-aligned field providing a four-byte binary Relative Record Number whose value is 1 to n, where n is the maximum record number defined for the data set.

6.8.2 VSAM Processing Options

The following determine the mode of VSAM access to be performed:

- The preceding call

A VSAM call is dependent upon the preceding call only in two cases: PUTV for update, or sequential GETV or PUTV calls requiring initial positioning.

In the first case, the PUTV call must be immediately preceded by a GETV for update, which identifies the record to be updated. The PUTV for update has no fourth parameter because the key, RRN or RBA was defined by the prior GETV. In the second case, a direct call providing a key, RRN or RBA and requesting positioning must be issued in order to process sequentially starting from that point in the file. To request positioning in this manner, specify S in the second byte of the FHCW for the direct call to GETV; the first record in the sequence will be returned. For an ESDS file, a GETV call without a fourth parameter results in sequential reads from the beginning of the file; the S in the FHCW is unnecessary.

- The presence or absence of the fourth parameter

With the exception of a PUTV for update, all calls for direct access specify a fourth parameter and all subsequent calls for sequential access specify only three parameters.

- The contents of the File Handler Control Word

The second and third bytes of the FHCW are used to complete the definition of the options desired. Alphabetic codes are used and positive tests are made for each defined code. When no defined code is present, the default option (blank) is used.

Bytes 1 and 2 of the FHCW are utilized the same as for OS Access Methods for Return Codes (Byte 1) and Special Requests (Byte 2). The first byte of the FHCW will contain a zoned decimal digit upon return from GETV or PUTV. A nonzero value indicates an error or an exceptional condition.

Byte 2 is used in conjunction with direct access. When an S is provided in byte 2, the direct access is treated as the first of a series of sequential requests which begins at a point specified by the fourth parameter. Therefore, a VSAM POINT will be issued and sequential access will subsequently be performed for the next call.

Byte 3 is used for all VSAM calls as illustrated in Figure 32. There are five default (blank) cases:

- GETV with three parameters (subsequent sequential access)
- GETV with four parameters (search key/RRN equal, no update)
- PUTV with three parameters with no prior GETV for update (sequential add/insert)
- PUTV with three parameters and with a prior GETV for update
- PUTV with four parameters (direct key/RRN add/insert)

6.8.3 FHCW Reason Codes for VSAM

Byte 4 is used to provide VSAM reason codes (from the RPL feedback field) upon completion of a VSAM file access request. In VSAM, a distinction is made between logical and physical errors. In either case VSAM returns a supplementary reason code in hexadecimal defining the condition more precisely. Accordingly, the File Handler will return this reason code in FHCW byte 4, for the caller's use. If the File Handler was called at an ISAM entry point (GET/PUT, READ/WRITE), the code returned in FHCW byte 1 may differ from GETV/PUTV calls (in order to maintain compatibility with existing ISAM subsystems). Figure 33 summarizes VSAM and ISAM/VSAM return codes. VSAM reason codes are fully documented in IBM's VSAM Macros manual.

6.8.4 Exclusive Control for VSAM Files

VSAM automatically provides exclusive control of a control interval (physical block) whenever a GETV for update is processed if the file was defined with SHAREOPTION 1 or 2. The subsystem must release this exclusive control via a call to RELEX before another GETV is issued for the same file, unless an intervening PUTV for update or erase is issued. If no subsequent GETV will be issued, the call to RELEASE will also release exclusive control. There is no VSAM exclusive control time-out. If the VSAM file is accessed by more than one region (Intercomm and/or batch), see IBM documentation on VSAM SHAREOPTIONS, and the Intercomm Operating Reference Manual.

6.8.5 Loading an ESDS Data Set

During File Handler initialization at startup, if an ESDS file is empty or if the FAR parameter WRITEOVER is specified, the ESDS file is flagged as output-only. The first call to SELECT for the file will return a code of 1. The program receiving this code may only use a PUTV call for the file. If that (or any other) program will need to get a record from the file, it must close the file via calls to RELEASE (see Section 6.3.1), after the first PUTV. The next call to SELECT will return a code of 0, and any subsequent call to GETV or PUTV will then cause the file to be reopened for input/output and multi-thread access is then permitted. PUTV calls should be single-threaded to ensure file integrity.

6.8.6 Alternate Path Processing of Keyed VSAM Files

Base Cluster and Alternate Path processing of keyed VSAM files is supported with the following (VSAM-imposed) restrictions:

- If defined in the JCL, the DD statement for the base cluster must be before those for any related paths, and open at startup must be requested via a FAR. Also, both the base cluster and the paths must be connected to an LSR buffer pool.
- Each path to be accessed on-line must be defined in the JCL and be SELECTed with the corresponding ddname. When created, the path must be defined with the UPDATE option.
- The FAR READONLY option must be specified for all paths and the base cluster (if defined) except for the path used for updating, when Shareoption 2 is in effect for the base cluster. If updating is only via the base cluster, then READONLY must be specified for all associated paths. VSAM will not allow any accesses to a base cluster under Shareoption 1 when one path has opened it for update. A base cluster under Shareoption 3 may be accessed for reads or updates by more than one path at any time, however no exclusive control (read/write file integrity) is provided by either VSAM or Intercomm. For Intercomm-provided exclusive control for Shareoption 4, see the Operating Reference Manual.
- If multiple paths are accessed, and/or retrieval/update is done via the path(s) and the base cluster, retrieval of updated versions of the records can be ensured via the FAR DSN and LSR parameters.
- Since duplicate keys may occur in an Alternate Index, the application program is responsible for checking for duplicate keys. Sequential processing (GETV type 1) can be used after the first GETV with key (and an S in byte 2 of the FHCW) in order to retrieve subsequent records. The program can test to see if the last record under a duplicate key was retrieved by checking the VSAM reason code which will be placed in byte 4 of the FHCW. See IBM's VSAM Macros manuals for reason code values.
- The alternate index data set must be defined with the UPGRADE attribute and be built prior to Intercomm startup. An attempt to retrieve a record from an empty file will cause a program check.
- Alternate index data sets should not be defined in the JCL unless access to a data record containing the prime keys is desired, or path processing is not used. Only readonly processing should be done for an AIX and for any related paths and for the base cluster, otherwise, retrieval of the current version of a record is unpredictable.

Type	Service Routine	Access or Action	FHCW Byte 3		KEY/RRN or RBA	Comments
			Update	No Update		
1	GETV	Sequential	U	default	---	In KEY or RRN sequence
2	GETV	Sequential	A	R	---	In RBA sequence (default for ESDS)
3	GETV	Direct	U	default	Full Key or RRN	Search =
4	GETV	Direct	L	F	Full Key	Search greater or = (not valid for RRDS)
5	GETV	Direct	=	E	Generic Key	Search = (not valid for RRDS)
6	GETV	Direct	>	G	Generic Key	Search greater or = (not valid for RRDS)
7	GETV	Direct	A	R	RBA	Addressed Access
8	PUTV	Sequential Add or Insert	default		---	No prior GETV for update (insert not allowed for Addressed Access)
9	PUTV	Update	default		---	Prior GETV for update required (Addressed Access update may not change length)
10	PUTV	Erase	E		---	Prior GETV for update required (not valid for Addressed Access)
11	PUTV	Direct Add or Insert	default		Key or RRN	(no prior GETV)
12	PUTV	Add	A		RBA	Insert not valid

Figure 32. File Handler VSAM Call Summary

Condition at Completion of Operation*	FHCW		
	Byte 1 (char)		Byte 4 (hexadecimal)
	VSAM	ISAM	
Successful completion (A)	0	0	04,08,0C,10,1C
Physical I/O error (A)	1	1	04,08,0C,10,14,18
End of data (1, 2)	2	2	04
No record found (3, 4, 5, 6, 7)	2	2	10
Key not within defined key ranges (3, 4, 5, 6, 7)	2	1	24
Duplicate key (8, 11)	9	2	08
Key out of ascending sequence (8)	9	2	0C
Update attempt with new key (9)	9	9	60
Key exceeds maximum (5, 6)	9	**	70
Addressed update changes length (9)	9	**	64
Invalid RBA provided (7, 12)	9	**	20
Required positioning not performed (1, 2, 8)	9	**	58
Direct or update call while loading (8) GETV for ESDS while loading (2,7)	9	9	74
Insufficient disk space (8, 9, 11, 12)	9	9	1C
Record on unmountable volume (1-7, 11, 12)	9	9	18
Invalid Relative Record Number (3,11)	9	**	C0
Invalid RBA access to a RRDS file (7,12)	9	**	C4
Invalid EXTDSCT or file unavailable or PUTV called for READONLY file	9	9	00
<p>*Characters in parentheses reference the type(s) of VSAM Call (Figure 32) which apply. A = all cases.</p> <p>**Should not occur. The File Handler will force a program check condition to terminate the message in progress.</p>			

Figure 33. File Handler VSAM Return and Feedback Codes

6.9 ISAM/VSAM COMPATIBILITY UNDER INTERCOMM

Subsystems accessing ISAM files can function with little or no modification when their files are converted to VSAM. Intercomm's ISAM/VSAM interface does not use IBM's VSAM/ISAM interface modules. See the Operating Reference Manual for steps necessary to activate the interface. When processing a VSAM data set, the File Handler uses QISAM compatible access for a GET or PUT call and BISAM compatible access for a READ or WRITE call.

An ISAM retrieval is converted to a VSAM GET for update. If a key is provided, it is, of course, treated as a full key. For GET with a key, positioning and a search for a greater or equal key is performed. For READ, a search is made for an equal key. File Handler logic will initialize the user FHCW prior to performing the VSAM function as follows:

- Byte 2 is set to 'S' to force sequential positioning.
- Byte 3 is set to 'U' or 'L' to force update mode.

ISAM delete code processing continues to function as usual via the OPTCD subparameter of AMP on the DD statement. The new OPTCD parameters (I, IL) which specify supplementary delete code processing are supported also.

The following considerations apply to ISAM users converting to VSAM and should be carefully observed:

- ISAM subsystems must already be operational for ISAM files before accessing VSAM files. Erroneous ISAM parameter lists will cause unpredictable results.
- Between a SELECT and a RELEASE, neither READ and GET nor WRITE and PUT may be intermixed.
- The caller may not provide his own DCB.
- The FHCW will be modified in order to convert the call to its VSAM equivalent.
- There is no equivalent to a QISAM physical block once the file has been converted to VSAM. All VSAM data records are equivalent to ISAM logical records. This means that users processing the file via READ in one subsystem and GET in another will both retrieve what would have been an ISAM logical record.

Figure 33 describes return codes when ISAM/VSAM compatibility is used.

Chapter 7

USING THE OUTPUT UTILITY

7.1 CONCEPTS

The Output Utility is a subsystem that processes messages destined for terminals operating under control of Intercomm. It is responsible for completing any device-dependent formatting requirements in a message before passing it to the teleprocessing interface (FESEND) for eventual transmission to the terminal device. It also checks the operational status of destination terminals. Should it find a destination terminal not operational, it will redirect messages to an alternate terminal, if one has been named for that particular destination terminal. Otherwise, the Front End will intercept a message to a nonoperational terminal and queue it in the output queue assigned to that terminal to await its availability. If an alternate terminal name has been provided to the Front End Network Table, and the alternate can receive output, then the Front End will dequeue the message queued for the nonoperational primary terminal and send it to the alternate as soon as possible (useful primarily for non-functional printers).

7.2 PROCESSING

An application subsystem may create four different types of output message text, identified by a value in the message header VMI field (MSGHVMI):

- Preformatted (VMI=X'57' or C'P')

Text consists of both data and device control characters. All spacing and other formatting (titles, column headings, etc.) is included in the message text. Output processing consists merely of passing the message to the Front End via FESEND. If the destination terminal (MSGHTID) is the name of a broadcast group, rather than an individual terminal, a separate message is created for each terminal of the group. Except for broadcast terminal-ids, subsystems should use the service routine FESEND, which is more efficient than queuing via Output.

- Formatting Required, Variable Text (VMI=X'50' or C'0')

Text consists of a string of character data items to be inserted into a final message format defined by an Output Format Table (OFT) entry. Each data field is prefixed with an item code and length prefix, and an occurrence factor (if a repetitive field), to identify the field. The OFT defines the position and content of titles, headings, etc., and defines the position where data fields from the message text are to be inserted. Output formats the final message, adding device-dependent control characters, and performs broadcast group processing, as described above.

- Formatting Required, Multiple Segments (VMI=n)

This form is used when multiple messages are to be created for the same hardcopy terminal (such as a printer) and interleaving of other messages for the same device is not desired. The text is variable format as described above. The VMI code for the first (or header) segment is X'51' or C'1'; for intermediate segments is X'52' or C'2' or X'5C' or C'4' depending on line types desired; and for the final segment is X'53' or C'3'. The final segment must be queued, even if no intermediate segments are created, in order that Output may release the terminal for other messages.

- Formatting Required, Fixed Text (VMI=X'72' or C'S')

Text consists of fixed length text fields in character or arithmetic format. This type of message is routed to the Change/Display Utility, where it is converted to a Variable Text message and routed to the Output Utility. The fixed text is described to Change/Display by a Format Description Record (FDR). The first twelve bytes of the fixed format text identify the particular FDR which details the fixed fields of the message. Byte 9 within this header provides the segment type (see Figure 34).

The application subsystem creates its output message (header and text) and directs the message to either the Output Utility or the Change/Display Utility by calling the service routine COBPUT. The receiving subsystem codes and VMI in the message header specify the destination subsystem and message text formatting requirements. Figure 34 summarizes message header specifications. In addition, the MSGHQPR field in the message header must be set to C'2' if the originating subsystem might process segmented input.

The sample subsystem in Chapter 12 provides examples of using the Output and Change/Display Utilities. For complete details regarding the Output Utility and Change/Display Utility, refer to the Utilities Users Guide.

OUTPUT Message Type	Message Header Fields			Change/ Display Prefix
	MSGHRSCH	MSGHRSC	MSGHVMI	
<u>Preformatted</u> (device-dependent)	X'00'	C'U'	X'57' or C'P'	N/A
<u>Variable Text Formatting:</u>			X'50' or C'0'	
<u>Single Segment Messages:</u> character format for item code, length (and occurrence number)	X'00' or C'0'	C'U'		N/A
binary format for item code, length (and occurrence number)	C'U'	C'U'		N/A
<u>Multi-Segment Messages:</u> character format first segment	X'00' or C'0'	C'V'	X'51 or C'1'	N/A
detail segment - repetitive data items			X'52' or C'2'	
detail segment - non-repetitive data items			X'5C' or C'G'	
final segment			X'53' or C'3'	
binary format first segment	C'V'	C'V'	X'51'	N/A
detail segment - repetitive items			X'52'	
detail segment -non-repetitive items			X'5C'	
final segment			X'53'	
<u>Fixed Field Formatting:</u>	X'00'	C'H'	X'72' or C'S'	
<u>Single-Segment Messages:</u>				C'0'
<u>Multi-Segment Messages:</u> first segment				C'1'
detail segment - repetitive items				C'2'
detail segment - non-repetitive items				C'4'
final segment				C'3'
NOTE: COBPUT converts character codes to the corresponding hexadecimal values for VMI codes, and MSGHRSCH to X'00'.				

Figure 34. Message Header Specifications for the Output Utility



Chapter 8

CONVERSATIONAL SUBSYSTEMS

8.1 GENERAL CONCEPTS

Conversational subsystems are defined as one or more subsystems designed to process more than one input message to complete a transaction. They effectively carry on a dialogue with the terminal operator, receiving an input message, retaining it and/or associated results of processing, issuing a response (perhaps a prompt for additional information), receiving another input message, retaining it, etc., until the transaction is complete. At the end of the conversation, appropriate files may be updated.

8.1.1 Conversational Applications

Typical applications which lend themselves to conversational processing are:

- Operator prompting (multiscreen input)
- Batch Data collection

Prompting, or multiscreen input, applications typically consist of dialogues in which the terminal operator enters an input message, the information is analyzed by the application subsystem and the results of processing are saved; the application subsystem then sends an output message to the terminal, prompting the operator for the next piece of information required. This dialogue continues until the application subsystem has obtained all the necessary information to complete processing for the given transaction.

Batch data collection may be conversational in that even though the input data is saved for later retrieval, the collecting application may need to return an error message requesting correction of invalid input data before saving the input record, or the application may need to request the input of a different type of record (for more detailed subsidiary information, intermediate totals, etc.).

8.1.2 Conversational Transactions

Conversational transactions involve the sending and receiving of more than one message in a terminal session. Each input message may be processed by related subsystems or by the same subsystem. A two-part conversational transaction is illustrated in Figure 35.

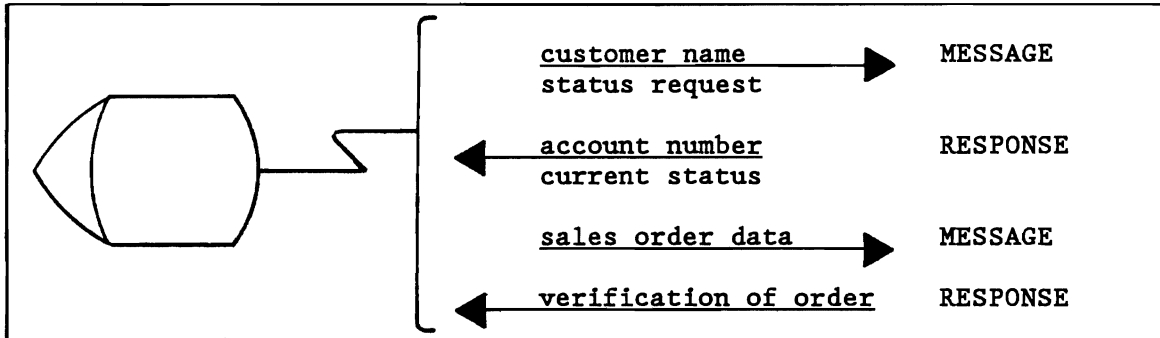


Figure 35. Typical Conversational Transactions

8.1.3 Retention of Information

Assume a conversation in which three input messages and three responses are necessary to complete the transaction. A terminal, a subsystem and a storage medium on which to save the input messages, and/or corresponding intermediate results of the processing, are necessary components in the conversational environment. In the example illustrated in Figure 36, the subsystem receives information and prompts the terminal operator for additional information until it obtains all the required data. This intermediate information is also stored either in core or on a disk data set. After the final input message is received and processed, appropriate files are updated, intermediate data is deleted, and a final response is issued.

Terminal XYZ	Subsystem ABC	Storage
Input Message 1--->	Receive, process and store---->	Input Message 1 + results
Output Message 1<---	Prompt for additional information	
Input Message 2--->	Receive, access Input Message 1<-- Process Also store Input Message 2----->	Input Message 1 + results Input Message 2 + results
Output Message 2<---	Prompt for additional information	
Input Message 3--->	Receive, analyze with prior <---- messages and results Update files, delete prior data	Input Message 1 & 2 + results
Output Message 3<---	Final response	

Figure 36. Input Message Data Retention During a Conversation

8.2 IMPLEMENTING CONVERSATIONAL SUBSYSTEMS

Conversational subsystems may be implemented in several ways, each characterized by the retention of initial and subsequent input and processing results. The method of retention differs, depending upon the method of implementation chosen.

Control of the conversation, or the retention of the input messages and/or corresponding results of processing may be accomplished by using any one of the following methods of implementation:

- The User SPA (User Extension to System Parameter List)
- The Store/Fetch Facility
- The Dynamic Data Queuing Facility
- The CONVERSE Service Routine
- The Table Facility (instead of the DDQ Facility)

In addition to the retention of the input environment, conversational subsystems have design considerations with respect to file updates and control of input verbs. These design considerations are discussed following a review of the first four methods of retention of input messages/data and corresponding results of processing.

Intercomm provides Front End conversational support to ensure that duplicate input from the same terminal is not processed. This is accomplished by defining applicable verbs and interactive terminals as conversational in the Front End tables. See the Operating Reference Manual.

8.3 SAVING INFORMATION IN USERSPA

The user extension to the SPA is called USERSPA and is accessible to all Intercomm subsystems since the SPA is the second entry parameter to all subsystems. The SPA is a 500-byte core-resident table. The user extension to the SPA begins at the 501st byte and may include application-oriented areas, such as tables, counters, and switches for application subsystem use. Thus, the size of USERSPA is installation-dependent. The user portion of the SPA is optionally checkpointable and can be restored at system restart time.

A portion of USERSPA may be divided into sections associating table space for each terminal, as illustrated by Figure 37. Each terminal-oriented area might be used for control data during conversational processing, until the conversation with that terminal completes.

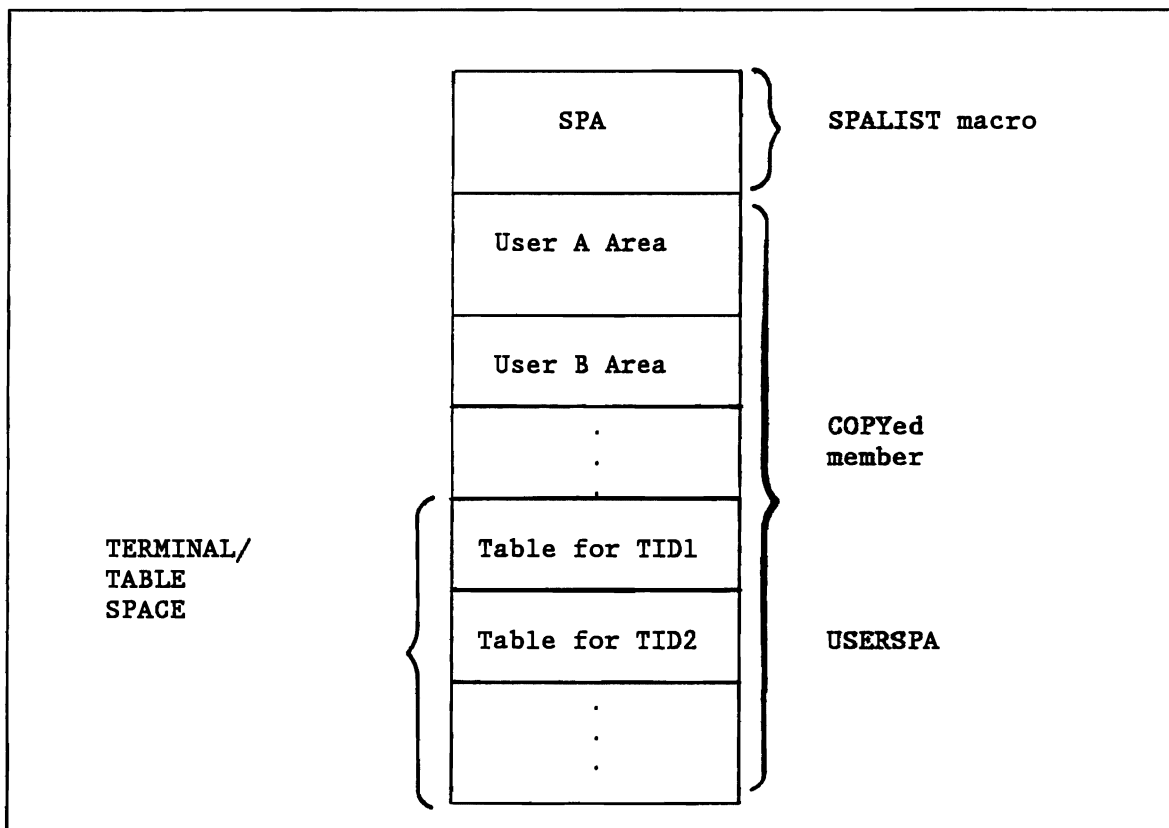


Figure 37. User and Terminal Table Space in the USERSPA

The SPA is expanded by updating the Assembler Language member USERSPA on the system release library SYMREL. The updated version should be stored on SYMUSR. When assembling INTSPA, USERSPA is copied as the last entry in the SPA Csect. Therefore, any user additions would be referenced beginning with the 501st byte. Any such additions should ordinarily be coordinated through the System Manager, as most application subsystems could be affected.

In the linkage section definition of SPA, as shown in Figure 38, three different applications have their own 50-byte areas defined: (USERA-AREA, USERB-AREA, USERC-AREA) plus a table for their common use (COMMON-TABLE). The Assembler Language member USERSPA for this example would contain a definition of an area corresponding to OURSPA. OURSPA could be defined as a systemwide COPY member for all COBOL routines, to be copied into the Linkage Section following the INTSPA statement.

```

01 SPA.
  02 INTSPA          PICTURE X(500).
  02 OURSPA.
    .
    .
  04 COMMON-TABLE   PICTURE X(200).
  04 USERA-AREA     PICTURE X(50).
  04 USERB-AREA.
    06 COUNT-FIELD1 PICTURE S9(8) COMP.
    06 ON-OFF-SWITCH PICTURE X.
    06 FILLER        PICTURE X(45).
  04 USERC-AREA     PICTURE X(50).

```

Figure 38. Sample USERSPA Declaration Within a Subsystem

The following chart summarizes the advantages and disadvantages of the USERSPA method of implementation of conversational processing.

<u>Advantages</u>	<p>Information saved in Core; no I/O overhead.</p> <p>Accessed easily.</p> <p>Checkpointable and restorable at restart.</p>
<u>Disadvantages</u>	<p>The entire USERSPA is accessible to all Intercomm subsystems. Therefore a problem of control develops with respect to the possibility of destruction of data by another subsystem, or security problems.</p> <p>Updating and maintenance of USERSPA may require recompiling all subsystems which reference it.</p> <p>A potentially large area of storage must be allocated.</p> <p>Addressability, if area larger than 3596 bytes.</p>

8.4 SAVING INFORMATION WITH STORE/FETCH

Conversational information may be stored and later retrieved (either in storage or on a disk data set) by the Store/Fetch Facility. Information is retained via the STORE function, and retrieved via the FETCH function. The storage space may be released via the UNSTORE function. Saved information may also be updated.

An operator prompting type of conversation involving one terminal and one or more application subsystem(s) could use Store/Fetch very efficiently for retaining information. Store/Fetch performs its function upon data strings. Data strings are logical entities of information (input messages to be retained or whatever other data the user intends to save), which are identified by unique user-defined keys. The information is accessible only to those subsystems which call a Store/Fetch service routine naming the data string by its unique key, which could include the current terminal-ID from the input message header. Therefore, there is more control over the information than there would be if it were to be saved in the USERSPA. The data strings are classified as either transient, semipermanent or permanent. The differences between these classifications are as follows:

Disposition	Availability	Storage Medium
Transient	Not available across restart	Core or disk
Semipermanent	Available across restart	Disk
Permanent	Available across every system start until explicitly unstored	Disk

In conversational processing, permanent data strings should not be used. As to whether to use transient or semipermanent strings, the user must decide whether the information is critical enough to be preserved across system restart. If so, the data strings would be classified as semipermanent and would reside on disk. At restart time, the operator could then resume a conversation at the point of failure if subsystem logic can determine when the conversation was interrupted. If stored data is specified as transient, data is eligible to reside in core. Processing would thus be speeded up, as I/O overhead would be eliminated. At restart time, the operator would then start the conversation from the beginning.

Detailed information on Store/Fetch, including the interface between application subsystems and the Store/Fetch service routines, may be found in Store/Fetch Facility. Application subsystem logic must determine whether the input message in progress is initial, intermediate or final. This determination is necessary to assure that the proper calls to Store/Fetch are issued when data is to be saved or retrieved. Once the determination is made, Store/Fetch may be used to manage the conversational information as shown in Figure 39.

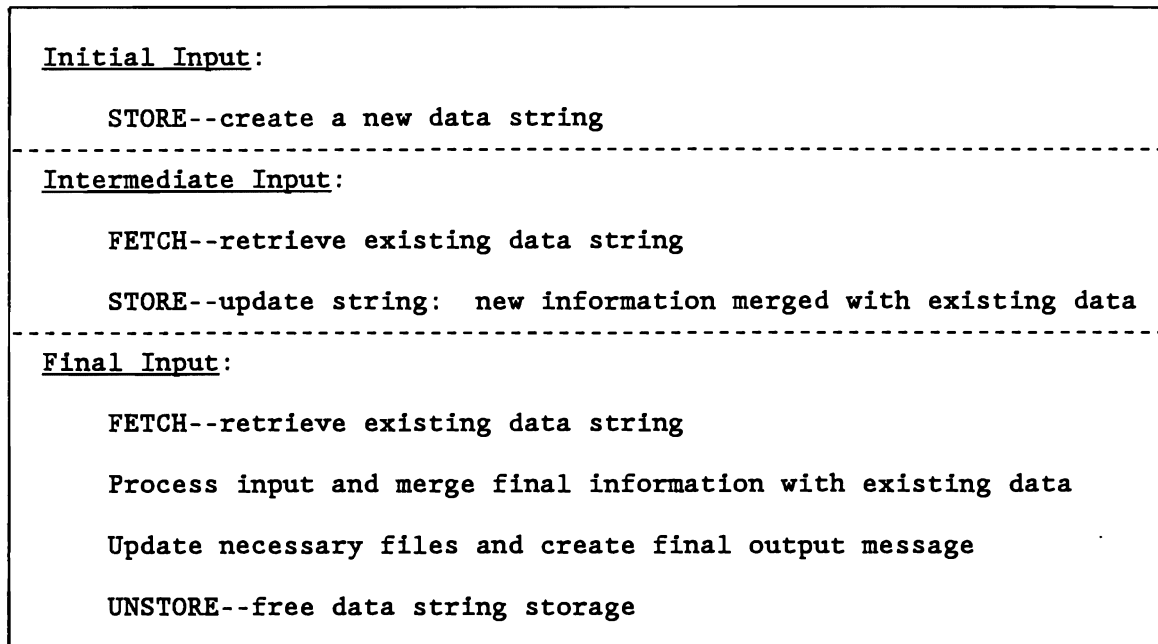


Figure 39. Conversational Processing Using Store/Fetch

Subsystem processing logic can be simplified by using one or more of the following techniques:

- A 'string-not-found' return code from a FETCH request indicates initial input (no intermediate data stored).
- A FETCH with the Delete option forces restart of the conversation from the beginning if the system fails, or the subsystem times out or program checks before the STORE of the intermediate data can be done. This technique also saves Store/Fetch and core storage resource overhead.
- The STORE of the intermediate data should be done after the output message is processed.
- File record(s) should not be updated until all intermediate data is collected. At this time the record(s) should be retrieved for update (exclusive control) and checked for external updates by unrelated processing since the conversation began.
- Do not send the final confirmation output message until successfully updating the file(s).

8.5 SAVING INFORMATION ON A DYNAMIC DATA QUEUE

The Dynamic Data Queuing Facility (DDQ) is a Special Feature available to Intercomm users. Detailed specifications on using DDQ may be found in Dynamic Data Queuing Facility. A DDQ provides the application subsystem with the ability to dynamically create, retrieve and delete logical data sets (or queues) of records on a BDAM data set. As illustrated in Figure 40, more calls are required to interface with the DDQ routines than are required to interface with Store/Fetch to obtain the same functions. However, a DDQ provides the ability to save several related data strings as a type of sequential file. The entire DDQ can then be processed by another subsystem or postponed for batch processing. A DDQ is most effectively used, not as a means for temporary storage of data during a conversation, but as a means for accumulating conversational results for subsequent processing, that is, for data collection. This facility can also be used for collecting data from related conversations with more than one terminal.

The data queues may be either transient, single-retrieval transient, semipermanent or permanent. Single-retrieval transient queues cannot be read more than once. This type of DDQ, therefore, would not be suitable for conversational processing. The other queue types are distinguished by the following characteristics:

Queue Type	Characteristics
Transient	<p>Must be passed to another subsystem or freed.</p> <p>Cannot be retrieved later.</p> <p>Not preserved across restart or normal startup.</p>
Semipermanent	<p>Retrieved at a later point in time via a user-provided Queue Identifier (QID).</p> <p>Extra I/O overhead is involved in saving the queue.</p> <p>Can be freed by user request.</p> <p>Queue must be completed (closed) in order to be preserved across restart.</p> <p>Existing semipermanent queues freed at normal startup.</p>
Permanent	<p>Same characteristics as semipermanent except that permanent queues are always preserved across any Intercomm start, warm or cold, if closed at least once.</p>

Figure 40 illustrates typical use of DDQ facilities in conversational processing. The application subsystem logic must determine whether input is initial, intermediate, or final. Final input, in this example, causes the queue to be closed and passed to another subsystem for asynchronous or postponed file updating. Thus, the terminal operator, upon receipt of the final output message, can begin another conversation without waiting for file updates to occur. This technique is particularly useful for files which do not require up-to-date inquiry response such as order entry, personnel, etc.

<u>Initial Input:</u>		
QBUILD	-- Create a new queue	
QWRITE	-- Save input message and related data	
QCLOSE	-- Save the DDQ	

<u>Intermediate Input:</u>		
QOPEN	-- Open the queue	
QREADX	-- Read the record with intent to update	or QWRITE to add to the queue
QWRITEX	-- Update the record	
QCLOSE	-- Save the DDQ	

<u>Final Input:</u>		
QOPEN	-- Open the queue	
QREADX	-- Retrieve the record	or QWRITE to add to the queue
QWRITEX	-- Update the record	
QCLOSE	-- Pass the DDQ to another subsystem which will update files and free the queue.	
Issue final output message.		

Figure 40. Conversational Processing Using Dynamic Data Queuing

8.6 SAVING INFORMATION VIA THE CONVERSE SERVICE ROUTINE

The final method of retaining information for a conversation is to use the Intercomm system service routine CONVERSE. The CONVERSE routine is called by an application subsystem when input from the same terminal is required to continue processing a transaction. The application subsystem thread stops processing until the next input message is received from that terminal. Control is returned to the next sequential instruction following the call to CONVERSE.

Application subsystems are designed more easily with CONVERSE, as it is simpler to control the sequential order of the messages. However, the use of CONVERSE is not encouraged, as it ties up Intercomm resources. Dynamic working storage associated with the initial and subsequent input messages is retained during the call to CONVERSE. Storage requirements for subsystems would be greater than when other conversational techniques are used, because one subsystem contains logic for all message types of a conversational transaction. It is far more efficient to design conversational subsystems which retain control only for the amount of time necessary to process one message than to tie up system resources while each input message in the conversation is in turn received, kept, analyzed and responded to in one execution of one application subsystem. When CONVERSE is used, dynamically loaded subsystems remain in storage until all "conversations in progress" have terminated. Intercomm restart processing of such subsystems restarts the conversation from the beginning. All intermediate messages are discarded.

The saving of information in the USERSPA or in a Store/Fetch data set or in a DDQ or Table Facility table does not require an application subsystem to contain logic for time-outs. The use of CONVERSE does. If the next input message is not received in the time limit specified by the user, a time-out occurs, which must be handled by subsystem logic.

An example of the use of CONVERSE in a two-part conversation is illustrated in Figure 41.

NOTE: CONVERSE is not supported for VS COBOL II, nor for OS/ANS COBOL subsystems loaded above the 16M.

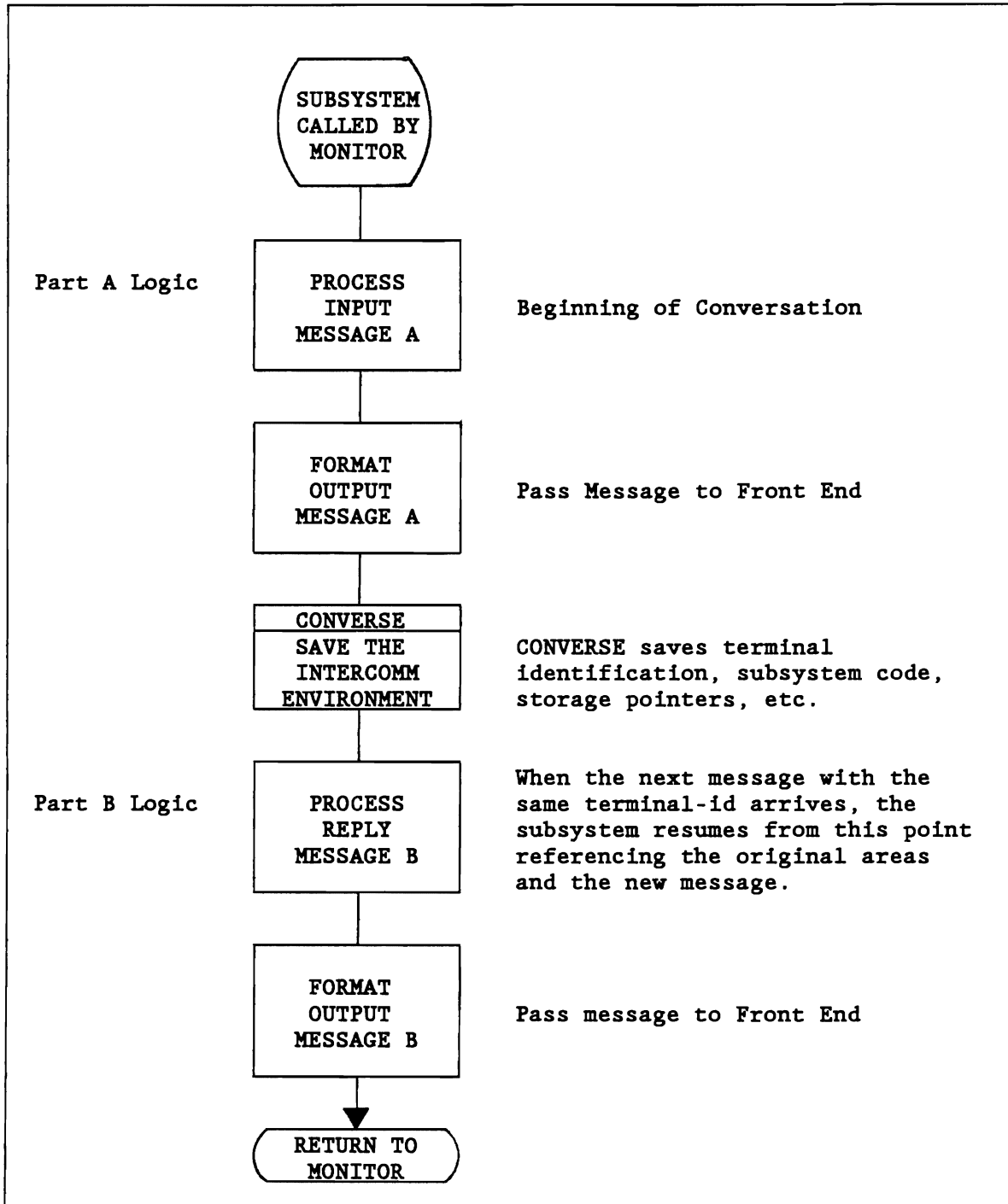


Figure 41. Conversational Subsystem Logic Using Converse

8.6.1 Subsystem Design Using CONVERSE

The Intercomm system service routine CONVERSE is called when awaiting additional input in response to some prompting message. Since any interval may elapse before the next message is received, CONVERSE will save information in its own control table for each conversation and return to the Subsystem Controller while waiting for the response.

The call to CONVERSE specifies a time limit within which a reply message should be received. If it is not received during the specified interval, then the subsystem is entered at the next instruction following the call to CONVERSE and its message parameter is adjusted to point to a time-out message supplied by CONVERSE. That message (header plus text) could then be switched to the Output Utility or FESEND. The terminal identification in the header is that of the non-responding terminal. A zero value for the time limit will bypass the automatic time-out feature.

Coding format:

```
CALL 'COBREENT' USING CONVERSE, word, time.
```

where:

word

is the name of an aligned fullword (PIC 9(8) COMP SYNC) in the subsystem's DWS required by CONVERSE for work space.

time

is the name of an aligned fullword binary value indicating a limit (in seconds) within which a subsequent message is anticipated.

When processing resumes following the call to CONVERSE, the environment appears as it was before the call--except the input message parameter (unless there was a time-out) now points to the most recent message from the terminal. It will have been edited if specified for the verb's definition in the Front End Verb Table. The Intercomm return code area will contain a binary value in the low-order byte indicating the condition for return from CONVERSE (see Figure 42).

The CONVERSE program keeps track of conversational requests by terminal and subsystem, and separates messages accordingly. Hence, each unique subsystem thread may be in conversation with a different terminal.

It is the subsystem's responsibility to verify that the message received following the call to CONVERSE is actually the appropriate message expected in the logical sequence of the conversation.

Note that the CONVERSE routine may only be called from a 24-Amode OS/VS or ANS COBOL subsystem. Due to complications arising in reestablishing COBOL internal tables on return from the call to CONVERSE, it may not be called by a COBOL subroutine of the subsystem.

For example:

- Monitor calls COBOL Subsystem AA which calls CONVERSE (valid sequence of program logic).
- Monitor calls COBOL Subsystem BB which calls Assembler Language subroutine B1 which calls CONVERSE (valid sequence of program logic). However, if the new input message processed by the Assembler Language subroutine on return from the call to CONVERSE is freed by the subroutine or passed by it to another subsystem or FESEND, then the subroutine must zero the first word in the parameter list passed to it (see Assembler Language Programmer's Guide), and under Release 10, it must also zero the first word of the original parameter list passed by PREPROG to a reentrant COBOL subsystem. PREPROG's parameter list address is stored in the field ITCBPMSS in the Intercomm Thread Control Block (ITCB) whose address is obtained via the INTTCB macro with OPT=S (see Basic System Macros). The calling COBOL subsystem may then not reference the input message area or any of its data fields (except for data fields in its DWS passed as parameters to the BAL subroutine for storing message data and/or a copy of the new message header for the next output message). Note that the BAL subroutine may use the new return code address parameter to pass a code back to the COBOL subsystem, or the COBOL subsystem may test it for the CONVERSE return code on return from the BAL subroutine.
- Monitor calls COBOL Subsystem CC which calls COBOL subroutine C1 which calls CONVERSE (invalid sequence of program logic).

The COBOL subsystem may not use an old copy of the message header for a new output message. If the subsystem calling CONVERSE is compiled with the ANS4 or a VS compiler, the Intercomm input-message and return-code parameters may not be addressable after the call. Use the IBM SERVICE RELOAD verb immediately following the CONVERSE call to solve this problem.

Conversational subsystem logic must be designed with care regarding file access. Selected files should be released prior to the call to CONVERSE. If not, other subsystems accessing the same files or other messages in process in the same subsystem may "time out." This may occur because an operating system control block is associated with the access to the file and is not "freed" until the file is released. If a file is accessed prior to the call to CONVERSE and released after the call to CONVERSE a "lock out" situation may occur.

Return Codes	Meaning
0 (X'00')	<p>Normal return: the entry parameter input-message reflects the address of the new input message. The message will have been edited successfully if the Front End Verb Table shows editing required. (If editing is unsuccessful, error messages will be sent to the terminal, and the subsystem is not reactivated until either a subsequent input message is edited successfully or an automatic time-out occurs.)</p> <p><u>CAUTION:</u> The CONVERSE automatic time-out is not extended if a message is found in error by the Edit Utility.</p>
17 (X'11')	<p>No core available for CONVERSE control blocks; conversational mode not initiated.</p>
18 (X'12')	<p>Time-out expired. The entry parameter input-message reflects the address of an error message generated by CONVERSE. The message header contains the appropriate terminal identification. The message text is:</p> <p>*PMI*CONVERSE*ANTICIPATED MESSAGE NOT RECEIVED WITHIN USER SPECIFIED TIME INTERVAL</p>

Figure 42. CONVERSE Return Codes

Control of the conversational program environment is accomplished by Intercomm in different ways, depending on the subsystem's residency:

- Resident

The dynamic-work-space (DWS) for one message from a terminal is retained pending arrival of the next message from that terminal; the subsystem will continue to process messages from other terminals.

- Overlay Loaded

Same as above, except the loaded overlay region may contain other subsystems to process other messages during (and after) "CONVERSE time."

- Dynamically Loaded

Same as above, except the subsystem remains in core until all "conversations in progress" have terminated.

8.7 DESIGN CONSIDERATIONS IN CONVERSATIONAL PROCESSING

In order to ensure file integrity, conversational subsystems performing file and/or data base updates should be designed to perform the updates for the last message in the conversation. Alternatively, control may be passed (via message queuing) to a non-conversational subsystem to perform the updates.

8.7.1 Control of the Input to Conversations

Conversational subsystems expect ordered input. They must be designed to analyze input messages and to determine which message in the sequence has been received. Control of the input may be exercised by the terminal operator or by the application subsystem(s).

The terminal operator may be given a specific sequential list of messages to input at the terminal for a given verb or verbs. This method would probably be used for data collection applications, in which more messages are sent to the application subsystem than are received at the terminal. It could also be used for any conversational application in which the order of input is fixed.

The application subsystem may control the input sequence by analyzing an input message, processing it, and issuing a response informing the operator about the content or format of the next input message. The response may direct the operator to input another verb (that of a related subsystem). Subsystem-controlled input is good for conversations in which the "next" desired piece of information may vary depending upon the contents of a file record, or a table, or the setting of a switch in the area saved between subsystem activations.

8.7.2 Assigning a Verb to a Terminal

To eliminate the requirement for an operator to key in a verb with each input message, the operator may enter a system control command message to LOCK a specific terminal to a particular verb. The Front End then prefixes that verb to each input message from that terminal. The operator may enter another control message, UNLK, to unlock the terminal from the verb. See System Control Commands.

The LOCK/UNLK commands processed by the Front End can also be issued by a subsystem. When a LOCK is in effect, all subsequent messages from the specified terminal will be automatically prefixed by the verb specified in the LOCK command. This LOCK remains in effect until UNLK is issued. With LOCK in effect, some advantages are:

- The terminal operator does not have to keep reentering the same verb.
- A new verb cannot be entered during the conversation.

Either the subsystem or the operator may control the input sequence by locking and unlocking the terminal to different verbs at different points in, or at the end of, the conversation.

Optionally, the Intercomm AUTOLOK feature may be defined for the verb in the Front End Verb Table, which dictates that when that verb is input from the terminal, the terminal is to be automatically locked to that verb. Subsequently, the terminal is to remain locked until specifically UNLKed by the operator or processing subsystem.

The format for the LOCK/UNLK commands (message text) is as follows:

```
LOCK$TPUxxxxx$vvvv@
UNLK$TPUxxxxx@
```

where:

xxxxx

is the five-character terminal identification

vvvv

is the four-character verb

@

is the end-of-transmission character (X'26')

\$

is the system separator character as defined for the installation.

The preformatted message constructed by a subsystem must be prefixed with the standard message header for FESEND (MSGHRSCH=X'00',MSGHRSC=X'00',VMI=X'57'). This message is passed to the Front End via FESENDC (see Chapter 9) and the LOCK or UNLK takes place. No response message is sent to the terminal when such processing is requested by a subsystem.

Chapter 9

USING INTERCOMM SERVICE ROUTINES AND FACILITIES

9.1 REENTRANT COBOL INTERFACE ROUTINE (COBREENT)

COBREENT is the interface routine called by all COBOL subsystems in order to maintain reentrancy during execution of a subroutine which potentially causes an I/O operation or gives up processing control to the Intercomm Dispatcher. The application program calls COBREENT specifying which subroutine is to be called (system program or user routine) and the appropriate parameters to pass to it. COBREENT saves registers, chains save areas, saves the program's Task Global Table (if OS/ANS COBOL) or it's THDCOM (if VS COBOL II) in a dynamic storage area, saves the COBOL save area, and saves the entry parameters for the called program. COBREENT then calls the specified subroutine. On return from that subroutine, COBREENT restores the environment and returns to the calling program. Coding format:

```
CALL 'COBREENT' USING routine-code, parameters.
```

where:

routine-code indicates the routine entry to be called.

parameters is the actual parameter list to be passed to the called routine. A maximum of ten parameters is recommended. Intercomm service routines require less than ten parameters; user subroutines should be designed with this limit in mind. The limit on the number of parameters passed by VS COBOL II programs is 64. If the calling subsystem (or subroutine) may be loaded above the 16M line, then all referenced parameters must be in the caller's DWS (have a 24-Mode address), except they may be in the Working-Storage Section if the caller is VS COBOL II.

Routine-codes name halfword offset values into the REENTSBS table of routine addresses. Offsets 0 through 100 are reserved for Intercomm system routines. Offsets 104 and up may be used for user subroutines. Figure 43 lists the routine-codes assigned as identifiers for Intercomm service routines in the released REENTSBS table. The COPY member (of routine-codes) for COBOL subsystems and subroutines is named ICOMSBS and is illustrated in Appendix B. See also Chapter 3 for sample coding using the ICOMSBS table. The hard-coded (with a VALUE clause) routine-code may be in the caller's Working-Storage Section of all COBOL programs.

Specifications and coding criteria for user subroutines are described in Section 7 of this chapter.

```

REENTSBS1 CSECT
*  NEGATIVE OFFSETS ARE USED BY SPECIFYING AN OFFSET ENDING IN B'11',
*  WHICH IS INCREMENTED BY 1 AND COMPLEMENTED TO OBTAIN TRUE OFFSET
*  BY COBREENT AND PMIPL1.
    SUBMODS NAME=INITLU6      OFFSET -104, CODED AS 103
    SUBMODS NAME=INTSORTC    OFFFSET -100, CODED AS 99
    SUBMODS NAME=DWSSNAP     OFFFSET -96, CODED AS 95
    SUBMODS NAME=MAPFREE     OFFFSET -92, CODED AS 91
    SUBMODS NAME=FECMRLSE   OFFFSET -88, CODED AS 87
    SUBMODS NAME=FESEND      OFFFSET -84, CODED AS 83
    SUBMODS NAME=FESENDC    OFFFSET -80, CODED AS 79
    SUBMODS NAME=ALLOCATE   OFFFSET -76, CODED AS 75
    SUBMODS NAME=ACCESS     OFFFSET -72, CODED AS 71
    SUBMODS NAME=MAPURGE    OFFFSET -68, CODED AS 67
    SUBMODS NAME=MAPCLR     OFFFSET -64, CODED AS 63
    SUBMODS NAME=MAPEND     OFFFSET -60, CODED AS 59
    SUBMODS NAME=MAPOUT    OFFFSET -56, CODED AS 55
    SUBMODS NAME=MAPIN      OFFFSET -52, CODED AS 51
    SUBMODS NAME=INTUNSTO   OFFFSET -48, CODED AS 47
    SUBMODS NAME=INTSTORE   OFFFSET -44, CODED AS 43
    SUBMODS NAME=INTFETCH   OFFFSET -40, CODED AS 39
    SUBMODS NAME=FECMFDBK   OFFFSET -36, CODED AS 35
    SUBMODS NAME=FECMDDQ    OFFFSET -32, CODED AS 31
    SUBMODS NAME=QWRITEX    OFFFSET -28, CODED AS 27
    SUBMODS NAME=QREADX     OFFFSET -24, CODED AS 23
    SUBMODS NAME=QWRITE     OFFFSET -20, CODED AS 19
    SUBMODS NAME=QREAD      OFFFSET -16, CODED AS 15
    SUBMODS NAME=QCLOSE     OFFFSET -12, CODED AS 11
    SUBMODS NAME=QOPEN      OFFFSET -8, CODED AS 7
    SUBMODS NAME=QBUILD     OFFFSET -4, CODED AS 3
    ENTRY REENTSBS
REENTSBS DS      OA              ALLOW FOR NEGATIVE OFFSETS
DC      A(REENTEND-REENTSBS-4)  REQUIRED
SUBMODS NAME=SELECT         CODE 4-  FILE SELECT
SUBMODS NAME=RELEASE        CODE 8-  FILE RELEASE
SUBMODS NAME=READ           CODE 12- FILE READ
SUBMODS NAME=WRITE          CODE 16- FILE WRITE
SUBMODS NAME=GET            CODE 20- FILE GET
SUBMODS NAME=PUT            CODE 24- FILE PUT
SUBMODS NAME=RELEX          CODE 28- RELEASE EXCL. CONTROL
SUBMODS NAME=FEOV           CODE 32- FILE FEOV
SUBMODS NAME=TABUILD        CODE 36- TABLE BUILD
SUBMODS NAME=TABOPEN        CODE 40- TABLE OPEN
SUBMODS NAME=TABPUT         CODE 44- TABLE PUT
SUBMODS NAME=TABGET         CODE 48- TABLE GET
SUBMODS NAME=TABSORT        CODE 52- TABLE SORT
SUBMODS NAME=TABEND         CODE 56- TABLE END
      .                      .
      .                      .
                                (Codes 60-64
                                are reserved)

```

Figure 43. COBREENT Routine Pointers (REENTSBS) (Page 1 of 2)

```

SUBMODS NAME=COBPUT      CODE 68- COBOL MESSAGE SWITCHING
SUBMODS NAME=MSGCOL      CODE 72- MESSAGE COLLECTION
SUBMODS NAME=COBSTORF    CODE 76- COBOL STORFREE
SUBMODS NAME=CONVERSE    CODE 80- CONVERSE
SUBMODS NAME=DBINT       CODE 84- DATA BASE REQUEST
SUBMODS NAME=LOGPUT      CODE 88- LOGPUT
SUBMODS NAME=PAGE        CODE 92- PAGE ROUTINE
SUBMODS NAME=GETV        CODE 96- VSAM GET
SUBMODS NAME=PUTV        CODE 100-VSAM PUT
*****
**      INSERT      USER      SUBMODS      MACROS      HERE      **
*****
COPY  USRSUBS
REENTEND EQU *                REQUIRED AFTER LAST SUBMODS
ENTRY REENTEND
REENTSBL CSECT
END

```

Figure 43. COBREENT Routine Pointers (REENTSBS) (Page 2 of 2)

9.2 INTERSUBSYSTEM QUEUING (COBPUT)

COBPUT is called to queue a message for a user or Intercomm subsystem. Queuing is controlled by the Receiving Subsystem Code fields in the message header. If segmented input messages may be processed, set the MSGHQPR field in the header to C'2' before calling COBPUT. If the Edit Utility is used in the system, ensure the VMI field (MSGHVMI) is non-zero so that an attempt to edit the message for/by the receiving subsystem is not made.

Coding format:

```
CALL 'COBREENT' USING COBPUT, message, return-code.
```

where:

message is the label of the first position of the message (header + text) to be queued

return-code is the label of a two-byte character field where COBPUT will place a return code.

COBPUT copies the message to be queued to a new area of dynamic storage, converting variable character format message text and header fields as necessary if the Receiving Subsystem Code is for the Output Utility (see Figure 34). COBPUT then calls Message Collection (MSGCOL) to accomplish the queuing of the message. Figure 44 lists COBPUT return codes.

The original message remains in the calling program's Dynamic Working Storage. If the message has not been processed or queued successfully, the subsystem may attempt to recover, or simply return to the Subsystem Controller with a return code of 8 or 12. Figure 45 lists various alternatives.

Return Code	Meaning
00	Message queued successfully <u>NOTE:</u> For Multiregion Facility users sending a message to another region, this return code signifies that the message was queued for sending to that region.
02	Item code, length, or line number greater than 255 in variable character data item prefix (Output Utility)
04	No room on subsystem queue, or (Rel 10) msg rejected for delayed subsystem--an entry was made on the system log (MSGHLOG=X'FC')
06	Nonnumeric item code (Output Utility)
08	No core for disk queue I/O area, or to copy message
10	N or R omitted in variable character data item prefix
12	I/O error on disk queue
14	COBPUT has detected a message length too short to convert character item codes and lengths
16	Invalid subsystem code--an entry was made on the system log (MSGHLOG=X'FB')
28	DVASN system routine could not reserve a device (on first segment of multi-segmented messages only)
<u>NOTE:</u>	A non-zero return code means the message was neither queued nor processed.

Figure 44. COBPUT Return Codes

Return Code	Alternative Action
02, 06, 10, 14, 16	Program error: no recovery action. Correct the invalid fields and recompile program.
04, 08	Requeue the original input message for reprocessing by the currently executing subsystem via calling COBPUT referencing the input message and the currently executing subsystem, or follow action for Return Code 28.
12	No recovery action: return to Subsystem Controller with return code 12.
28	Attempt a time delay and call COBPUT to attempt queuing of the message again.

Figure 45. Recovery From COBPUT Errors

9.3 INPUT MESSAGE SWITCHING (MSGCOL)

COBPUT is called to queue an output message to activate another subsystem. It copies the message from the Dynamic Working Storage area of the calling subsystem to a new dynamic area and calls Message Collection. Thus, the output message area within the dynamic-workspace of a subsystem is reusable upon return from COBPUT.

The logic of an application subsystem might be such that the input message is modified within its dynamic area to become an output message to switch to another subsystem. To do this, the length of the input message may not be increased (data may not be added). If the length is shortened by 8 bytes or more, see the next section on freeing the remainder, and adjusting MSGHLEN in the header. Queuing the message for the next subsystem is then done by calling Message Collection (MSGCOL), instead of COBPUT; Message Collection then owns and is responsible for the management of the message area. All queuing is controlled by the receiving subsystem code fields (MSGHRSCH and MSGHRSC) in the message header. When returning to the System Monitor, the subsystem return code must be set to 900 (see Figure 14).

Coding format:

```
CALL 'COBREENT' USING MSGCOL, message, SPA-addr, return-code.
```

where:

message is the label of the input message to be queued.

SPA-addr is the second entry parameter in the Linkage Section.

return-code is a fullword computational field where COBREENT will place the return code from MSGCOL.

MSGCOL return codes indicate the result of the queuing. The return code is fullword binary and can therefore use the same field as the Intercomm return code. (See Figure 46.) Regardless of the result, the calling program no longer has any control over the area of dynamic storage occupied by the input message and must return a code of 900.

Return Code	Meaning
0	Message queued successfully
4	No room on queue (entry made on system log), or message rejected for delayed subsystem (Rel 10)
8	No core for disk queue I/O area
12	I/O error on disk queue
16	Invalid subsystem code (entry made on system log)

Figure 46. Message Collection Return Codes

Recovery action for unsuccessful queuing might be to return to the System Monitor with a return code of 8 or 12. A message would then be sent to the terminal that originated the input message being processed, if USRCANC (PMICANC) is included in the Intercomm linkedit.

9.4 FREE DYNAMIC (MESSAGE AREA) STORAGE (COBSTORF)

COBSTORF may be called to free an area of dynamic-work-space not utilized for a message passed to another subsystem and not to be freed by the Subsystem Controller when the subsystem returns (see Section 9.4.1). COBSTORF may also be used to free the end of an input message area when the message text is shortened before being queued for another subsystem (see previous section).

Coding format:

```
CALL 'COBREENT' USING COBSTORF, area, length.
```

where:

area is the name defining the first (leftmost) position of the area to be freed.

length is the name of an aligned fullword containing a binary value indicating the number of bytes to free.

CAUTION: Dynamic storage is managed as doublewords. The area specified should be aligned on a doubleword boundary (COBSTORF will round up the address if not). The length specified should be a multiple of 8 (COBSTORF will round down the length if not). See also SYCTTBL macro, GET and FREE parameters, defining the DWS obtained and freed by the Subsystem Controller as described in Chapter 2. Note also that COBSTORF calls may not be used to free part of the DWS if DWS overflow checking is desired. When freeing part of an input message, only the rightmost portion may be freed and the remaining length must be stored in the first two bytes (MSGHLEN) of the message header before calling MSGCOL.

9.4.1 INTERSUBSYSTEM MESSAGE QUEUING VIA MESSAGE COLLECTION (MSGCOL)

Since a message created by a reentrant subsystem resides in dynamic-work-space, it is not a requirement that COBPUT be used to copy a message to be queued to another subsystem. Message Collection may be called directly, depending on the SCT specification for the subsystem (GET and FREE parameters of the SYCTTBL macro). This feature may not be used if DWS checking is in effect (see Chapter 3), nor if the message is in a VS COBOL II program's Working-Storage Section. For the coding format and return codes, see Section 9.3.

The SCT entry specifies the amount of core for dynamic-work-space obtained upon entry to a reentrant subsystem and also, the amount of core to be freed when that subsystem returns to the Monitor. They need not be equal. If unequal, the area of core remaining to be freed is the "leftmost" or first portion of the area obtained. The application programmer is then responsible for the "rightmost" area of dynamic-work-space. A new message may be created in that area (must start on a doubleword boundary) and queued for any other subsystem by calling MSGCOL. MSGCOL then owns and is responsible for that amount of core specified in the message header length field. Any remaining area of dynamic-work-space beyond (to the right of) the message area must be freed by the application subsystem by calling COBSTORF before returning (GOBACK) to the Monitor.

For example, consider two reentrant subsystems:

- Subsystem XX - SCT specifies:

-- 1024 bytes of dynamic-work-space obtained

-- 1024 bytes freed on return to the System Monitor

Must use COBPUT to queue any messages for other subsystems, that is, OUTPUT, etc.

- Subsystem YY - SCT specifies:

-- 1024 bytes of dynamic-work-space obtained

-- 512 bytes freed on return to the System Monitor

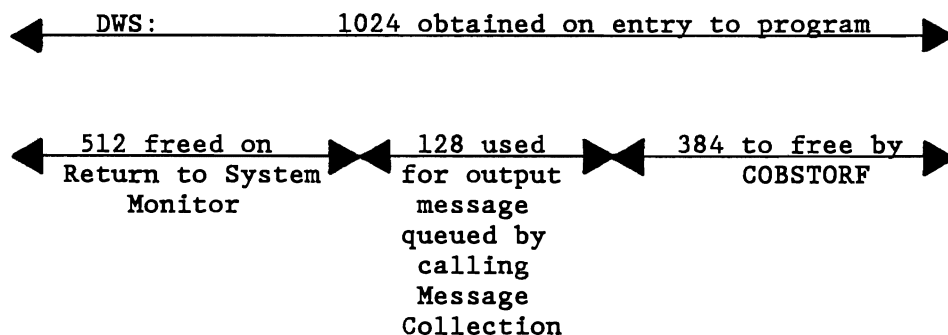
May use MSGCOL to queue a message for another subsystem if defined in last 512 bytes of the dynamic-work-space (DWS); or

May use COBPUT instead.

Must use COBSTORF to free any part of DWS not freed on return to System Monitor and not referenced by a call to MSGCOL.

To illustrate:

If subsystem YY queued a message for Output with MSGHLEN=128, subsystem YY is responsible for freeing the remaining 384 bytes of the 512 not freed by the System Monitor.



9.5 SEND MESSAGE TO FRONT END (FESEND)

FESEND is called to pass a message to the Intercomm Front End for transmission to a terminal. The message header field MSGHTID specifies the destination terminal or broadcast group name. The entry point FESENDC of FESEND is used by high-level language subsystems. FESENDC copies (from the caller's DWS) the message to be passed to the Front End to a new area of storage and proceeds via logic in the program FESEND. FESEND then requests queuing of the message on the associated terminal queue. If a broadcast group is specified, FESEND creates an individual message for each terminal of the group and requests queuing for each of those messages. All terminals in the broadcast group must be of the same type, as defined in the Back End Station and Device tables (see Chapter 2).

FESEND accepts two types of messages: preformatted (VMI=X'57') message text, which contains the control characters and data for transmission to the terminal except for start-of-text sequence(s) to be added by the Front End; and fully-formatted (VMI=X'67') message text, which contains all control characters and data ready for transmission to the terminal. (MMU produces fully-formatted messages.) If segmented input messages may be processed, set MSGHQPR to C'2' before calling FESENDC. If passing the message to the Front End is for any reason unsuccessful, the subsystem is notified by a return code, and recovery action may be taken.

FESEND tests whether messages sent to the Front End might be system commands or for control purposes. Such messages control Front End operation and generally cause no output to a terminal. Front End Control Messages (FECMs) are described later in this chapter. All system control commands and message text contents are documented in System Control Commands.

Coding format:

```
CALL 'COBREENT' USING FESENDC, message, return-code[, option-codes].
```

where:

message is the label of the output message (header and text) to be passed to the terminal queue.

return-code is the name of a two-byte character field where FESENDC will place a return code indicating whether or not processing was successfully completed.

option-codes is the name of an optional four-byte character field containing Front End processing codes as follows:

Byte 1: CRT Release option code:
 blank or X'00'--do not release (prevent screen overlay) next message (default)
 C'R'--release (allow overlay) next message to CRT
 C'C'--release next message, but do not cancel Front End conversational time-out

Byte 2: VTAM Response option code (overrides Front End Network Table definition for terminal):
 blank or X'00'--no override (default)
 C'D'--D1 response
 C'E'--E1 response
 C'F'--D2 response
 C'G'--E2 response

Bytes 3 and 4: Not used (set to blanks or binary zeros).

FESENDC return codes and possible recovery actions are listed in Figure 47. A nonzero return code means the message was not queued for the Front End. Return codes 16-24 should only occur during subsystem testing.

Return Code	Meaning
00	Message queued successfully.
04	Queue-full condition encountered; attempt a retry by invoking FESEND again.
08	Low-core condition encountered; attempt a retry by invoking FESEND again or return to Intercomm. (See Figure 14.)
12	I/O error (see Figure 14) encountered on disk queue; return to Intercomm.
16	Invalid terminal-ID; no recovery action required. Check with System Manager to verify terminal/broadcast group named in MSGHTID field.
20	Invalid VMI or syntax error in Front End control or command message text.
24	Invalid message header; return to Intercomm. See also error message MG602I and Snap 51.

Figure 47. FESENDC Return Codes

9.6 USER LOG ENTRIES (LOGPUT)

An application subsystem may require entries on the system log for many different situations:

- Application-dependent security violation or other error recording.
- Log entries rather than snaps used to trace the progress of a message while testing.
- Any application-oriented requirement for a record on the system log.
- Before- and/or after-image records of file updates (if not using the Intercomm File Recovery special feature).

User log entries are identified by unique codes in the message header log code field (MSGHLOG) and hence can be recognized by any batch program processing the log off-line. Messages to be logged consist of a standard 42-byte header and message text. The log code field (MSGHLOG) in the message header must be set to any value from X'41' to X'6F'. Logging is performed by calling the Intercomm system service routine LOGPUT. The date and time stamp in the message header (MSGHDAT and MSGHTIM) will be updated by LOGPUT prior to writing to the log. Log entries may subsequently be suppressed for later Intercomm executions by modifying the LOGTROUT translate table in the LOGPUT routine. Any message having a log code in the header which translates to X'FF' will not be logged.

The length of the record on the log is controlled by the value of MSGHLEN in the message header and must be at least 42. LOGPUT will not write out messages longer than the logical record size of the log (see INTERLOG JCL description in the Operating Reference Manual).

Coding format:

```
CALL 'COBREENT' USING LOGPUT, message.
```

where:

message is the label of the message (header plus text) to be logged.

There is no return code from LOGPUT.

9.7 CALLING USER SUBROUTINES FROM REENTANT COBOL SUBSYSTEMS

All subroutines called by an application subsystem must be called via COBREENT. Passed parameter values must be in 24-Amode storage (such as the caller's DWS), except they may be in the Working-Storage of a VS COBOL II caller. No other special conventions need be followed in order to call:

- An Intercomm system service routine.
- A user-coded Assembler Language (BAL) subroutine.
- A user-coded COBOL subroutine.
- A data base interface routine.

9.7.1 Defining User Subroutines to Intercomm

A user-coded subroutine (Assembler Language or COBOL) must be defined to Intercomm via coding of a SUBMODS macro in a user member USRSUBS which is copied at the end of the subroutine table REENTSBS (before REENTEND) at assembly time (see Figure 43). Resident, reentrant Assembler Language subroutines are defined by the NAME parameter of SUBMODS, all others via the LNAME parameter, plus additional parameters defining language, residency, etc. Additionally, the routine's reference name and corresponding index code should be added to ICOMSBS (see Appendix B) for easy access by subsystems when calling COBREENT. The SUBMODS macro is described in Basic System Macros.

9.7.2 Interfacing to User-Coded Assembler Language Subroutines

Assembler Language subroutines must be coded as reentrant if they may give up control to the Intercomm Dispatcher (via I/O requests, MMU requests, message queuing, etc.). When called from a COBOL program via COBREENT, standard linkage conventions are used. COBREENT issues a MODCNTRL macro to link to non-resident Assembler subroutines. At entry, register 13 points to the beginning of a 256-byte link/save area which precedes the DWS acquired for the COBOL program. Therefore, the caller's registers must be saved on entry to the Assembler subroutine, and reloaded before return, and save area chaining must be done. The COBOL link/save area may not otherwise be used by a called subroutine. An Assembler subroutine may not call a COBOL subroutine.

9.7.3 Interfacing to User-coded COBOL Subroutines

A reentrant COBOL subroutine is coded like a COBOL subsystem in that it uses a Linkage Section and a Dynamic Working Storage area, and it calls COBREENT to interface to Intercomm service routines and other user subroutines. Non-resident reentrant COBOL subroutines loaded above the 16M line under Release 10 must use the coding conventions described in Chapter 3. Subroutine calls may be nested, but must return to the caller, as illustrated previously in Figure 5.

The Linkage Section may optionally contain definitions for the input message (if not previously freed via a MAPIN call), the SPA, the SCT (SYCTTBL entry for the calling subsystem), and the Intercomm return code system parameter areas. Any, or all, of these parameters (in the above order) are requested via the SUBMODS macro definition of the subroutine. These must be the first 01 level definitions in the Linkage Section. The required Dynamic Working Storage area is defined via the GET parameter of the SUBMODS macro. The 01 level definition for the dynamic working storage area is coded after the system parameters (if requested) and before the 01 level definitions for parameters passed by the caller.

The Intercomm return code area may be used to pass a return code back to the calling COBOL subsystem because both the subroutine and the subsystem are referencing the same area via Linkage Section definitions. The subsystem may pass that return code back to the Intercomm Monitor (if standard Intercomm return code conventions are used by the subroutine) or may take action based on the return code and then change the passed value in the return code area to a standard Intercomm return code value. See the sample programs in Chapter 10. Coding conventions for subroutine interfaces prior to Intercomm Release 9.0 are defined in Appendix D.

9.8 FRONT END CONTROL MESSAGES

The Front End Control Message (FECM) facility provides three types of Front End control messages which may be used by application subsystems for:

- Front End data queuing (FECMDDQ)
- Front End feedback messages (FECMFDBK)
- Front End queue release (FECMRLSE)

A FECM is generated by an application program call to a service routine. The generated FECM message text is complete. The header field MSGHLEN has been set; bytes 3-42 are not modified. If the user has copied a valid header to the FECM message area prior to the call, only the sending subsystem codes (SSCH,SSC) and the VMI (X'57') must be set. The generated FECM must then be passed to the Front End by a call to FESENDC in the application program.

After a call to any Front End Control Message facility, a return code is placed in the first byte of the status word:

Return Code Value	Meaning
C'0'	FECM successfully created
C'8'	No storage for FECM processing (Assembler only)

9.8.1 Front End Data Queuing

Front End data queuing (FECMDDQ) works in conjunction with the Dynamic Data Queuing Facility. It provides the user with a more efficient way of handling groups of related output messages. An application may pass a Dynamic Data Queue (DDQ) to the Front End via a FECM. The DDQ contains messages to be sent to a terminal. This is a more efficient design approach than sending one message at a time to the Front End via FESEND, and prevents interleaving of unsolicited messages with those on the DDQ. This feature is particularly useful for printed reports. The messages on the DDQ must be preformatted (VMI=X'57') or fully formatted (VMI=X'67'). The Dynamic Data Queuing Facility manual contains detailed information on DDQ concepts, facilities and implementation, and specific design considerations for Front End Data Queuing. MMU uses this facility (FECMDDQ), when requested for multipage printer output. Coding format:

```
CALL 'COBREENT' USING FECMDDQ, status-word, fecm-area,
                        ddq-id[, ddq-disp].
```

where:

status-word is a 4-byte (fullword aligned) area required by the facility.

fecm-area is a 112-byte area to contain the FECM (header and text). The user should initialize the header prior to the call, probably by copying the input message header to this area.

ddq-id is the sixteen (16) byte DDQ identifier.

ddq-disp is a one-byte code indicating DDQ disposition after all messages are transmitted:

C'S' means SAVE the DDQ (required if MSGHTID is a broadcast group name)

C'F' means FREE the DDQ (default)

NOTE: The ddq-disp parameter may be omitted if the DDQ is to be freed after all the messages are transmitted (default). All of the above parameters must be in the DWS if the calling program is loaded above the 16M line.

9.8.2 Front End Feedback Messages

This type of FECM (FECMFDBK) is used by an application to determine that all prior messages queued for a terminal (before the FECM) have been transmitted. In this way, an application subsystem can be notified that certain critical messages have indeed been successfully transmitted.

Subsystem logic creates all normal output messages and passes them to the Front End (via FESEND, MMU, or by queuing messages for Output). Generation of a feedback message is then requested by a call to a FECM service routine. The feedback message is then processed in the same way as the other messages for the terminal (queued via FESEND or the Output Utility). When the Front End retrieves the feedback message, it is routed to the subsystem specified when the feedback message was generated rather than to the destination terminal.

Feedback messages may also be used in conjunction with Front End Data Queuing. A feedback message could be an intermediate, or the last, message on a DDQ passed to the Front End. If the DDQ was created via MMU (a MAPEND call option), then the feedback FECM must be created and queued by the subsystem on return from the MAPEND call. Coding format:

```
CALL 'COBREENT' USING FECMFDBK, status-word, fecm-area,
                    fecm-rsc, fecm-text.
```

where:

status-word is a 4-byte (fullword aligned) area required by the facility.

fecm-area is a 78-byte area to contain the FECM (header and text). The user should initialize the header area prior to the call, probably by copying the input message header to this area.

fecm-rsc is a two-byte receiving subsystem code (high/low) to specify the feedback message destination subsystem.

fecm-text is a 16-byte area containing the desired feedback message text.

9.8.3 Front End Queue Release

This type of FECM (FECMRLSE) allows the subsystem to override the normal Front End Logic for CRTs, which requires a one-for-one correspondence between input and output messages. When the release FECM is processed by the Front End, it causes a subsequent response message queued for the same terminal (as identified by MSGHTID in the FECMRLSE message header) to be transmitted immediately, rather than waiting for input (RLSE command) from the terminal operator. Because of protocol restrictions (HDFP) on VTAM Front End IBM SDLC 3270 CRT processing, the CRT release option for the first call to FESEND should be used (see Section 9.5) as a release; because if the terminal is already in send mode, it is necessary to turn the line around before sending the released message, which may confuse the terminal operator. The CRT release option locks the terminal in receive mode, preventing new input by the operator.

A release FECM might be used if a subsystem queues more than one output message to the CRT terminal due to a considerable amount of processing (file/data base I/O) being necessary between messages. The first message might be an immediate response to the terminal operator

indicating the input request is being processed, but allowing new input by the operator. Then, the second message (following the release FECM) is the ultimate result of the requested processing. A release FECM could also be used to force immediate transmission of a critical message to another CRT (other than the input terminal). Such processing should be used with caution because unsolicited messages can cause confusion for the terminal operator and may clear an existing screen format or displayed message. Coding format:

```
CALL 'COBREENT' USING FECMRLSE, status-word, fecm-area.
```

where:

status-word is a 4-byte (fullword aligned) area required by the facility.

fecm-area is a 60-byte area to contain the FECM (header and text). The user should initialize the header area prior to the call, probably by copying the input message header to this area.

9.9 IN-CORE TABLE SORT FACILITY (INTSORT) (Release 10 only)

If the Table Facility is not used to create an in-core table, then to sort a user in-core table, the INTSORT Facility (entry point INTSORTC for COBOL) is provided. Such a table might contain data stored in Store/Fetch strings or file data record via online transactions or offline processing. The table can have any number of fixed-length entries, and each entry can have a total size of 1 to 32767 bytes. The key to be sorted on can be anywhere within the first 256 bytes of the entry, but must be in the same place, and of the same length, in each entry. Coding format:

```
CALL 'COBREENT' USING INTSORTC, entries, entry-length, table,  
key-offset, key-length, return-code.
```

where:

entries is a 4-byte (fullword aligned) area containing the number of table entries in binary format.

entry-length is a 4-byte (fullword aligned) area containing the size of each entry (up to 32767) in binary format.

table is the name of the area containing the table to be sorted.

key-offset is a 4-byte (fullword aligned) area containing the offset (-1) in binary format of the key within each entry (value must be 0 if key at the beginning of the table entry; 1 if it starts in the second position of the table entry, etc.). May be 0 to 255.

key-length is a 4-byte (fullword aligned) area containing the length in binary format of the key (to be sorted on) of each entry (can be the same as entry-length). Length may be 1 to 256 depending on key-offset (key-offset+key-length must be less than 257).

return-code is a 4-byte (fullword aligned) area to contain the return code (in binary in the low-order byte) from INTSORTC, as follows:

Return Code	Meaning
X'00	INTSORT completed successfully (no duplicate keys)
X'04'	Number of entries less than 1 or table size greater than 16M-1
X'08'	Length of an entry is less than 1 or greater than 32767
X'12'	No Table name (address) supplied
X'16'	Key-offset greater than 255
X'20'	Key-length plus key-offset exceeds 256 bytes
X'24'	Successfully sorted table contains duplicate keys (entries).

For all non-zero return codes except 24, the sort is not executed.

9.10 OTHER INTERCOMM SERVICE FACILITIES

The following service routines for application programs are accessed via the following subroutine entry names listed in REENTSBS:

- MMU (MAPIN, MAPOUT, MAPEND, MAPCLR, MAPURGE, MAPFREE)
- Store/Fetch (INTSTORE, INTFETCH, INTUNSTO)
- DDQ (QBUILD, QOPEN, QREAD, QREADX, QWRITE, QWRITEX, QCLOSE)
- Page Facility (PAGE)
- DBMS (DBINT) - data base interfacing
- Dynamic File Allocation (ALLOCATE, ACCESS)
- Table Facility (TABUILD, TABOPEN, TABPUT, TABGET, TABSORT, TABEND)
- LU6.2 transaction invocation (INITLU6)

Code names for all routines are provided in the COPY member ICOMSBS (see Appendix B). Detailed documentation for use of the above facilities is provided in separate manuals (see Chapter 2). Special coding and call conventions for specific data base support are described in Data Base Management System Users Guide and vendor manuals.

Other service routines described at the end of Chapter 2 and in the Assembler Language Programmers Guide such as binary table search, ESS user-id search, dispatcher related routines, and data field search routines (when Edit and Output Utilities used), can be accessed by adding the entry name to USRSUBS with a SUBMODS macro (use NAME parameter only) and adding the name and offset code to ICOMSBS.

9.10.1 Features Accessible via Assembler Macros

Several Intercomm facilities are accessible only via a call to an assembler-coded subroutine which issues an Intercomm macro to use the facility. Such features include:

- Enqueue/Dequeue--to request exclusive or shared control of a resource (INTENQ, INTDEQ)
- Start/Stop--function control or status test (SSSTART, SSSTOP, SSTEEST)
- Write-to-operator--to issue a message to the CPU console (PMIWTO, PMIWTOR)
- Snap--to issue a snap of the passed program areas for debugging if DWSSNAP not used (Release 10 - see Chapter 3) (PMISNAP)
- Timed wait--to request a timed delay of subsystem processing if IJKDELAY not used (see Chapter 2) (INTWAIT)
- Asynchronous processing--dispatch a time-delayed routine, post or wait on an asynchronous processing routine (DISPATCH, INTPOST, INTWAIT)
- Acquire current time and/or date (INTTIME, GETDATE)
- Acquire device-dependent information about a terminal (EXTERM)
- Track user accounting information for SAM (USRTRACK)
- Convert hexadecimal fields to printable character (LAYOUT, HEXCON)
- Format subsystem codes for printing (SSCONV)
- Test authority of the currently signed-on (under ESS) user to use a logical function, such as Data Base access (SECTEST).

Note that use of most of these facilities will add to subsystem processing time (increase TCTV). Further documentation may be found in the Assembler Language Programmers Guide and Basic System Macros. GETDATE macro may only be used under Release 10.



Chapter 10

SAMPLE PROCESSING PROGRAMS

The sample program SQCOBOLA, shown in Figure 48, demonstrates coding of an OS/ANS COBOL subsystem which is either resident or dynamically loadable below the 16M line. To be eligible for loading above the 16M line, the MMU map group and map names would have to be copied to fields added to the DWS, and those new fields referenced for MMU calls which need those names in the passed parameter list.

The program processes an inquiry transaction (MURQ) containing a part number and a warehouse number for a stock status display. MMU is used to transform the incoming message into a fixed field format. The part number is transformed into a RBN for accessing a BDAM part description file (PARTFILE). The RBN and a part description record area are passed as parameters to a called (via COBREENT) COBOL subroutine SQCOBOLB, illustrated in Figure 49, which is eligible for residence anywhere. The subroutine retrieves the requested record from PARTFILE and passes back the File Handler return code to the calling subsystem via the Intercomm return code field.

Together, the part number and warehouse number provide a VSAM key for accessing a stock status file (STOKFILE). The File Handler is used for accessing both files. MMU is used for formatting an output display. Error messages, for conditions such as non-existent or erroneous warehouse or part numbers, or file I/O errors, are built within the program and formatted by MMU using an error map area.

The ICOMSBS, ICOMINMG and ICOMDWS basic Release 10 (see Appendix B) source text members defining the service routine pointers and Intercomm message header fields are COPY'd by the COBOL compiler. The COBLOGCH source text member used for terminal attribute and command override for MMU processing, and the symbolic map areas, are also copied into the program.

All required table entries, JCL, sample input messages and testing procedures, plus sample execution output, are illustrated in Chapter 11, "Subsystem Testing." The subsystem code used in the SYCTTBL macro to identify the sample subsystem is RQ. Intercomm's BTAM simulator is used for testing. Test messages are included to test as many error combinations as possible.

Chapter 12 illustrates a similar subsystem (without the COBOL subroutine) coded for the same purpose but using the Edit and Output Utilities, a COBPUT call, and Test Mode for testing. Chapter 13 illustrates SQCOBOLA redesigned for the VS COBOL II compiler (DWS fields moved to Working-Storage), and defines changes to Chapter 11 for testing in the VS COBOL II environment.

PP 5740-CB1 RELEASE 2.4		IBM OS/VS COBOL	
00001	000100	IDENTIFICATION DIVISION.	00010000
00002	000200	PROGRAM-ID. SGC3BOLA.	00020000
00003	000300	ENVIRONMENT DIVISION.	00030000
00004	000400	DATA DIVISION.	00040000
00005	000500	WORKING-STORAGE SECTION.	00050000
00006	000600	77 SLASH PIC X VALUE '//.	00060000
00007	000700	77 BDAM-READ-GOOD PIC X VALUE 'D'.	00070000
00008	000800	77 VSAM-READ-GOOD PIC X VALUE 'V'.	00080000
00009	000900	77 DD-STOCK PIC X(8) VALUE 'STOKFILE'.	00090000
00010	001000	77 DD-PART PIC X(8) VALUE 'PARTFILE'.	00100000
00011	001100	01 ID-GROUP-NAME PIC X(8) VALUE 'STKSTAT'.	00110000
00012	001200	01 ID-MAP-NAME PIC X(8) VALUE 'MAP1'.	00120000
00013	001300	01 ERR-MAP-NAME PIC X(8) VALUE 'ERRMAP'.	00130000
00014	001400	01 MESSAGE-TABLE.	00140000
00015	001500	04 MSG-A PIC X(12) VALUE 'PART NUMBER '.	00150000
00016	001600	04 MSG-B PIC X(11) VALUE ' NOT FOUND.'.	00160000
00017	001700	04 MSG-C PIC X(5) VALUE 'PART '.	00170000
00018	001800	04 MSG-D PIC X(24) VALUE	00180000
00019	001900	' NOT FOUND IN WAREHOUSE '.	00190000
00020	002000	04 MSG-E PIC X(20) VALUE '. MESSAGE CANCELLED.'.	00200000
00021	002100	04 MSG-F PIC X(17) VALUE 'MAP ERROR MCW IS '.	00210000
00022	002200	04 MSG-G PIC X(36) VALUE	00220000
00023	002300	'INVALID DATA: PARTNO MUST BE NUMERIC'.	00230000
00024	002400	04 MSG-H PIC X(35) VALUE	00240000
00025	002500	'INVALID DATA: WMSNO MUST BE NUMERIC'.	00250000
00026	002600	04 MSG-I PIC X(46) VALUE	00260000
00027	002700	'INVALID DATA: PARTNC AND WMSNO MUST BE NUMERIC'.	00270000
00028	002800	01 LOGICAL-DEVICE-DESCRIPTION COPY COBLOGCH.	00280000
00029	C	01 LOGICAL-DEVICE-DESCRIPTION.	
00030	C	02 UAN PICTURE X VALUE ' '.	
00031	C	02 UANMDT PICTURE X VALUE ' '.	
00032	C	02 UANSEL PICTURE X VALUE ' '.	
00033	C	02 UANMSEL PICTURE X VALUE ' '.	
00034	C	02 UAHSEL PICTURE X VALUE ' '.	
00035	C	02 UAHMSEL PICTURE X VALUE ' '.	
00036	C	02 UAX PICTURE X VALUE ' '.	
00037	C	02 UAXMDT PICTURE X VALUE ' '.	
00038	C	02 UNN PICTURE X VALUE ' '.	
00039	C	02 UNNMDT PICTURE X VALUE ' '.	
00040	C	02 UNNSEL PICTURE X VALUE ' '.	
00041	C	02 UNNMSEL PICTURE X VALUE ' '.	
00042	C	02 UNHSEL PICTURE X VALUE ' '.	
00043	C	02 UNHMSEL PICTURE X VALUE ' '.	
00044	C	02 UNX PICTURE X VALUE ' '.	
00045	C	02 UNXMDT PICTURE X VALUE ' '.	
00046	C	02 PAN PICTURE X VALUE ' '.	
00047	C	02 PANMDT PICTURE X VALUE ' '.	
00048	C	02 PANSEL PICTURE X VALUE ' '.	
00049	C	02 PANMSEL PICTURE X VALUE ' '.	
00050	C	02 PAHSEL PICTURE X VALUE ' '.	
00051	C	02 PAHMSEL PICTURE X VALUE ' '.	
00052	C	02 PAX PICTURE X VALUE ' '.	
00053	C	02 PAXMDT PICTURE X VALUE ' '.	
00054	C	02 PSN PICTURE X VALUE ' '.	

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 1 of 14)

```

00055 C      02  PSNMDT PICTURE X VALUE ' '.
00056 C      02  PSNSEL PICTURE X VALUE ' '.
00057 C      02  PSNMDSSEL PICTURE X VALUE ' '.
00058 C      02  PSMSSEL PICTURE X VALUE ' '.
00059 C      02  PSHMDSSEL PICTURE X VALUE ' '.
00060 C      02  PSX PICTURE X VALUE ' '.
00061 C      02  PSXMDT PICTURE X VALUE ' '.
00062 C      02  SUPR PICTURE X VALUE ' '.
00063 C      02  WRITE1 PICTURE X VALUE ' '.
00064 C      02  ERASWRIT PICTURE X VALUE ' '.
00065 C      02  ERASWRAL PICTURE X VALUE ' '.
00066 C      02  RMDT PICTURE X VALUE ' '.
00067 C      02  RKEYBD PICTURE X VALUE ' '.
00068 C      02  RMDTKEYB PICTURE X VALUE ' '.
00069 C      02  ALARM PICTURE X VALUE ' '.
00070 C      02  ALRMRMDT PICTURE X VALUE ' '.
00071 C      02  ALRMRKEY PICTURE X VALUE ' '.
00072 C      02  ALRMRKY PICTURE X VALUE ' '.
00073 C      02  PRNTNL PICTURE X VALUE ' '.
00074 C      02  PRNT40 PICTURE X VALUE ' '.
00075 C      02  PRNT64 PICTURE X VALUE ' '.
00076 C      02  PRNT80 PICTURE X VALUE ' '.
00077 C      02  PRNLRMDT PICTURE X VALUE ' '.
00078 C      02  PR4ORMDT PICTURE X VALUE ' '.
00079 C      02  PR64ORMDT PICTURE X VALUE ' '.
00080 C      02  PR8ORMDT PICTURE X VALUE ' '.
00081 C      02  PRNLRKEY PICTURE X VALUE ' '.
00082 C      02  PR4ORKEY PICTURE X VALUE ' '.
00083 C      02  PR64RKEY PICTURE X VALUE ' '.
00084 C      02  PR8ORKEY PICTURE X VALUE ' '.
00085 C      02  PRNLRMKY PICTURE X VALUE ' '.
00086 C      02  PR4ORMKY PICTURE X VALUE ' '.
00087 C      02  PR64RMKY PICTURE X VALUE ' '.
00088 C      02  PR8ORMKY PICTURE X VALUE ' '.
00089 C      02  PRNLALRM PICTURE X VALUE ' '.
00090 C      02  PR4OALRP PICTURE X VALUE ' '.
00091 C      02  PR64ALRM PICTURE X VALUE ' '.
00092 C      02  PR8OALRM PICTURE X VALUE ' '.
00093 C      02  PRNLARM PICTURE X VALUE ' '.
00094 C      02  PR4OARM PICTURE X VALUE ' '.
00095 C      02  PR64ARM PICTURE X VALUE ' '.
00096 C      02  PR8OARM PICTURE X VALUE ' '.
00097 C      02  PRNLARKY PICTURE X VALUE ' '.
00098 C      02  PR4OARKY PICTURE X VALUE ' '.
00099 C      02  PR64ARKY PICTURE X VALUE ' '.
00100 C      02  PR8OARKY PICTURE X VALUE ' '.
00101 C      02  PRNLAMKY PICTURE X VALUE ' '.
00102 C      02  PR4OAMKY PICTURE X VALUE ' '.
00103 C      02  PR64AMKY PICTURE X VALUE ' '.
00104 C      02  PR8OAMKY PICTURE X VALUE ' '.
00105 C      *02  NULL PICTURE X VALUE ' '.
00106 C      02  NL PICTURE X VALUE ' '.
00107 C      02  FF PICTURE X VALUE ' '.
00108 C      02  CR PICTURE X VALUE ' '.
00109 C      02  SI PICTURE X VALUE ' '.

```

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 2 of 14)


```

00111 002900 01 COBREENT-CODES COPY ICOMSBS. 00290000
00112 C      01 COBREENT-CODES. 00001000
00113 C      * THESE CODES REPRESENT OFFSETS FOR ROUTINE ADDRESSES IN THE 00002000
00114 C      * TABLE NAMED REENTSBS. ONLY THE MOST COMMONLY USED VALUES 00003000
00115 C      * ARE INCLUDED HERE; THE USERS MANUAL HAS A COMPLETE LIST. 00004000
00116 C      * IF OFFSET ODD, THEN TRUE OFFSET--(OFFSET+1) 00005000
00117 C      05 INTSORTC PIC 99 COMP VALUE 99. 00005300
00118 C      05 DWS-SNAP PIC 99 COMP VALUE 95. 00005400
00119 C      05 MAPFREE PIC 99 COMP VALUE 91. 00005500
00120 C      05 FECMRLSE PIC 99 COMP VALUE 87. 00006000
00121 C      05 FESEND PIC 99 COMP VALUE 83. 00007000
00122 C      05 FESENDC PIC 99 COMP VALUE 79. 00008000
00123 C      05 DYN-ALLOCATE PIC 99 COMP VALUE 75. 00009000
00124 C      05 DYN-ACCESS PIC 99 COMP VALUE 71. 00010000
00125 C      05 MAPURGE PIC 99 COMP VALUE 67. 00011000
00126 C      05 MAPCLR PIC 99 COMP VALUE 63. 00012000
00127 C      05 MAPEND PIC 99 COMP VALUE 59. 00013000
00128 C      05 MAPOUT PIC 99 COMP VALUE 55. 00014000
00129 C      05 MAPIN PIC 99 COMP VALUE 51. 00015000
00130 C      05 INTUNSTO PIC 99 COMP VALUE 47. 00016000
00131 C      05 INTSTORE PIC 99 COMP VALUE 43. 00017000
00132 C      05 INTFETCH PIC 99 COMP VALUE 39. 00018000
00133 C      05 FECMFDDBK PIC 99 COMP VALUE 35. 00019000
00134 C      05 FECMDDQ PIC 99 COMP VALUE 31. 00020000
00135 C      05 DQ-WRITEX PIC 99 COMP VALUE 27. 00021000
00136 C      05 DQ-READX PIC 99 COMP VALUE 23. 00022000
00137 C      05 DQ-WRITE PIC 99 COMP VALUE 19. 00023000
00138 C      05 DQ-READ PIC 99 COMP VALUE 15. 00024000
00139 C      05 DQ-CLOSE PIC 99 COMP VALUE 11. 00025000
00140 C      05 DQ-OPEN PIC 99 COMP VALUE 07. 00026000
00141 C      05 DQ-BUILD PIC 99 COMP VALUE 03. 00027000
00142 C      05 FH-SELECT PIC 99 COMP VALUE 4. 00028000
00143 C      05 FH-RELEASE PIC 99 COMP VALUE 8. 00029000
00144 C      05 FH-READ PIC 99 COMP VALUE 12. 00030000
00145 C      05 FH-WRITE PIC 99 COMP VALUE 16. 00031000
00146 C      05 FH-GET PIC 99 COMP VALUE 20. 00032000
00147 C      05 FH-PUT PIC 99 COMP VALUE 24. 00033000
00148 C      05 FH-RELEX PIC 99 COMP VALUE 28. 00034000
00149 C      05 FH-FEQV PIC 99 COMP VALUE 32. 00035000
00150 C      05 COBPUT PIC 99 COMP VALUE 68. 00036000
00151 C      05 MSGCOL PIC 99 COMP VALUE 72. 00037000
00152 C      05 COBSTORF PIC 99 COMP VALUE 76. 00038000
00153 C      05 CONVERSE PIC 99 COMP VALUE 80. 00039000
00154 C      05 DBINT PIC 99 COMP VALUE 84. 00040000
00155 C      05 LOGPUT PIC 99 COMP VALUE 88. 00041000
00156 C      05 PAGE-FILE PIC 99 COMP VALUE 92. 00042000
00157 C      05 FH-GETV PIC 99 COMP VALUE 96. 00043000
00158 C      05 FH-PUTV PIC 999 COMP VALUE 100. 00044000
00159 C      * CODES 104 AND UP INDICATE USER ADDITIONS TO THE TABLE 00045000

00161 003000 05 SQCOBOLB PIC 999 COMP VALUE 104. 00300000
00162 003100 01 FILLER PIC X(22) VALUE 00310000
00163 003200 'END OF WORKING STORAGE'. 00320000

```

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 3 of 14)

```

00165 003300 LINKAGE SECTION.                                00330000
00166 003400 01 INPUT-MESSAGE COPY ICOMINMG.              00340000
00167 C          01 INPUT-MESSAGE.                          00000100
00168 C          04 MESSG-HDR.                                00000200
00169 C              06 MSGH-LENGTH                          PIC S9999  COMP.  00000300
00170 C              06 MSGH-QPR                             PIC X.           00000400
00171 C              06 MSGH-RSCH                            PIC X.           00000500
00172 C              06 MSGH-RSC                            PIC X.           00000600
00173 C              06 MSGH-SSC                            PIC X.           00000700
00174 C              06 MSGH-MMN                            PIC XXX.         00000800
00175 C              06 MSGH-DATE.                            00000900
00176 C                  08 MSGH-YR                          PIC 99.          00001000
00177 C                  08 MSGH-PERIOD                      PIC X.           00001100
00178 C                  08 MSGH-JULIAN-DAY                  PIC 999.         00001200
00179 C              06 MSGH-TIME.                            00001300
00180 C                  08 MSGH-MM                          PIC 99.          00001400
00181 C                  08 MSGH-MM                          PIC 99.          00001500
00182 C                  08 MSGH-SS                          PIC 99.          00001600
00183 C                  08 MSGH-TM                          PIC 99.          00001700
00184 C              06 MSGH-TID.                            00001800
00185 C                  08 MSGH-TI1                          PIC X.           00001900
00186 C                  08 MSGH-TI2-3                       PIC XX.          00002000
00187 C                  08 MSGH-TI4-5                       PIC 99.          00002100
00188 C              06 MSGH-CON                             PIC S9999  COMP.  00002200
00189 C              06 MSGH-FLGS                            PIC X(2).        00002300
00190 C              06 MSGH-BMN                             PIC X(3).        00002350
00191 C              06 MSGH-SSCH                           PIC X.           00002400
00192 C              06 MSGH-USR                             PIC X.           00002500
00193 C              06 MSGH-ADDR                            PIC XX.          00002600
00194 C              06 MSGH-LOG                             PIC X.           00002700
00195 C              06 MSGH-BLK                             PIC X.           00002750
00196 C              06 MSGH-VMI                             PIC X.           00002800

00198 003500 02 INPUT-TEXT.                                00350000
00199 003600 04 INPUT-VERB PIC X(4).                       00360000
00200 003700 01 ICDM-SPA PIC X(500).                       00370000
00201 003800 01 ICDM-SCT PIC X(100).                       00380000
00202 003900 01 ICDM-RETURN PIC S9(7) COMP.                00390000
00203 004000 01 DYNAMIC-WORK-SPACE COPY ICOMDWS.           00400000
00204 C          01 DYNAMIC-WORK-SPACE.                       00000100
00205 C          02 OUTPUT-MESSAGE.                            00000200
00206 C          04 OMESSG-HDR.                                00000300
00207 C              06 OMSGH-LENGTH                          PIC S9999  COMP.  00000400
00208 C              06 OMSGH-QPR                             PIC X.           00000500
00209 C              06 OMSGH-RSCH                            PIC X.           00000600
00210 C              06 OMSGH-RSC                            PIC X.           00000700
00211 C              06 OMSGH-SSC                            PIC X.           00000800
00212 C              06 OMSGH-MMN                            PIC XXX.         00000900
00213 C              06 OMSGH-DATE.                            00001000
00214 C                  08 OMSGH-YR                          PIC 99.          00001100
00215 C                  08 OMSGH-PERIOD                      PIC X.           00001200
00216 C                  08 OMSGH-JULIAN-DAY                  PIC 999.         00001300
00217 C              06 OMSGH-TIME.                            00001400
00218 C                  08 OMSGH-MM                          PIC 99.          00001500
00219 C                  08 OMSGH-MM                          PIC 99.          00001600
00220 C                  08 OMSGH-SS                          PIC 99.          00001700
00221 C                  08 OMSGH-TM                          PIC 99.          00001800

```

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 4 of 14)

00222 C		06	MSGH-TID.		00001900
00223 C		08	MSGH-T11	PIC X.	00002000
00224 C		08	MSGH-T12-3	PIC XX.	00002100
00225 C		08	MSGH-T14-5	PIC 99.	00002200
00226 C		06	MSGH-COM	PIC 59999 COMP.	00002300
00227 C		06	MSGH-FLGS	PIC X(2).	00002400
00228 C		06	MSGH-BMN	PIC X(3).	00002450
00229 C		06	MSGH-SSCH	PIC X.	00002500
00230 C		06	MSGH-USR	PIC X.	00002600
00231 C		06	MSGH-ADDR	PIC XX.	00002700
00232 C		06	MSGH-LOG	PIC X.	00002750
00233 C		06	MSGH-BLK	PIC X.	00002800
00234 C		06	MSGH-VMI	PIC X.	00002900
00236	004100	02	SYMBOLIC-MAP.		00410000
00237	004200	03	COPY STKSTATS		00420000
00238	004300	03	MAP1.		00430000
00239	004400	05	VERBF.		00440000
00240	004500	06	VERBL	PIC 9(4) COMP.	00450000
00241	004600	06	VERBT	PIC X.	00460000
00242	004700	06	VERB	PIC X(4).	00470000
00243	004800	04	PARTNOF.		00480000
00244	004900	05	PARTNOL	PIC 9(4) COMP.	00490000
00245	005000	05	PARTNOT	PIC X.	00500000
00246	005100	05	PARTNC.		00510000
00247	005200	06	FILLER	PIC 59(4).	00520000
00248	005300	06	RBNBYTE	PIC 59.	00530000
00249	005400	04	USEG1.		00540000
00250	005500	05	WHSNOF.		00550000
00251	005600	06	WHSNOL	PIC 9(4) COMP.	00560000
00252	005700	06	WHSNOT	PIC X.	00570000
00253	005800	06	WHSNO	PIC 5999.	00580000
00254	005900	05	PRTDATAF.		00590000
00255	006000	06	PRTDATAL	PIC 9(4) COMP.	00600000
00256	006100	06	PRTDATAT	PIC X.	00610000
00257	006200	06	PRTDATA	PIC X(54).	00620000
00258	006300	05	ORDUNTF.		00630000
00259	006400	06	ORDUNTL	PIC 9(4) COMP.	00640000
00260	006500	06	ORDUNTT	PIC X.	00650000
00261	006600	06	ORDUNT	PIC X(5).	00660000
00262	006700	05	PRTPRCF.		00670000
00263	006800	06	PRTPRCL	PIC 9(4) COMP.	00680000
00264	006900	06	PRTPRCT	PIC X.	00690000
00265	007000	06	PRTPRC	PIC 5999V9(4) COMP-3.	00700000
00266	007100	05	WHSLOCF.		00710000
00267	007200	06	WHSLOCL	PIC 9(4) COMP.	00720000
00268	007300	06	WHSLOCT	PIC X.	00730000
00269	007400	06	WHSLOC	PIC X(23).	00740000
00270	007500	05	STKLEVF.		00750000
00271	007600	06	STKLEVL	PIC 9(4) COMP.	00760000
00272	007700	06	STKLEVY	PIC X.	00770000
00273	007800	06	STKLEV	PIC 59(7) COMP-3.	00780000
00274	007900	05	LEVDATEF.		00790000
00275	008000	06	LEVDATEL	PIC 9(4) COMP.	00800000
00276	008100	06	LEVDATEY	PIC X.	00810000
00277	008200	06	LEVDATE	PIC X(8).	00820000
00278	008300	05	STKCRDF.		00830000

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 5 of 14)

```

00279 008400          06 STKORDL PIC 9(4) COMP.          00840000
00280 008500          06 STKORDT PIC X.                00850000
00281 008600          06 STKORD PIC 59(7) COMP-3.       00860000
00282 008700          05 ORDCATEF.                    00870000
00283 008800          06 ORDDATEL PIC 9(4) COMP.       00880000
CC284 008900          06 ORDDATET PIC X.              00890000
00285 009000          06 ORDDATE PIC X(8).            00900000
00286 009100          04 FILLER PIC X(7).             00910000
00287 009200 03 ERRMAP.                                00920000
00288 009300          05 ERRMSGF.                    00930000
00289 009400          06 ERRMSG L PIC 9(4) COMP.       00940000
00290 009500          06 ERRMSGT PIC X.              00950000
00291 009600          06 ERRMSG PIC X(50).            00960000
00292 009700          04 FILLER PIC X(7).             00970000
00293 009800 02 RECORD-AREA.                          00980000
00294 009900          04 PART-RECORD.                00990000
00295 010000*          NOTE 100 CHARACTER BDAM RECORD WITHOUT KEYS. 01000000
00296 010100          06 P-REC-PART-DATA.            01010000
00297 010200          08 P-REC-PIN PIC X(5).          01020000
00298 010300          08 P-REC-DES PIC X(54).         01030000
00299 010400          08 P-REC-UNT PIC X(5).          01040000
00300 010500          06 P-REC-PRC PIC 99V9(4) COMP-3. 01050000
00301 010600          06 P-REC-MFR-NUM PIC X(15).     01060000
00302 010700          06 FILLER PIC X(17).           01070000
00303 010800          04 STOCK-RECORD.              01080000
00304 010900*          NOTE 80 CHARACTER VSAM RECORD. 01090000
00305 011000          06 DELETE-CHARACTER PIC X.     01100000
00306 011100          06 S-REC-KEY-FIELD.            01110000
00307 011200          08 S-REC-WHS PIC 9(3).          01120000
00308 011300          08 S-REC-PNO PIC 9(5).          01130000
00309 011400          06 FILLER PIC X(28).           01140000
00310 011500          06 S-REC-STOCK-DATA.           01150000
00311 011600          08 S-REC-WLC PIC X(23).         01160000
00312 011700          08 S-REC-LEV PIC 9(7) COMP-3.   01170000
00313 011800*          NOTE S-REC-LEV IS 4 CHARACTERS LONG. 01180000
00314 011900          08 S-REC-LDT PIC X(6).         01190000
00315 012000          08 S-REC-ORD PIC 9(7) COMP-3.   01200000
00316 012100*          NOTE S-REC-ORD IS 4 CHARACTERS LONG. 01210000
00317 012200          08 S-REC-ODT PIC X(6).         01220000

```

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 6 of 14)

00319	012300	02	STATWD	PIC S9(7)	COMP SYNC.	01230000
00320	012400*		NOTE THIS PUTS US ONTO A FULLWORD BOUNDARY ALIGNMENT.			01240000
00321	012500	02	FH-STATUS	REDEFINES	STATWD.	01250000
00322	012600	04	FH-STAT1	PIC X.		01260000
00323	012700		88	IOK	VALUE 0.	01270000
00324	012800		88	IGERROR	VALUE 1.	01280000
00325	012900		88	NOT-FOUND	VALUE 2.	01290000
00326	013000		88	NO-DD	VALUE 9.	01300000
00327	013100	04	F-M-STAT2	PIC X.		01310000
00328	013200	04	FILLER	PIC X(2).		01320000
00329	013300	02	EXTDSC1	PIC X(48).		01330000
00330	013400*		NOTE WE ARE STILL ALIGNED HERE.			01340000
00331	013500	02	RBN-WORD	PIC S9(7)	COMP.	01350000
00332	013600	02	RBN-FILLER	REDEFINES	RBN-WOFD.	01360000
00333	013700	04	FILLER	PIC X.		01370000
00334	013800	04	RBN	PIC X(3).		01380000
00335	013900	02	CURRENT-FILE	PIC X(8).		01390000
00336	014000	02	MCW	PIC 9(8)	COMP SYNC.	01400000
00337	014100	02	MCW-CODE-BYTES	REDEFINES	MCW.	01410000
00338	014200	04	MCW-RETURN-CODE	PIC X.		01420000
00339	014300		88	MAPPING-OK	VALUE ZERO.	01430000
00340	014400		88	MAPEND-SUCCESSFUL	VALUE '8'.	01440000
00341	014500	04	MCW-OPTION-2	PIC X.		01450000
00342	014600	04	MCW-OPTION-3	PIC X.		01460000
00343	014700	04	MCW-OPTION-4	PIC X.		01470000
00344	014800	02	MCW-CODES-PART	REDEFINES	MCW.	01480000
00345	014900	04	MCW-CODES1-2	PIC X(2).		01490000
00346	015000	04	FILLER	PIC X(2).		01500000
00347	015100	02	MCB	PIC X(48).		01510000
00348	015200	02	KEY-FIELD	PIC 9(8).		01520000
00349	015300	02	DATE-EDIT.			01530000
00350	015400	04	D-E-MO	PIC X(2).		01540000
00351	015500	04	D-E-DAY	PIC X(2).		01550000
00352	015600	04	D-E-YEAR	PIC X(2).		01560000
00353	015700	02	DATE-MOVE.			01570000
00354	015800	04	D-M-MO	PIC X(2).		01580000
00355	015900	04	SLASH2	PIC X.		01590000
00356	016000	04	D-M-DAY	PIC X(2).		01600000
00357	016100	04	SLASH1	PIC X.		01610000
00358	016200	04	D-M-YEAR	PIC X(2).		01620000

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 7 of 14)

00360	016300	02	INVALID-INPUT-MESSAGE.		01630000
00361	016400	04	MSG-7	PIC X(50).	01640000
00362	016500	02	NO-PART-MESSAGE REDEFINES	INVALID-INPUT-MESSAGE.	01650000
00363	016600	04	MSG-1	PIC X(12).	01660000
00364	016700	04	NOPART-PNO	PIC X(5).	01670000
00365	016800	04	MSG-2	PIC X(11).	01680000
00366	016900	02	NOWARES-MESSAGE REDEFINES	INVALID-INPUT-MESSAGE.	01690000
00367	017000	04	MSG-3	PIC X(5).	01700000
00368	017100	04	NOWARES-PNO	PIC X(5).	01710000
00369	017200	04	MSG-4	PIC X(24).	01720000
00370	017300	04	NOWARES-WMS	PIC X(3).	01730000
00371	017400	02	CANCEL-MESSAGE REDEFINES	INVALID-INPUT-MESSAGE.	01740000
00372	017500	04	CAN-CODE	PIC X(15) JUST RIGHT.	01750000
00373	017600	04	CAN-FILE-NAME	PIC X(8).	01760000
00374	017700	04	MSG-5	PIC X(20).	01770000
00375	017800	02	MAPPING-ERR-MESSAGE REDEFINES	INVALID-INPUT-MESSAGE.	01780000
00376	017900	04	MSG-6	PIC X(17).	01790000
00377	018000	04	ERROR-TAG	PIC X(4).	01800000
00378	018100	02	MAP-FLAG	PIC X.	01810000
00379	018200	88	MAP-GOOD	VALUE 'G'.	01820000
00380	018300	88	MAP-ERR	VALUE 'E'.	01830000
00381	018400	88	MAP-OUT-ABORT	VALUE 'A'.	01840000
00382	018500	02	FH-READ-FLAG	PIC X.	01850000
00383	018600	88	BDAM-READ-OK	VALUE 'D'.	01860000
00384	018700	88	VSAM-READ-OK	VALUE 'V'.	01870000

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 8 of 14)

00386	018900	PROCEDURE DIVISION USING INPUT-MESSAGE	018900C0
00387	019000	ICOM-SPA	01900000
00388	019100	ICOM-SCT	01910000
00389	019200	ICOM-RETURN	01920000
00390	019300	DYNAMIC-WORK-SPACE.	019300C0
00392	019500	0100-MAIN-LINE.	01950000
00393	019600	PERFORM 1000-HOUSEKEEPING.	01960000
00394	019700	PERFORM 2000-HEADER-MOVE.	01970000
00395	019800	PERFORM 3000-MAP-IN.	01980000
00396	019900	MOVE LOW-VALUES TO VERB.	01990000
00397	020000	IF PARTNOT NOT EQUAL TO LOW-VALUES	02000000
00398	020100	OR WMSNOT NOT EQUAL TO LOW-VALUES	02010000
00399	020200	PERFORM 8900-INVALID-INPUT-RTN	02020000
00400	020300	ELSE	02030000
00401	020400	IF NOT MAPPING-OK	020400C0
00402	020500	PERFORM 8850-MAPPING-ERR-RTN	02050000
00403	020600	ELSE	02060000
00404	020700	PERFORM 3500-MAP-CLEAR-RTN	02070000
00405	020800	PERFORM 4000-READ-PART-FILE	02080000
00406	020900	PERFORM 5000-FH-BDAM-READ	02090000
00407	021000	IF BDAM-READ-OK	02100000
00408	021100	PERFORM 6000-READ-STOCK-FILE	02110000
00409	021200	PERFORM 7000-FH-VSAM-READ	021200C0
00410	021300	IF VSAM-READ-OK	02130000
00411	021400	PERFORM 8000-MAP-OUT	02140000
00412	021500	IF NOT MAPPING-CK	02150000
00413	021600	PERFORM 8850-MAPPING-ERR-RTN.	02160000
00414	021700	IF MAP-GOOD	02170000
00415	021800	PERFORM 8500-GOOD-MAP-END	02180000
00416	021900	ELSE	02190000
00417	022000	IF MAP-ERR	022000C0
00418	022100	PERFORM 8600-ERR-MAP-END.	02210000
00419	022200	GOBACK.	02220000
00421	022400	1000-HOUSEKEEPING.	02240000
00422	022500	MOVE +0 TC ICOM-RETURN.	02250000
00423	022600	MOVE 'G' TO MAP-FLAG.	02260000
00425	022800	2000-HEADER-MOVE.	02280000
00426	022900	MOVE MESSG-HDR TO OMESSG-HDR.	022900C0
00428	023100	3000-MAP-IN.	02310000
00429	023200	MOVE SPACES TO MCH-CODE-BYTES.	02320000
00430	023300	CALL 'COBREENT' USING MAPIN	023300C0
00431	023400	MCB	02340000
00432	023500	ID-GRUP-NAME	02350000
00433	023600	IC-MAP-NAME	02360000
00434	023700	INPUT-MESSAGE	023700C0
00435	023800	MCH	02380000
00436	023900	MAP1.	02390000

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 9 of 14)

```

00438 024100 3500-MAP-CLEAR-RTN.                                02410000
00439 024200     MOVE SPACES TO MCW-CODE-BYTES.                02420000
00440 024300     MOVE 'A' TO MCW-OPTION-4.                    02430000
00441 024400     CALL 'COBREENT' USING MAPCLR                  02440000
00442 024500         MCW                                        02450000
00443 024600         ID-GROUP-NAME                            02460000
00444 024700         ID-MAP-NAME                              02470000
00445 024800         MAP1                                      02480000
00446 024900         OMSGH-TID.                                02490000

00448 025100 4000-READ-PART-FILE.                                02510000
00449 025200     MOVE RBNBYTE TO RBN-WORD.                    02520000
00450 025300     MOVE DD-PART TO CURRENT-FILE.                02530000

00452 025500 5000-FH-BDAM-READ.                                02550000
00453 025600     CALL 'COBREENT' USING SOCOBOLB                02560000
00454 025700         PART-RECORD                              02570000
00455 025800         RBN.                                      02580000
00456 025900     IF ICOM-RETURN EQUAL 1                        02590000
00457 026000         PERFORM 9600-ID-ERROR-ROUTINE            02600000
00458 026100     ELSE                                          02610000
00459 026200         IF ICOM-RETURN EQUAL 2                    02620000
00460 026300             PERFORM 9700-NOT-FOUND-RTN            02630000
00461 026400     ELSE                                          02640000
00462 026500         IF ICOM-RETURN EQUAL 9                    02650000
00463 026600             PERFORM 9500-AQ-DD-ROUTINE            02660000
00464 026700     ELSE                                          02670000
00465 026800         IF P-REC-PIN NOT EQUAL PARTNO            02680000
00466 026900             PERFORM 9700-NOT-FOUND-RTN            02690000
00467 027000     ELSE                                          02700000
00468 027100         MOVE P-REC-DES TO PRDDBA                  02710000
00469 027200         MOVE P-REC-UNT TO ORDUNT                  02720000
00470 027300         MOVE P-REC-PRC TO PRTPRC                  02730000
00471 027400         MOVE BDAM-READ-GOOD TO FH-READ-FLAG.     02740000

00473 027600 6000-READ-STOCK-FILE.                                02760000
00474 027700     MOVE DD-STOCK TO CURRENT-FILE.                02770000
00475 027800     MOVE WMSND TO S-REC-WMS.                      02780000
00476 027900     MOVE PARTNO TO S-REC-PNO.                     02790000
00477 028000     MOVE S-REC-KEY-FIELD TO KEY-FIELD.            02800000

```

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 10 of 14)


```

00475 002270 7000-FH-VSAM-READ. 00028300
00476 002280 MOVE ZERO TO FH-READ-FLAG. 00028400
00477 002290 MOVE LOW-VALUES TO EXTDSCT. 00028500
00478 002300 PERFORM 7400-FH-SELECT-ROUTINE. 00028600
00479 002310 IF NO-DD 00028700
00480 002320 PERFORM 9500-NO-DD-ROUTINE 00028800
00481 002330 ELSE 00028900
00482 002340 PERFORM 7200-FH-VSAM-READ-CONTINUE 00029000
00483 002350 IF IOERROR 00029100
00484 002360 PERFORM 9600-IO-ERROR-ROUTINE 00029200
00485 002370 ELSE 00029300
00486 002380 IF NOT-FOUND 00029400
00487 002390 MOVE MSG-C TO MSG-3 00029500
00488 002400 MOVE MSG-D TO MSG-4 00029600
00489 002410 MOVE PARTNO TO NOWARES-PNO 00029700
00490 002420 MOVE WMSNO TO NOWARES-WMS 00029800
00491 002430 MOVE NOWARES-MESSAGE TO ERRMSG 00029900
00492 002440 PERFORM 9800-SEND-ERRCR-MESSAGE 00030000
00493 002450 ELSE 00030100
00494 002460 MOVE VSAM-READ-GOOD TO FH-READ-FLAG 00030200
00495 002470 MOVE S-REC-WLC TO WMSLOC 00030300
00496 002480 MOVE S-REC-LEV TO STKLEY 00030400
00497 002490 MOVE S-REC-LDT TO DATE-EDIT 00030500
00498 002500 PERFORM 7300-DATE-EDITING 00030600
00499 002510 MOVE DATE-MOVE TO LEVDATE 00030700
00500 002520 MOVE S-REC-ORD TO STKORD 00030800
00501 002530 MOVE S-REC-GDT TO DATE-EDIT 00030900
00502 002540 PERFORM 7300-DATE-EDITING 00031000
00503 002550 MOVE DATE-MOVE TO ORDDATE. 00031100
00504 002560 PERFORM 7500-FH-RELEASE-ROUTINE. 00031200

00506 002580 7200-FH-VSAM-READ-CONTINUE. 00031400
00507 002590 MOVE SPACES TO FH-STATUS. 00031500
00508 002600 CALL 'COBREENT' USING FH-GETV 00031600
00509 002610 EXTDSCT 00031700
00510 002620 FH-STATUS 00031800
00511 002630 STOCK-RECORD 00031900
00512 002640 KEY-FIELD. 00032000

```

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 11 of 14)

00529	033200	7300-DATE-EDITING.	03320000
00530	033300	MOVE D-E-YEAR TO D-M-YEAR.	03330000
00531	033400	MOVE SLASH TO SLASH1.	03340000
00532	033500	MOVE D-E-DAY TO D-M-DAY.	03350000
00533	033600	MOVE SLASH TO SLASH2.	03360000
00534	033700	MOVE D-E-MO TO D-M-MO.	03370000
00536	033900	7400-FH-SELECT-ROUTINE.	03390000
00537	034000	MOVE SPACES TO FH-STATUS.	03400000
00538	034100	CALL 'COBREENT' USING FH-SELECT	03410000
00539	034200	EXTDST	03420000
00540	034300	FH-STATUS	03430000
00541	034400	CURRENT-FILE.	03440000
00543	034600	7500-FH-RELEASE-ROUTINE.	03460000
00544	034700	MOVE SPACES TO FH-STATUS.	03470000
00545	034800	CALL 'COBREENT' USING FH-RELEASE	03480000
00546	034900	EXTDST	03490000
00547	035000	FH-STATUS.	03500000
00549	035200	8000-MAP-OUT.	03520000
00550	035300	MOVE SPACES TO MCW-CODE-BYTES.	03530000
00551	035400	CALL 'COBREENT' USING MAPOUT	03540000
00552	035500	MCB	03550000
00553	035600	ID-GRUP-NAME	03560000
00554	035700	ID-MAP-NAME	03570000
00555	035800	MAP1	03580000
00556	035900	MCW	03590000
00557	036000	DMSGH-TID.	03600000
00559	036200	8500-GOOD-MAP-END.	03620000
00560	036300	MOVE ' Q ' TO MCW-CODE-BYTES.	03630000
00561	036400	PERFORM 8700-CALL-MAP-END.	03640000
00562	036500	IF NOT MAPEND-SUCCESSFUL	03650000
00563	036600	PERFORM 8800-MAP-PURGE-RTN	03660000
00564	036700	PERFORM 8850-MAPPING-ERR-RTN	03670000
00565	036800	PERFORM 8600-ERR-MAP-END.	03680000
00567	037000	8600-ERR-MAP-END.	03700000
00568	037100	MOVE ' Q ' TO MCW-CODE-BYTES.	03710000
00569	037200	MOVE WRITE1 TO MCW-OPTION-3.	03720000
00570	037300	PERFORM 8700-CALL-MAP-END.	03730000
00571	037400	IF NOT MAPEND-SUCCESSFUL	03740000
00572	037500	PERFORM 8800-MAP-PURGE-RTN	03750000
00573	037600	MOVE +8 TO ICOM-RETURN.	03760000
00575	037800	8700-CALL-MAP-END.	03780000
00576	037900	CALL 'COBREENT' USING MAPEND	03790000
00577	038000	MCB	03800000
00578	038100	OUTPUT-MESSAGE	03810000
00579	038200	MCW.	03820000
00581	038400	8800-MAP-PURGE-RTN.	03840000
00582	038500	CALL 'COBREENT' USING MAPPURGE	03850000
00583	038600	MCB.	03860000

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 12 of 14)

00585	038800	8850-MAPPING-ERR-RTN.	03880000
00586	038900	MOVE MSG-F TO MSG-6.	03890000
00587	039000	MOVE MCM-CODES1-2 TO ERROR-TAG.	03900000
00588	039100	MOVE MAPPING-ERR-MESSAGE TO ERRMSG.	03910000
00589	039200	PERFORM 9800-SEND-ERROR-MESSAGE.	03920000
00591	039400	8900-INVALID-INPUT-RTN.	03940000
00592	039500	IF PARTNOT NOT EQUAL TO LOW-VALUES	03950000
00593	039600	IF WMSNOT NOT EQUAL TO LOW-VALUES	03960000
00594	039700	MOVE MSG-I TO MSG-7	03970000
00595	039800	ELSE	03980000
00596	039900	MOVE MSG-G TO MSG-7	03990000
00597	040000	ELSE	04000000
00598	040100	IF WMSNCT NOT EQUAL TO LOW-VALUES	04010000
00599	040200	MOVE MSG-H TO MSG-7.	04020000
00600	040300	MOVE INVALID-INPUT-MESSAGE TO ERRMSG.	04030000
00601	040400	PERFORM 9800-SEND-ERROR-MESSAGE.	04040000
00603	040600	9500-NO-DD-ROUTINE.	04060000
00604	040700	MOVE MSG-E TO MSG-5.	04070000
00605	040800	MOVE 'NO DD FOR FILE ' TO CAN-CODE.	04080000
00606	040900	MOVE CURRENT-FILE TO CAN-FILE-NAME.	04090000
00607	041000	MOVE CANCEL-MESSAGE TO ERRMSG.	04100000
00608	041100	MOVE +0 TO ICOM-RETURN.	04110000
00609	041200	PERFORM 9800-SEND-ERROR-MESSAGE.	04120000
00611	041400	9600-ID-ERROR-ROUTINE.	04140000
00612	041500	MOVE MSG-E TO MSG-5.	04150000
00613	041600	MOVE 'ID ERROR ON ' TO CAN-CODE.	04160000
00614	041700	MOVE CURRENT-FILE TO CAN-FILE-NAME.	04170000
00615	041800	MOVE CANCEL-MESSAGE TO ERRMSG.	04180000
00616	041900	MOVE +0 TO ICOM-RETURN.	04190000
00617	042000	PERFORM 9800-SEND-ERROR-MESSAGE.	04200000
00619	042200	9700-NOT-FOUND-RTN.	04220000
00620	042300	MOVE +0 TO ICOM-RETURN.	04230000
00621	042400	MOVE MSG-A TO MSG-1.	04240000
00622	042500	MOVE MSG-B TO MSG-2.	04250000
00623	042600	MOVE PARTNO TO NOPART-PNO.	04260000
00624	042700	MOVE NO-PART-MESSAGE TO ERRMSG.	04270000
00625	042800	PERFORM 9800-SEND-ERROR-MESSAGE.	04280000
00627	043000	9800-SEND-ERROR-MESSAGE.	04300000
00628	043100	MVCE SPACES TO MCM-CODE-BYTES.	04310000
00629	043200	MVCE 'E' TO MAP-FLAG.	04320000
00630	043300	CALL 'COBREENT' USING MAPOUT	04330000
00631	043400	MCB	04340000
00632	043500	IO-GRUP-NAME	04350000
00633	043600	ERR-MAP-NAME	04360000
00634	043700	ERRMAP	04370000
00635	043800	MCM	04380000
00636	043900	OMSGH-TID.	04390000
00637	044000	IF NOT MAPPING-OK	04400000
00638	044100	MOVE +8 TO ICOM-RETURN	04410000
00639	044200	MOVE 'A' TO MAP-FLAG.	04420000

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 13 of 14)

DNM-8-194	03	MSGH-ADDR	BLL-3	025	DNM-8-194	DS 2C	DISP
DNM-8-213	03	MSGH-LOG	BLL-3	027	DNM-8-213	DS 1C	DISP
DNM-8-234	03	MSGH-BLK	BLL-3	028	DNM-8-234	DS 1C	DISP
DNM-8-252	03	MSGH-VMI	BLL-3	029	DNM-8-252	DS 1C	DISP
DNM-8-273	02	INPUT-TEXT	BLL-3	02A	DNM-8-273	DS OCL4	GROUP
DNM-8-299	03	INPUT-VERB	BLL-3	02A	DNM-8-299	DS 4C	DISP
DNM-8-319	01	ICOM-SPA	BLL-4	000	DNM-8-319	DS 500C	DISP
DNM-8-337	01	ICOM-SCT	BLL-5	000	DNM-8-337	DS 100C	DISP
DNM-8-355	01	ICOM-RETURN	BLL-6	000	DNM-8-355	DS 4C	COMP
DNM-8-376	01	DYNAMIC-WORK-SPACE	BLL-7	000	DNM-8-376	DS OCL634	GROUP
DNM-8-407	02	OUTPUT-MESSAGE	BLL-7	000	DNM-8-407	DS OCL42	GROUP
DNM-8-434	03	OMESSG-MDR	BLL-7	000	DNM-8-434	DS OCL42	GROUP
DNM-8-457	04	OPSGH-LENGTH	BLL-7	000	DNM-8-457	DS 2C	COMP
DNM-8-479	04	OPSGH-QPR	BLL-7	002	DNM-8-479	DS 1C	DISP
DNM-9-000	04	OMSGH-RSCH	BLL-7	003	DNM-9-000	DS 1C	DISP
DNM-9-020	04	OMSGH-RSC	BLL-7	004	DNM-9-020	DS 1C	DISP
DNM-9-042	04	OMSGH-SSC	BLL-7	005	DNM-9-042	DS 1C	DISP
DNM-9-061	04	OMSGH-MMN	BLL-7	006	DNM-9-061	DS 3C	DISP
DNM-9-083	04	OPSGH-DATE	BLL-7	009	DNM-9-083	DS OCL6	GROUP
DNM-9-109	05	OMSGH-YR	BLL-7	009	DNM-9-109	DS 2C	DISP-NM
DNM-9-127	05	OPSGH-PERIOD	BLL-7	008	DNM-9-127	DS 1C	DISP
DNM-9-152	05	OMSGH-JULIAN-DAY	BLL-7	00C	DNM-9-152	DS 3C	DISP-NM
DNM-9-178	04	OMSGH-TIME	BLL-7	00F	DNM-9-178	DS OCL8	GROUP
DNM-9-201	05	OPSGH-MH	BLL-7	00F	DNM-9-201	DS 2C	DISP-NM
DNM-9-219	05	OPSGH-MM	BLL-7	011	DNM-9-219	DS 2C	DISP-NM
DNM-9-237	05	OMSGH-SS	BLL-7	013	DNM-9-237	DS 2C	DISP-NM
DNM-9-255	05	OMSGH-TH	BLL-7	015	DNM-9-255	DS 2C	DISP-NM
DNM-9-273	04	OPSGH-TID	BLL-7	017	DNM-9-273	DS OCL5	GROUP
DNM-9-295	05	OMSGH-T11	BLL-7	017	DNM-9-295	DS 1C	DISP
DNM-9-314	05	OPSGH-T12-3	BLL-7	018	DNM-9-314	DS 2C	DISP
DNM-9-335	05	OMSGH-T14-5	BLL-7	01A	DNM-9-335	DS 2C	DISP-NM
DNM-9-359	04	OMSGH-CGN	BLL-7	01C	DNM-9-359	DS 2C	COMP
DNM-9-381	04	OPSGH-FLGS	BLL-7	01E	DNM-9-381	DS 2C	DISP
DNM-9-401	04	OPSGH-BNN	BLL-7	020	DNM-9-401	DS 3C	DISP
DNM-9-420	04	OMSGH-SSCH	BLL-7	023	DNM-9-420	DS 1C	DISP
DNM-9-443	04	OMSGH-USR	BLL-7	024	DNM-9-443	DS 1C	DISP
DNM-9-465	04	OMSGH-ADDR	BLL-7	025	DNM-9-465	DS 2C	DISP
DNM-9-488	04	OPSGH-LOG	BLL-7	027	DNM-9-488	DS 1C	DISP
DNM-10-000	04	OPSGH-BLK	BLL-7	028	DNM-10-000	DS 1C	DISP
DNM-10-022	04	OPSGH-VMI	BLL-7	029	DNM-10-022	DS 1C	DISP
DNM-10-041	02	SYMBOLIC-MAP	BLL-7	02A	DNM-10-041	DS OCL222	GROUP
DNM-10-069	03	MAP1	BLL-7	02A	DNM-10-069	DS OCL162	GROUP
DNM-10-086	04	VERBF	BLL-7	02A	DNM-10-086	DS OCL7	GROUP
DNM-10-104	05	VERBL	BLL-7	02A	DNM-10-104	DS 2C	COMP
DNM-10-122	05	VERBT	BLL-7	02C	DNM-10-122	DS 1C	DISP
DNM-10-137	05	VERB	BLL-7	02D	DNM-10-137	DS 4C	DISP
DNM-10-151	04	PARTNCF	BLL-7	031	DNM-10-151	DS OCL8	GROUP
DNM-10-174	05	PARTNOL	BLL-7	031	DNM-10-174	DS 2C	COMP
DNM-10-194	05	PARTNOD	BLL-7	033	DNM-10-194	DS 1C	DISP
DNM-10-214	05	PARTND	BLL-7	034	DNM-10-214	DS OCL5	GROUP
DNM-10-233	06	FILLER	BLL-7	034	DNM-10-233	DS 4C	DISP-NM
DNM-10-247	06	RBNBYTE	BLL-7	038	DNM-10-247	DS 1C	DISP-NM
DNM-10-264	04	USEG1	BLL-7	039	DNM-10-264	DS OCL140	GROUP
DNM-10-285	05	W-SMDF	BLL-7	039	DNM-10-285	DS OCL6	GROUP
DNM-10-304	06	W-SMOL	BLL-7	039	DNM-10-304	DS 2C	COMP
DNM-10-320	06	WMSACT	BLL-7	038	DNM-10-320	DS 1C	DISP
DNM-10-336	06	WMSNC	BLL-7	03C	DNM-10-336	DS 3C	DISP-NM

Figure 48. Sample Reentrant Subsystem (IBM ANS COBOL) (Page 14 of 14)

```

PP 5740-CB1 RELEASE 2.4                                18P CS/V5 COBOL

1

000C1 000010 IDENTIFICATION DIVISION.
000C2 000020 PROGRAM-ID. SQCOBGLB
000C3 000030 ENVIRONMENT DIVISION.
000C4 000040 DATA DIVISION.
000C5 000050 WORKING-STORAGE SECTION.
000C6 000060 77 DD-PART                                PIC X(8)      VALUE 'PARTFILE'.
000C7 000070 01 COBREENT-CODES CCOPY ICOP5BS.
000C8 01 COBREENT-CODES.                                000010C0
00009 C * THESE CODES REPRESENT OFFSETS FOR ROUTINE ADDRESSES IN THE 00002000
00010 C * TABLE NAMED REENTSBS. ONLY THE MOST COMMONLY USED VALUES 00003000
00011 C * ARE INCLUDED HERE; THE USERS MANUAL HAS A COMPLETE LIST. 00004000
00012 C * IF OFFSET CDD, THEN TRUE CFFSET--(OFFSET+1) 00005000
00013 C 05 INTSORTC PIC 99 CCMP VALUE 99. 000053C0
00014 C 05 DMS-SNAP PIC 99 CCMP VALUE 95. 000054C0
00015 C 05 MAPFREE PIC 99 CCMP VALUE 91. 000055C0
00016 C 05 FECPRLSE PIC 99 CCMP VALUE 87. 00006000
00017 C 05 FESEND PIC 99 CCMP VALUE 83. 00007000
00018 C 05 FESENO C PIC 99 CCMP VALUE 79. 00008000
00019 C 05 DYN-ALLOCATE PIC 99 CCMP VALUE 75. 000090C0
00020 C 05 DYN-ACCESS PIC 99 CCMP VALUE 71. 00010000
00021 C 05 MAPURGE PIC 99 CCMP VALUE 67. 00011000
00022 C 05 MAPCLR PIC 99 CCMP VALUE 63. 00012CC0
00023 C 05 MAPEND PIC 99 CCMP VALUE 59. 000130C0
00024 C 05 MAPCUT PIC 99 CCMP VALUE 55. 00014000
00025 C 05 MAPIN PIC 99 CCMP VALUE 51. 000150C0
00026 C 05 INTUNSTC PIC 99 CCMP VALUE 47. 00016000
00027 C 05 INTSTORE PIC 99 CCMP VALUE 43. 00017000
00028 C 05 INTFETCH PIC 99 CCMP VALUE 39. 000180C0
00029 C 05 FECPFCBK PIC 99 CCMP VALUE 35. 00019000
00030 C 05 FECPDC PIC 99 CCMP VALUE 31. 00020C00
00031 C 05 DC-WRITEX PIC 99 CCMP VALUE 27. 000210C0
00032 C 05 DC-READX PIC 99 CCMP VALUE 23. 00022000
00033 C 05 DC-WRITE PIC 99 CCMP VALUE 19. 000230C0
00034 C 05 DQ-READ PIC 99 CCMP VALUE 15. 00024000
00035 C 05 DQ-CLOSE PIC 99 CCMP VALUE 11. 00025000
00036 C 05 DC-OPEN PIC 99 CCMP VALUE 07. 000260C0
00037 C 05 DQ-BUILD PIC 99 CCMP VALUE 03. 00027000
00038 C 05 FT-SELECT PIC 99 CCMP VALUE 4. 00028000
00039 C 05 FT-RELEASE PIC 99 CCMP VALUE 8. 000290C0
00040 C 05 FM-READ PIC 99 CCMP VALUE 12. 00030000
00041 C 05 FT-WRITE PIC 99 CCMP VALUE 16. 000310C0
00042 C 05 FM-GET PIC 99 CCMP VALUE 20. 00032000
00043 C 05 FM-PUT PIC 99 CCMP VALUE 24. 00033000
00044 C 05 FT-RELEX PIC 99 CCMP VALUE 28. 000340C0
00045 C 05 FT-FEDV PIC 99 CCMP VALUE 32. 00035000
00046 C 05 COBPUT PIC 99 CCMP VALUE 68. 00036000
00047 C 05 MSGCCL PIC 99 CCMP VALUE 72. 000370C0
00048 C 05 COBSTORF PIC 99 CCMP VALUE 76. 00038000
00049 C 05 CONVERSE PIC 99 CCMP VALUE 80. 00039000
00050 C 05 DBINT PIC 99 CCMP VALUE 84. 000400C0
00051 C 05 LOGPUT PIC 99 CCMP VALUE 88. 00041000
00052 C 05 PAGE-FILE PIC 99 CCMP VALUE 92. 00042000
00053 C 05 FM-GETV PIC 99 CCMP VALUE 96. 00043CC0
00054 C 05 FM-PUTV PIC 999 CCMP VALUE 100. 00044000

```

Figure 49. Sample COBOL Subroutine (Page 1 of 3)

```

2

00055 C * CODES 104 AND UP INDICATE USER ADDITIONS TO THE TABLE 00045000

00057 000080 LINKAGE SECTION.
00058 000090 01 ICOM-RETURN PIC S9(7) COMP.
00059 000100 01 DYNAMIC-WORKING-SPACE.
00060 000110 02 STATWD PIC S9(7) COMP SYNC.
00061 000120 02 FH-STATUS REDEFINES STATWD.
00062 000130 04 FH-STAT1 PIC X.
00063 000140 88 IO-ERROR VALUE 1.
00064 000150 88 NCT-FOUND VALUE 2.
00065 000160 88 NO-DC VALUE 9.
00066 000170 04 FILLER PIC X(3).
00067 000180 02 EXTDSCT PIC X(48).
00068 000185 02 DD-NAME PIC X(8).
00069 000190 01 PART-RECORD PIC X(100).
00070 000200 01 RBN PIC X(3).

00072 000220 PRCCEDURE DIVISION USING ICOMP-RETURN
00073 000230 DYNAMIC-WORKING-SPACE
00074 000240 PART-RECORD
00075 000250 RBN.
00076 000260 MCVE SPACES TO FH-STATUS.
00077 000265 MCVE DD-PART TO DD-NAME.
00078 000270 CALL 'COBREENT' USING FH-SELECT
00079 000280 EXTDSCT
00080 000290 FH-STATLS
00081 000300 DC-NAME.
00082 000310 IF NO-DC
00083 000320 MOVE 9 TO ICOMP-RETURN
00084 000330 ELSE
00085 000340 MCVE SPACES TO FH-STATUS
00086 000350 PERFORM FH-READ-RTN
00087 000360 IF IO-ERROR
00088 000370 MOVE 1 TO ICOMP-RETURN
00089 000380 ELSE
00090 000390 IF NOT-FOUND
00091 000400 MOVE 2 TO ICOMP-RETURN.
00092 000410 IF ICOM-RETURN NOT EQUAL TO 9
00093 000420 MCVE SPACES TO FH-STATUS
00094 000430 CALL 'COBREENT' USING FH-RELEASE
00095 000440 EXTDSCT
00096 000450 FH-STATLS.
00097 000460 GCBACK.
00098 000470 FH-READ-RTN.
00099 000480 CALL 'COBREENT' USING FH-READ
00100 000490 EXTDSCT
00101 000500 FH-STATUS
00102 000510 PART-RECORD
00103 000520 RBN.

```

Figure 49. Sample COBOL Subroutine (Page 2 of 3)

INTRNL NAME	LVL	SOURCE NAME	BASE	DISPL	INTRNL NAME	DEFINITION	USAGE	R	O	D	O	M
DNM-1-050	77	DC-PART	BL=1	00C	DNM-1-050	DS 8C	DISP					
DNM-1-067	01	COBREENT-COCES	BL=1	008	DNM-1-067	DS OCL84	GROUP					
DNM-1-054	02	INTSCRTC	BL=1	008	DNM-1-067	DS 2C	COMP					
DNM-1-112	02	DMS-SNAP	BL=1	00C	DNP-1-C94	DS 2C	COMP					
DNM-1-130	02	PAPFREE	BL=1	00C	DNP-1-112	DS 2C	COMP					
DNM-1-147	02	FECMLSE	BL=1	00E	DNP-1-130	DS 2C	COMP					
DNP-1-165	02	FESEND	BL=1	010	DNM-1-147	DS 2C	COMP					
DNP-1-161	02	FESEND	BL=1	012	DNM-1-165	DS 2C	COMP					
DNM-1-198	02	DYN-ALLOCATE	BL=1	014	DNP-1-181	DS 2C	COMP					
DNM-1-220	02	DYN-ACCESS	BL=1	016	DNP-1-198	DS 2C	COMP					
DNM-1-240	02	PAPLRGE	BL=1	018	DNP-1-220	DS 2C	COMP					
DNM-1-257	02	MAPCLR	BL=1	01A	DAM-1-240	DS 2C	COMP					
DNP-1-273	02	MAPEND	BL=1	01C	DNP-1-257	DS 2C	COMP					
DMA-1-289	02	MAPCUT	BL=1	01E	DNP-1-273	DS 2C	COMP					
DNM-1-305	02	MAPIN	BL=1	02C	DNP-1-289	DS 2C	COMP					
DNP-1-320	02	INTUNSTO	BL=1	022	DNM-1-305	DS 2C	COMP					
DNP-1-338	02	INTSTORE	BL=1	024	DNP-1-320	DS 2C	COMP					
DNM-1-356	02	INTFETCH	BL=1	026	DNP-1-338	DS 2C	COMP					
DNP-1-374	02	FECMFBK	BL=1	028	DNP-1-356	DS 2C	COMP					
DNM-1-392	02	FECMDDO	BL=1	02A	DNP-1-374	DS 2C	COMP					
DNP-1-409	02	DC-WRITE	BL=1	02C	DNM-1-392	DS 2C	COMP					
DNP-1-431	02	DC-READ	BL=1	02E	DNP-1-409	DS 2C	COMP					
DNM-1-449	02	DC-WRITE	BL=1	030	DNP-1-431	DS 2C	COMP					
DNM-1-467	02	DC-READ	BL=1	032	DNP-1-449	DS 2C	COMP					
DNP-1-484	02	DC-CLOSE	BL=1	034	DNP-1-467	DS 2C	COMP					
DNM-2-000	02	DC-OPEN	BL=1	036	DNP-1-484	DS 2C	COMP					
DNM-2-017	02	DC-BUILD	BL=1	038	DNP-2-000	DS 2C	COMP					
DNM-2-035	02	FP-SELECT	BL=1	03A	DNP-2-017	DS 2C	COMP					
DNP-2-054	02	FP-RELEASE	BL=1	03E	DNP-2-035	DS 2C	COMP					
DNP-2-074	02	FP-READ	BL=1	03E	DNP-2-054	DS 2C	COMP					
DNP-2-091	02	FP-WRITE	BL=1	040	DNP-2-074	DS 2C	COMP					
DNM-2-109	02	FP-GET	BL=1	042	DNP-2-091	DS 2C	COMP					
DNM-2-125	02	FP-PUT	BL=1	044	DNP-2-109	DS 2C	COMP					
DNM-2-141	02	FP-RELEX	BL=1	046	DNP-2-125	DS 2C	COMP					
DNP-2-159	02	FP-FEDV	BL=1	048	DNP-2-141	DS 2C	COMP					
DNP-2-176	02	COBPUT	BL=1	048	DNP-2-159	DS 2C	COMP					
DNM-2-192	02	MSGCCL	BL=1	04C	DNP-2-176	DS 2C	COMP					
DNM-2-211	02	CCBSTORE	BL=1	04E	DNP-2-192	DS 2C	COMP					
DNP-2-229	02	CONVERSE	BL=1	050	DNP-2-211	DS 2C	COMP					
DNP-2-247	02	DBINT	BL=1	052	DNP-2-229	DS 2C	COMP					
DNM-2-262	02	LCGPUT	BL=1	054	DNP-2-247	DS 2C	COMP					
DNM-2-278	02	PAGE-FILE	BL=1	056	DNP-2-262	DS 2C	COMP					
DNP-2-297	02	FM-GETV	BL=1	058	DNP-2-278	DS 2C	COMP					
DNP-2-314	02	FM-PUTV	BL=1	05A	DNP-2-297	DS 2C	COMP					
DNM-2-331	01	ICOM-RETURN	BL=3	000	DNP-2-314	DS 2C	COMP					
DNM-2-352	01	DYNAMIC-WORKING-SPACE	BL=4	000	DNP-2-331	DS 4C	COMP					
DNM-2-389	02	STATW	BL=4	000	DNP-2-352	DS OCL60	GROUP					
DNP-2-405	02	FM-STATUS	BL=4	000	DNP-2-389	DS 1F	COMP					
DNP-2-427	03	FM-STAT1	BL=4	000	DNP-2-405	DS OCL4	GROUP					
DNP-2-448	88	IO-ERRDR	BL=4	000	DNP-2-427	DS 1C	DISP					
DNM-2-469	88	ACT-FOUND			DNP-2-448							
DNP-2-491	88	MC-CD			DNP-2-469							

Figure 49. Sample COBOL Subroutine (Page 3 of 3)

Chapter 11

SUBSYSTEM TESTING

11.1 INTRODUCTION

After a new subsystem has been thoroughly desk-checked and compiles cleanly, it becomes necessary to test the subsystem's execution under the control of Intercomm. Three methods of testing are available:

- Simulated--batch execution of Intercomm with a simulated BTAM Front End. Message input streams are created via the CREATSIM utility program. Additionally, 3270 terminal input and output screen, or output printer, images are formatted if the SIM3270 utility is implemented for the simulation mode execution. Illustration of this mode of testing is provided in this Chapter, and is particularly useful for testing messages processed via the Message Mapping Utilities.
- Test Mode--batch execution of a Back End Intercomm with message input from a card-image data set, as described in Chapter 12.
- On-line Testing--an on-line system is necessary for final testing of all error conditions, multithread processing, etc. and can be either a single region system, or a satellite region used primarily for testing within a Multiregion production system.

11.2 DEBUGGING APPLICATION PROGRAM PROBLEMS

Text and descriptions of error messages issued by Intercomm as a result of invalid program logic paths, along with descriptions of general debugging techniques for accompanying snaps and abends are available in Messages and Codes. Additional debugging facilities such as dispatcher trace reports, thread dumps and indicative dumps are also described in the Operating Reference Manual.

11.3 TESTING A SUBSYSTEM WITH THE FRONT END SIMULATOR

As described in the Operating Reference Manual, a test execution with a simulated Front End is very useful to determine Front End message interface problems that may be harder to debug when using an on-line test system. Although the simulation is of certain BTAM devices, including a local 3270, the access method interfaces required for a remote 3270 or a TCAM or VTAM Front End are essentially transparent to the application programmer as the interface dependent code is handled by Intercomm.

This chapter illustrates testing of the subsystem and subroutine described in Chapter 10 using the BTAM simulator for 3270 CRT messages processed via maps defined for the Message Mapping Utilities.

To test an application system in a simulated Intercomm environment, do the following:

NOTE: Steps preceded by an asterisk (*) may often be performed for the application programmer by an installation's Intercomm System Manager. Appendix C summarizes the Intercomm Table entries.

1. Compile and linkedit the user subsystem(s) and subroutine(s), if any. Appendix A describes Intercomm-supplied COBOL JCL procedures.
- *2. Create or add to a USRSCTS member on a user test library to contain a Subsystem Control Table Entry (SYCTTBL macro) which describes the subsystem. Reassemble and link INTSCT which copies the USRSCTS member from the test library (see Figure 50).
- *3. Define input message verbs in the copy member USRBTVRB via BTVERB macros and reassemble and link the Front End Verb Table BTVRBTB (see Figure 50).
- *4. Code a SUBMODS macro addition to the COPY member USRSUBS to define the COBOL subroutine and reassemble and linkedit REENTSBS which copies USRSUBS (see Figure 50).
5. Assemble and linkedit MMU maps (Map Group STKSTAT--see Figure 51) to the MMU load module library. Load maps to the appropriate Store/Fetch data set. See Message Mapping Utilities.
6. Prepare input test message data set(s) using the CREATSIM utility as illustrated in Figure 52. Note that the first message turns on the DWS protection option described in Chapter 3. The next message generates, via the MMU command MMUC, the screen template to be used for entering an inquiry transaction. All subsequent input messages are for testing the COBOL subsystem and subroutine, including input error conditions handled by the application program.

- *7. Add control cards to the linkedit deck for the user programs, unless the routines are dynamically loadable (see Figure 53).
- *8. Add INCLUDE statements for the simulator (BTAMSIM) and 3270 display formatter (SIM3270) to an Intercomm linkedit deck which was created for the BTAM Front End (see Figure 53).
- *9. Linkedit to create a new Intercomm load module (see Figure 53).
10. Add DD statements to the Intercomm execution JCL for the printed SIM3270 output and the input message data set(s) (see Figure 53).
11. Create test data sets and add DD statements for them to the execution JCL (see Figure 53). Note that if a VSAM data set is used with a user catalog, place the STEPCAT DD statement after the //PMISTOP DD statement (see Figure 53); do not use a JOBCAT DD statement. STEPCAT should be omitted if using ICF catalogs.
- *12. Execute in simulation mode:
 - a. Single-thread test all subsystems; to test a reentrant subsystem, specify MNCL=1 in the subsystem's SYCTTBL macro.
 - b. Multithread test reentrant subsystems (change MNCL) using several test message input data sets or use a single data set as input from more than one terminal.

The parameter 'STARTUP' must be coded on the Intercomm EXEC statement. Figure 53 illustrates a sample execution deck with test message input (DD statement TEST1) for the sample inquiry program and JCL to print the system log.

The resulting SIM3270 printouts for the simulated execution of the sample inquiry subsystem are illustrated in Figure 54. Note that the underlined positions on each screen display indicate attribute byte positions; codes are described under the display. On an actual terminal, the attribute byte position appears as a blank to the terminal operator. See Message Mapping Utilities and IBM documentation on programming for the 3270 CRT for further information on attribute codes.

The Intercomm Log printed after the simulated execution of the sample inquiry subsystem is shown in Figure 55.

13. Test the subsystem concurrently with other application subsystems.

```

//TABLES      JOB
//**
//**          DEFINE SYCTTBL FOR SUBSYSTEM
//**
//STEP1       EXEC  LIBELINK,Q=TEST,NAME=INTSCT,LMOD=INTSCT
//LIB.SYSIN   DD   *
./ ADD NAME=USRSCTS
./ NUMBER    NEW1=100,INCR=100
USRSCTS      DS   0H
RQ           SYCTTBL SUBH=R,SUBC=Q,SBSP=SQCOBOLA,LANG=RCOB,OVLY=0,      X
              NUMCL=10,MNCL=2,TCTV=60,GET=640
/*
//ASM.SYSIN   DD   DSN=INT.SYMREL(INTSCT),DISP=SHR
//**
//**          DEFINE BTVERB FOR SUBSYSTEM
//**
//STEP2       EXEC  LIBELINK,Q=TEST,NAME=BTVRBTB,LMOD=BTVRBTB
//LIB.SYSIN   DD   *
./ ADD NAME=USRBTVRB
./ NUMBER    NEW1=100,INCR=100
USRBTVRB     DS   0H
              BTVERB VERB=MURQ,SSCH=R,SSC=Q,CONV=18000
/*
//ASM.SYSIN   DD   DSN=INT.SYMREL(BTVRBTB),DISP=SHR
//**
//**          DEFINE SUBMODS FOR SUBROUTINE
//**
//STEP3       EXEC  LIBELINK,Q=TEST,NAME=REENTSBS,LMOD=REENTSBS
//LIB.SYSIN   DD   *
./ ADD NAME=USRSUBS
./ NUMBER    NEW1=100,INCR=100
USRSUBS      DS   0H
              SUBMODS LNAME=SQCOBOLB,TYPE=COBOL,DELTIME=30,PARM=RC,      X
              GET=60
/*
//ASM.SYSIN   DD   DSN=INT.SYMREL(REENTSBS),DISP=SHR
//

```

Figure 50. Table Updates to Implement Simulation Mode Testing

STKSTAT	MAPGROUP MODE=1/0,DEVICE=IBM3270	00000010
MAP1	MAP SIZE=(20,80),START=(1,1)	00000020
VERB	FIELD RELPOS=VERB	00000030
	FIELD RELPOS=(1,7),INITIAL=*ENTER TRANSACTION CODE*,ATTRIB=PSN	00000040
	FIELD RELPOS=(3,23),INITIAL=*ENTER DATA:*,ATTRIB=PSN	00000050
	FIELD RELPOS=(5,7),INITIAL=*PART NO:*,ATTRIB=PAHSEL	00000060
PARTNO	SEGMENT	00000065
FILLER	FIELD RELPOS=(5,16),FORMAT=(4,,ZD),ATTRIB=UNN	00000070
RBNBYTE	FIELD RELPOS=(5,20),FORMAT=(1,,ZD)	00000075
	SEGMENT	00000077
	FIELD RELPOS=(5,22),FORMAT=1,ATTRIB=PSN	00000080
	FIELD RELPOS=(6,7),INITIAL=*WHS NO:*,ATTRIB=PAHSEL	00000090
WHSNO	FIELD RELPOS=(6,15),FORMAT=(3,,ZD),ATTRIB=UNN	00000100
	FIELD RELPOS=(6,19),FORMAT=1,ATTRIB=PSN	00000110
	FIELD RELPOS=(8,23),INITIAL=*STOCK STATUS:*,ATTRIB=PSN	00000120
	FIELD RELPOS=(10,7),INITIAL=*DESCRIPTION:*,ATTRIB=PSN	00000130
PRTDATA	FIELD RELPOS=(10,20),FORMAT=54,ATTRIB=UAN	00000140
	FIELD RELPOS=(10,76),FORMAT=1,ATTRIB=PSN	00000150
	FIELD RELPOS=(11,7),INITIAL=*ORDER UNITS:*,ATTRIB=PSN	00000160
ORDUNT	FIELD RELPOS=(11,20),FORMAT=5,ATTRIB=UAN	00000170
	FIELD RELPOS=(11,26),FORMAT=1,ATTRIB=PSN	00000180
	FIELD RELPOS=(11,40),INITIAL=*PRICE:*,ATTRIB=PSN	00000190
PRTPRC	FIELD RELPOS=(11,47),FORMAT=(9,4,SPDS4),ATTRIB=UAN	00000200
	FIELD RELPOS=(11,57),FORMAT=1,ATTRIB=PSN	00000210
	FIELD RELPOS=(13,23),INITIAL=*STOCK STATUS AT WAREHOUSE:*,ATTRIB=PSN	X00000220
	FIELD RELPOS=(15,7),INITIAL=*LOCATION:*,ATTRIB=PSN	00000230
WHSLOC	FIELD RELPOS=(15,17),FORMAT=23,ATTRIB=UAN	00000240
	FIELD RELPOS=(15,41),FORMAT=1,ATTRIB=PSN	00000250
	FIELD RELPOS=(16,7),INITIAL=*ON HAND:*,ATTRIB=PSN	00000260
STKLEV	FIELD RELPOS=(16,16),FORMAT=(7,4,PD),ATTRIB=UAN	00000270
	FIELD RELPOS=(16,24),FORMAT=1,ATTRIB=PSN	00000280
	FIELD RELPOS=(16,40),INITIAL=*AS OF:*,ATTRIB=PSN	00000290
LEVDATE	FIELD RELPOS=(16,47),FORMAT=8,ATTRIB=UAN	00000300
	FIELD RELPOS=(16,56),FORMAT=1,ATTRIB=PSN	00000310
	FIELD RELPOS=(17,7),INITIAL=*ON ORDER:*,ATTRIB=PSN	00000320
STKORD	FIELD RELPOS=(17,17),FORMAT=(7,4,PD),ATTRIB=UAN	00000330
	FIELD RELPOS=(17,25),FORMAT=1,ATTRIB=PSN	00000340
	FIELD RELPOS=(17,40),INITIAL=*AS OF:*,ATTRIB=PSN	00000350
ORODATE	FIELD RELPOS=(17,47),FORMAT=8,ATTRIB=UAN	00000360
	FIELD RELPOS=(17,56),FORMAT=1,ATTRIB=PSN	00000370
ERRMAP	MAP SIZE=(15,80),START=(10,1)	00000380
	FIELD RELPOS=(1,1),ATTRIB=SUPR,INITIAL=X*125B5F*	00000390
***	ABOVE CLEARS STOCK STATUS INFO. WHEN ERROR MESSAGE APPEARS ***	00000400
	FIELD RELPOS=(14,33),INITIAL=*ERROR MESSAGE:*,ATTRIB=PAHSEL	00000410
ERRMS6	FIELD RELPOS=(15,10),FORMAT=50,ATTRIB=UAHSEL	00000420
	FIELD RELPOS=(15,61),FORMAT=1,ATTRIB=PSN	00000430
	ENDGROUP	00000440
	END	00000450
		00000460

Figure 51. MMU Maps Used by Sample Subsystem

```

//CREATSIM JOB                                00000100
//CRS      PROC  T=                            00000200
//* SCRATCH OLD TEST INPUT DATA SET (IF ANY)  00000300
//S        EXEC  PGM=IEFBRI4                   00000400
//SCR      DD   DSN=INT.T&T,DISP=(OLD,DELETE)  00000500
//CRS      EXEC  PGM=CREATSIM                   00000600
//* CREATE NEW TEST INPUT DATA STREAM FOR 3270 DEVICE 00000700
//STEPLIB DD   DSN=INT.MOOLIB,DISP=SHR         00000800
//        DD   DSN=INT.MODREL,DISP=SHR         00000900
//SYSPRINT DD  SYSOUT=A                         00001000
//SYSUT2   DD   DSN=INT.T&T,DISP=(,CATLG,CATLG),UNIT=3350,
//        VOL=SER=INT001,SPACE=(TRK,(1,1))     00001100
//        EXEC  PGM=IEBPTPCH                     00001200
//* PRINT MESSAGES GENERATED ON TEST INPUT DATA SET 00001300
//SYSPRINT DD  SYSOUT=A                         00001400
//SYSUT1   DD   DSN=*.CRS.SYSUT2,DISP=OLD      00001500
//SYSUT2   DD   SYSOUT=A                         00001600
//        PEND                                   00001700
//* FOR THIS EXECUTION OF CREATSIM, THE END-OF-CARD CHARACTER IS A
//* SEMI-COLON, (USE ALSO AFTER THE VERB-FRONT END SEES THE SBA),
//* THE MESSAGE END CHARACTER IS AN EXCLAMATION POINT (EOB). 00001800
//EXECCRS EXEC CRS,T=TEST1                     00001900
//CRS.SYSIN DD *                                00002000
GRAPHIC,ADD,;FF                                CONTINUATION CODE 00002100
GRAPHIC,ADD,<7D                                ENTER KEY         00002200
SBA,M2                                         USING MODEL 2 SCREEN SIZE 00002300
< STRT,DWSCK!                                 00002400
< MMUC,SHOW,(STKSTAT,MAP1)!                  00002500
< ;                                           00002600
SBA,0102;                                     00002700
MURQ;                                         00002800
SBA,0516;                                     00002900
12345;                                        00003000
SBA,0615;                                     00003100
200!                                         00003200
< ;                                           00003300
SBA,0102;                                     00003400
MURQ;                                         00003500
SBA,0516;                                     00003600
55555;                                        00003700
SBA,0615;                                     00003800
200!                                         00003900
< ;                                           00004000
SBA,0102;                                     00004100
MURQ;                                         00004200
SBA,0516;                                     00004300
1234R;                                        00004400
SBA,0615;                                     00004500
300!                                         00004600

```

Figure 52. Input Test Messages Generated via CREATSIM (Page 1 of 2)

```

< ;
SBA,0102;
MURQ;
SBA,0516;
12341;
SBA,0615;
600!
< ;
SBA,0102;
MURQ;
SBA,0516;
A2345;
SBA,0615;
200!
< ;
SBA,0102;
MURQ;
SBA,0516;
12345;
SBA,0615;
900!
< ;
SBA,0102;
MURQ;
SBA,0516;
1234X;
SBA,0615;
20Y!
< ;
SBA,0102;
MURQ;
SBA,0516;
12349;
SBA,0615;
100!
< ;
SBA,0102;
MURQ;
SBA,0516;
12342;
SBA,0615;
100!
//DUMP.SYSIN DD *
PRINT TYPORG=PS,TOTCONV=XE,CNTRL=2
//
00005000
00005100
00005200
00005300
00005400
00005500
00005600
00005700
00005800
00005900
00006000
00006100
00006200
00006300
00006400
00006500
00006600
00006700
00006800
00006900
00007000
00007100
00007200
00007300
00007400
00007500
00007600
00007700
00007800
00007900
00008000
00008100
00008200
00008300
00008400
00008500
00008600
00008700
00008800
00008900
00009000
00009100
00009200
00009300
00009400

```

Figure 52. Input Test Messages Generated via CREATESIM (Page 2 of 2)

```

//EXECUTEST JOB (ICOMTEST,,20),'COBOL TEST',CLASS=A,
//  RESTART=(GENLINK.ASM)
//PROCLIB DD DSN=INT.PROCLIB,DISP=SHR          (AS NEEDED)
//*****
//* THE RESTART PARM IN THE JOB STATEMENT RESTARTS THE TEST AT THE *
//* BEGINNING. IF YOU WISH TO RESTART AT A DIFFERENT STEP, CODE *
//* RESTART=STEPNAME OR RESTART=STEPNAME.PROCSTEPNAME *
//* *
//* NOTE: WHEN USING A VSAM FILE, IT MAY BE NECESSARY TO EXECUTE *
//* IDCAMS TO VERIFY THE FILE IF A PREVIOUS EXECUTION ABENDED. *
//*****
//*****
//* STEP GENLINK GENERATES A STANDARD BTAM FRONT END LINKEDIT DECK *
//* VIA ASSEMBLY OF THE ICOMLINK MACRO. IF ONLY A VTAM FRONT END IS *
//* USED ON-LINE, A SETGLOBE WITH THE BTAM GLOBAL SET TO 1 MUST BE *
//* IN THE LIBRARY SPECIFIED BY THE Q= PARM. ADD OR CHANGE PARMS FOR *
//* THE ICOMLINK MACRO BASED ON INTERCOMM FACILITIES USED. *
//* THE GENERATED DECK (SIMLINK) IS PLACED ON INT.SYMTEST. *
//* NOTE: THE SPECIFIED FRONT END NETWORK TABLE (FENETWRK) THAT IS *
//* ON MODREL CONTAINS A DEFINITION FOR THE TEST TERMINAL *
//* TEST1 AS A LOCAL BTAM 3270 CRT. (COPY TO MODTEST) *
//* STEP NUM NUMBERS GENERATED LINK DECK IN INCREMENTS OF 1000 *
//* FOR ADDING INCLUDE STATEMENTS IN GENINCL STEP. *
//*****
//GENLINK EXEC ASMPC,DECK=DECK,Q=TEST
//ASM.SYSIN DD *
//          ICOMLINK MMU=YES,FETABLE=FENETWRK,COBOL=YES,RECOBOL=YES
//          END
//SYSPUNCH DD DSN=INT.SYMTEST(SIMLINK),DISP=SHR
//*          NUMBER GENERATED LINKEDIT DECK
//NUM      EXEC  LIBE,Q=TEST
//LIB.SYSIN DD *
//. / CHANGE NAME=SIMLINK
//. / NUMBER NEW1=1000,INCR=1000
//*
//*****
//* STEPS SCRSCR AND ALLOCSCR DELETE AND RE-ALLOCATE THE LOAD *
//* MODULE LIBRARY USED IN THE TEST (ALSO USED FOR DYNLLIB) *
//*****
//SCRSCR  EXEC PGM=IEFBR14
//FILE1   DD DSN=INT.MODSCR,DISP=(OLD,DELETE)
//ALLOCSCR EXEC PGM=IEFBR14
//A       DD DSN=INT.MODSCR,DISP=(,CATLG),UNIT=SYSDA,
//        DCB=INT.MODREL,VOL=SER=INT001,
//        SPACE=(TRK,(30,,7))          7 RECORDS PER TRK/3380

```

Figure 53. Linkedit and Execution JCL for Simulation Mode (Page 1 of 3)

```

/*****
/** STEP GENINCL CREATES INCLUDE DECK USED BY THE LINK EDIT STEP: *
/** THE ADDED INCLUDE STATEMENTS ARE FOR THE SAMPLE SUBSYSTEM *
/** (ASSUMED TO HAVE BEEN LINKED TO MODTEST), *
/** AND THE REQUIRED SIMULATION MODE MODULES. *
/** IF THE TEST1 TERMINAL IS NOT IN THE SYSTEM PMISTATB TABLE, USE: *
/** INCLUDE MODREL(PMISTATB) *
/** INCLUDE MODREL(PMIDEVTB) *
/** INCLUDE MODREL(PMIBROAD) *
/** THE ABOVE ASSUMES THE CONTROL TERMINAL IS NAMED CNT01. *
/*****
//GENINCL EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=A TO PRINT CHANGES
//SYSUT1 DD DSN=INT.SYMTEST,DISP=SHR
//SYSUT2 DD DSN=&&INCL,DISP=(,PASS),UNIT=SYSDA,SPACE=(TRK,(8,1,1)),
// DCB=(BLKSIZE=80,LRECL=80)
//SYSIN DD *
./ CHANGE NAME=SIMLINK,LIST=ALL
INCLUDE SYSLIB(SQCOBOLA) TEST SUBSYSTEM 00000010
INCLUDE SYSLIB(BTAMSIM) BTAM SIMULATOR 00000020
INCLUDE SYSLIB(SIM3270) SCREEN PRINTING 00000030
/*
/*****
/** LINK EDIT THE TEST INTERCOMM SYSTEM. *
/** NOTE THAT THE INTERCOMM LKEDT PROC PLACES THE LOAD MODULE ON *
/** THE MODSCR LOAD LIBRARY CREATED ABOVE. *
/** IT IS NOT NECESSARY TO RE-DO THE WHOLE LINK TO REPLACE 1 MODULE *
/** IN THIS CASE, ALL YOU SHOULD DO IS: *
/** 1) REASSEMBLE OR RECOMPILE THE CHANGED NEW MODULE INTO A *
/** SEPARATE LOAD LIBRARY *
/** 2) CHANGE THE SYSIN DD STATEMENT TO //SYSIN DD * *
/** FOLLOW IT WITH INCLUDE CARDS *
/** FOR THE MODULES YOU WISH TO REPLACE *
/** 3) FOLLOW THOSE INCLUDES WITH THE FOLLOWING 3 CARDS: *
/** INCLUDE SYSLMOD(SIMICOM) *
/** ENTRY PMISTUP *
/** NAME SIMICOM(R) *
/** 4) INSERT A DD STATEMENT FOR THE LOAD LIBRARY ON WHICH THE *
/** REPLACEMENT MODULES RESIDE *
/** 5) CHANGE THE RESTART PARM ON THE JOB STATEMENT *
/** TO POINT TO THE LKED.LKED STEP. *
/*****
//LKED EXEC LKEDT,Q=TEST,LMOD=SIMICOM,
// PARM.LKED='LIST,LET,XREF,NCAL,SIZE=(250K,100K)'
//SYSIN DD DSN=&&INCL(SIMLINK),DISP=(OLD,PASS)
//MODREL DD DSN=INT.MODREL,DISP=SHR
/*****
/** LINKEDIT THE DYNAMICALLY LOADABLE SUBROUTINE *
/** FROM MODTEST (Q= POINTS TO IT) TO MODSCR *
/*****
//LINKSQB EXEC LKEDT,Q=TEST,LMOD=SQCOBOLB
//SYSIN DD *
INCLUDE SYSLIB(SQCOBOLB)

```

Figure 53. Linkedit and Execution JCL for Simulation Mode (Page 2 of 3)


```

//*****
//* EXECUTE INTERCOMM IN SIMULATION MODE *
//*****
//GO EXEC PGM=SIMICOM,PARM='STARTUP',TIME=(,30)
//STEPLIB DD DSN=INT.MODSCR,DISP=(OLD,PASS)
// DD DSN=INT.MODLIB,DISP=SHR
// DD DSN=INT.MODREL,DISP=SHR
// DD DSN=SYS1.COBLIB,DISP=SHR COBOL LOAD LIBRARY
//INTERLOG DD DSN=INTLOG,DISP=(NEW,PASS),
// DCB=(DSORG=PS,RECFM=VB,BLKSIZE=4096,LRECL=4092,NCP=8,OPTCD=C),
// SPACE=(TRK,(10,5)),VOL=SER=INT100,UNIT=SYSDA
//SMLOG DD SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=120,RECFM=FA)
//STSLOG DD SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=120,RECFM=FA)
//SYSPRINT DD SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=141,LRECL=137,RECFM=VA)
//RCT000 DD DSN=INT.RCT000,DISP=SHR,DCB=(DSORG=DA,OPTCD=RF)
//PMIQUE DD DSN=INT.PMIQUE,DCB=(DSORG=DA,OPTCD=R),DISP=SHR
//BTAMQ DD DSN=INT.BTAMQ,DCB=(DSORG=DA,OPTCD=R),DISP=SHR
//INTSTOR2 DD DSN=INTSTOR2,DCB=(DSORG=DA,OPTCD=EF,LIMCT=3),DISP=SHR
//INTSTOR3 DD DSN=INTSTOR3,DCB=(DSORG=DA,OPTCD=EF,LIMCT=3),DISP=SHR
//* TEST DATA SETS FOR SAMPLE SUBSYSTEM
//STOKFILE DD DSN=VSAMSD1.STCKFILE.CLUSTER,DISP=OLD,
// AMP=(AMORG,'RECFM=F')
//PARTFILE DD DSN=INT.BETA.PARTFILE,DISP=OLD,
// DCB=(DSORG=DA,OPTCD=R)
//* DATA SETS FOR SIMULATED TERMINAL -- TEST1
//TEST1 DD DSN=INT.TTEST1,DCB=DSORG=PS,DISP=OLD
//SCRTEST1 DD SYSOUT=A,DCB=(DSORG=PS,RECFM=FA,BLKSIZE=121)
//SIMCARDS DD *
TEST1,002
//PMISTOP DD DUMMY DELIMIT INTERCOMM FILES
//* FAR PARAMETERS
//* (TO USE, CHANGE ICOMIN TO DD *, FOLLOW WITH FARS INLINE)
//ICOMIN DD DUMMY
//* DYNAMIC LINKEDIT DATA SETS (IF NEEDED)
//DYNLLIB DD DSN=INT.MODSCR,DISP=(OLD,PASS)
//DYNLPRNT DD SYSOUT=A
//DYNLWORK DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(,PASS) (REL 9 ONLY)
//*
//STEPCAT DD DSN=VSAMSD1,DISP=SHR (IF NEEDED)
//SNAPDD DD SYSOUT=A,SPACE=(CYL,5),FREE=CLOSE
//SYSUDUMP DD SYSOUT=A
//*
//ABNLIGNR DD DUMMY FORCE ABEND-AID TO IGNORE DUMP (PRODUCE IBM DUMP)
//*****
//* PRINT INTERCOMM LOG GENERATED BY THE TEST *
//*****
//INTERLOG EXEC PGM=LOGPRINT,COND=EVER
//STEPLIB DD DSN=INT.MODREL,DISP=SHR
//SYSPRINT DD SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=121)
//INTERLOG DD DSN=INTLOG,DISP=OLD,DCB=BLKSIZE=5000
//SYSIN DD DUMMY
//

```

Figure 53. Linkedit and Execution JCL for Simulation Mode (Page 3 of 3)

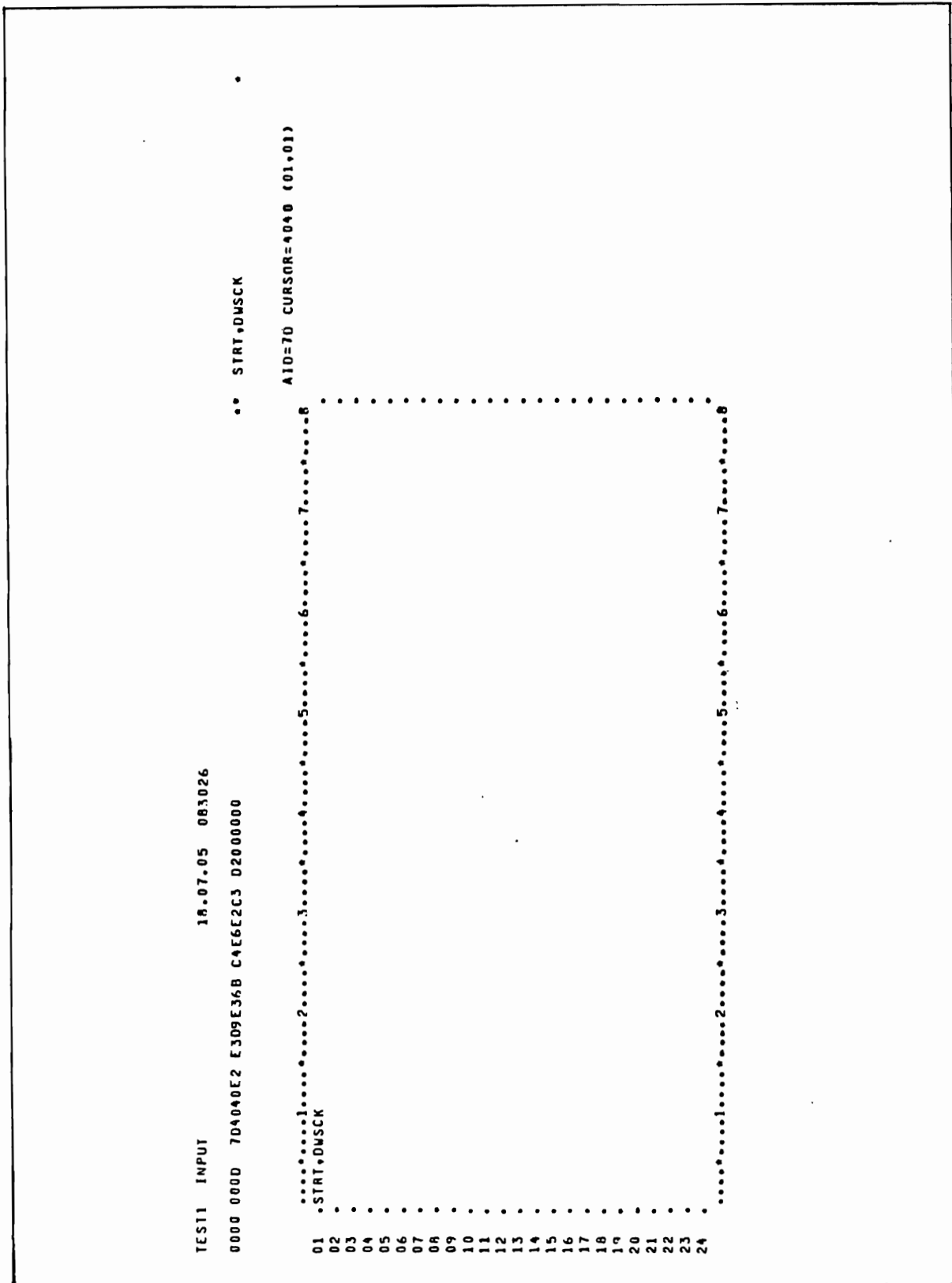


Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 1 of 22)

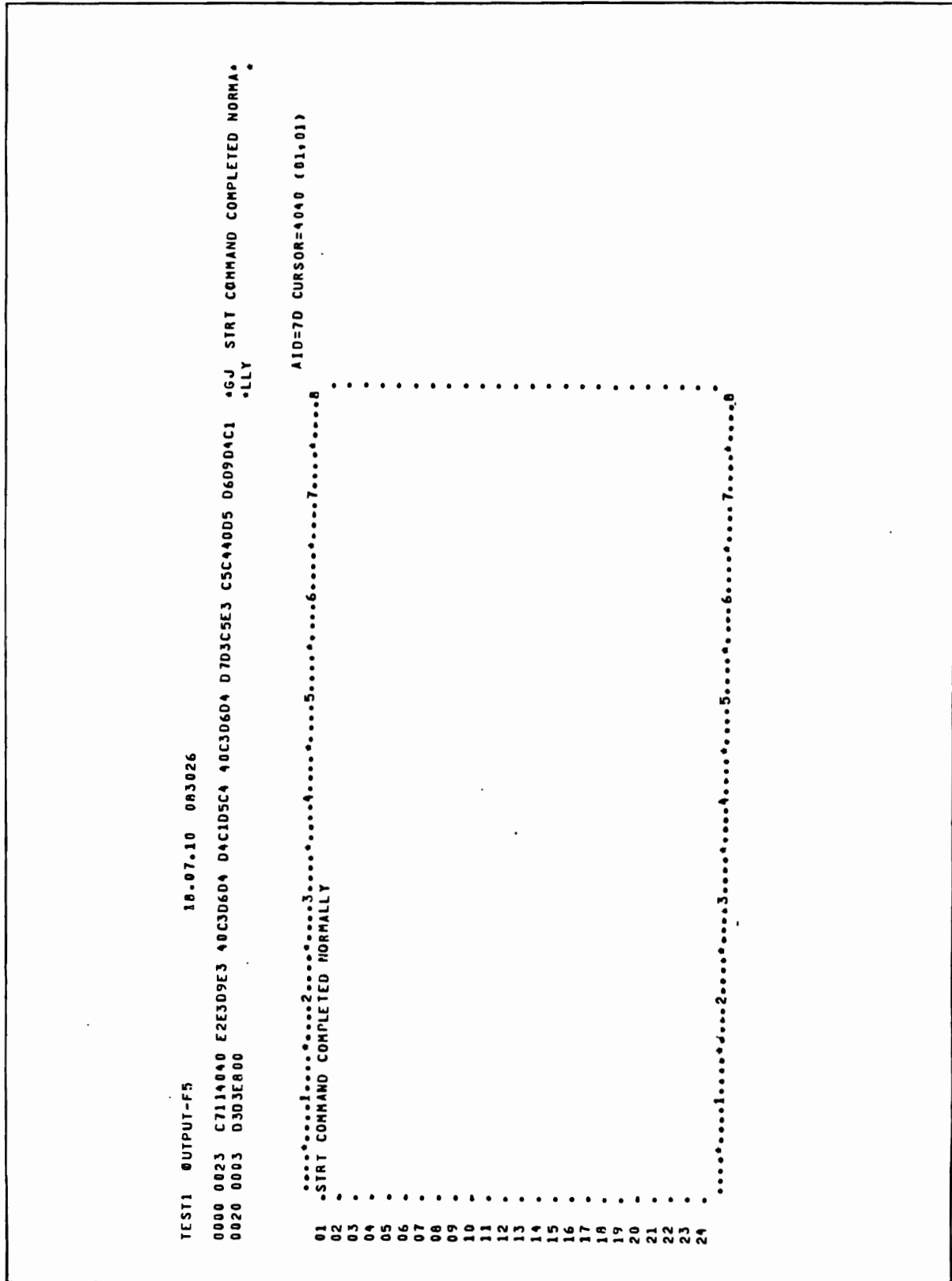


Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 2 of 22)

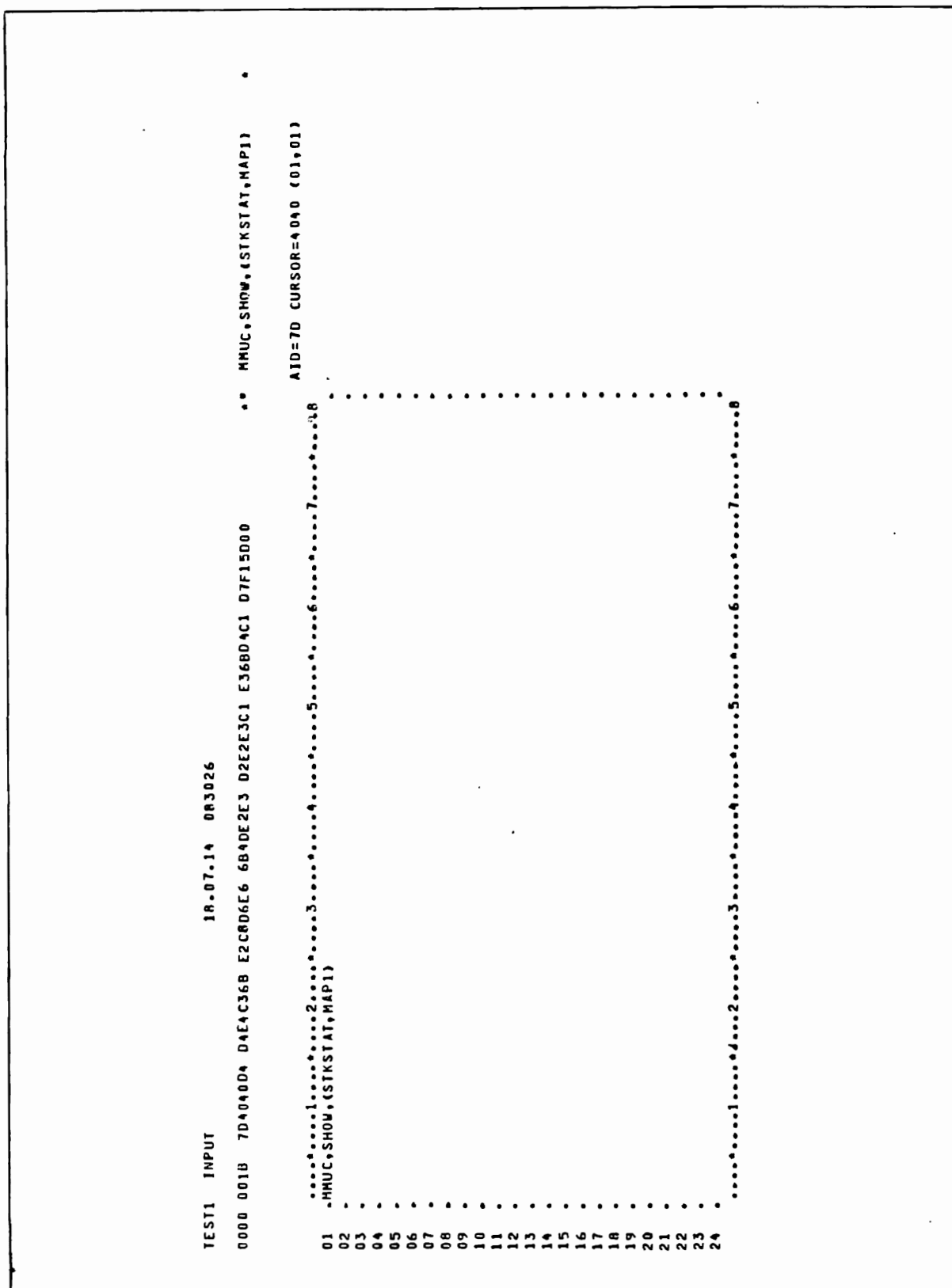


Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 3 of 22)

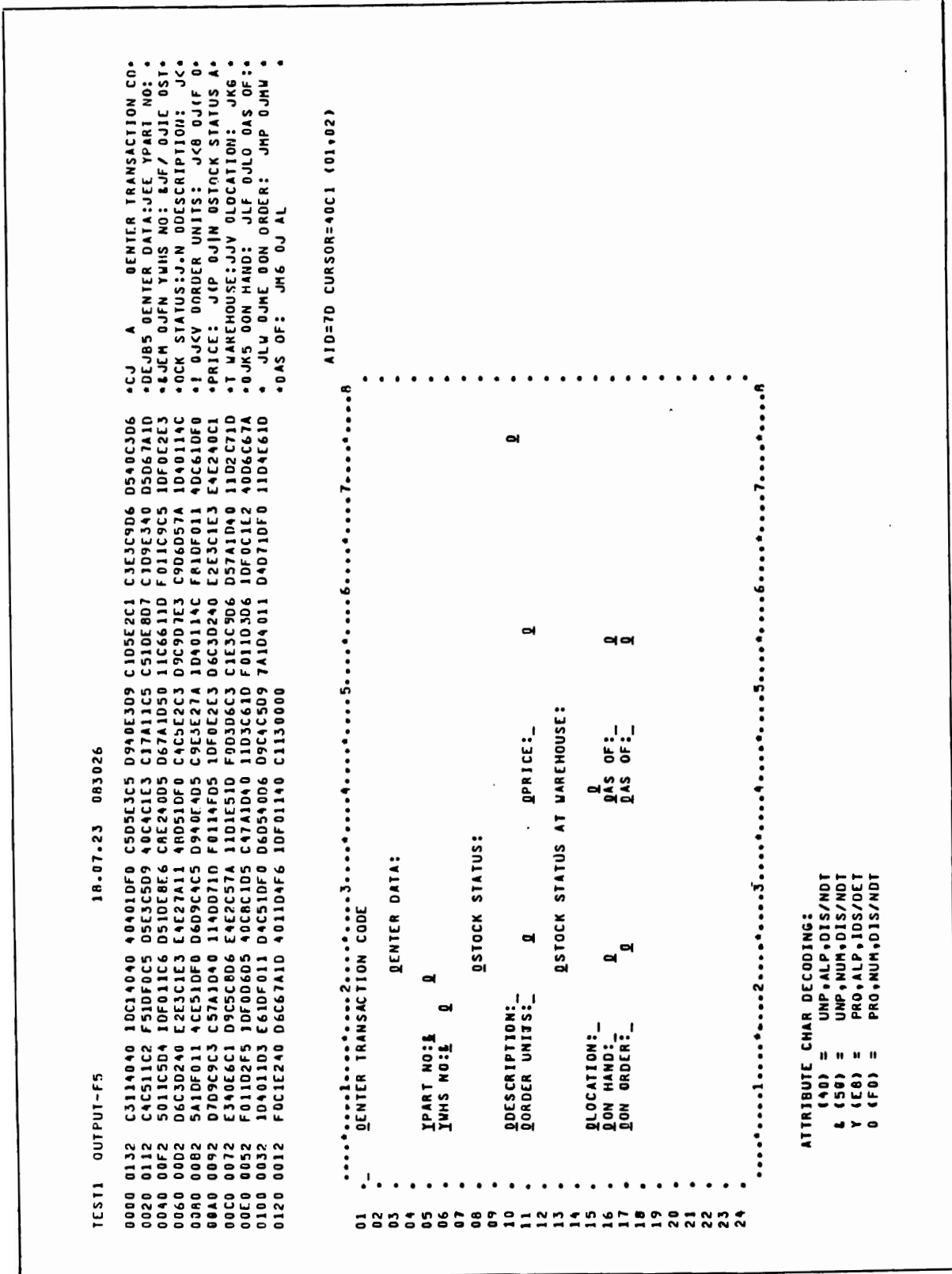


Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 4 of 22)

```

TEST1 INPUT                18.07.27 083026
0000 001R 70A04011 40C1D4EA D9D011C5 4FF3F2F3 F4F511C6 5EF2F0F0
                                ** J AMUROJE|12345JF:200
                                AID=7D CURSOR=4040 (01,01)
.....1.....2.....3.....4.....5.....6.....7.....8
*AMUROENTER TRANSACTION CODE
01 .....
02 .....
03 .....
04 .....
05 .....
06 .....
07 .....
08 .....
09 .....
10 .....
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .....
21 .....
22 .....
23 .....
24 .....

                                QENTER DATA:
                                QPART NO: J12345Q
                                QWHS NO: J200Q
                                QSTOCK STATUS:
                                QDESCRIPTION:
                                QORDER UNITS: Q
                                QPRICE: Q
                                QSTOCK STATUS AT WAREHOUSE:
                                QLOCATION: Q
                                QON HAND: Q
                                QON ORDER: Q
                                QAS OF: Q
                                QAS OF: Q

ATTRIBUTE CHAR DECODING:
(40) = UNP,ALP,DIS/MDT
A (C1) = UNP,ALP,DIS/MDT,MDT
J (01) = UNP,NUM,DIS/MDT,MDT
Y (E8) = PRO,ALP,IDS/DET
0 (F0) = PRO,NUM,DIS/MDT
    
```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 5 of 22)

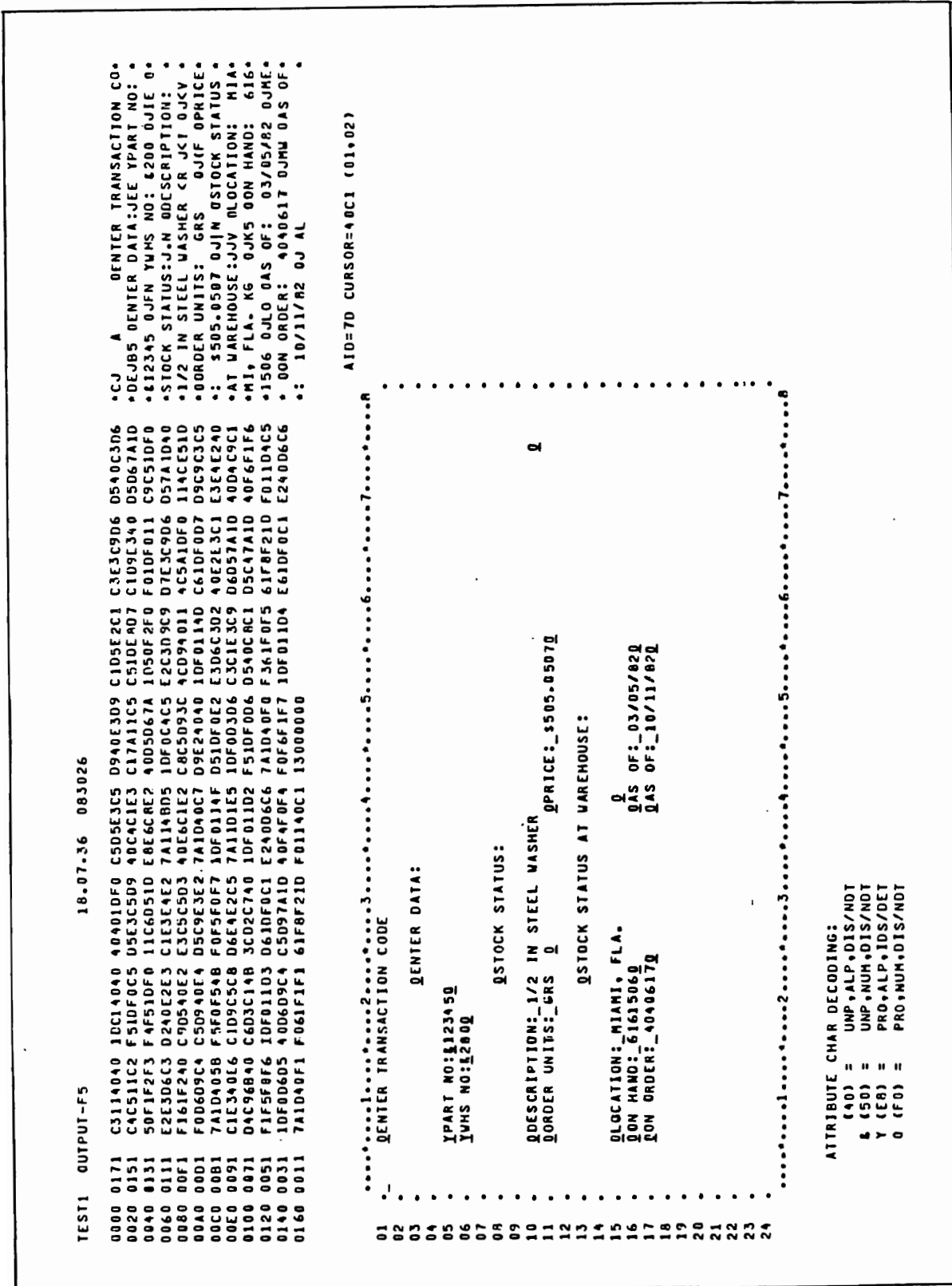


Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 6 of 22)

```

TEST1 INPUT                18.07.41 083026
0000 001A 7D40A011 40C104E4 09D811C5 4FF5F5F5 F5F511C6 5FF2F0F0
                                * * J AMURQJE |55555JF|200
                                AID=70 CURSOR=4040 (01,01)
.....1.....2.....3.....4.....5.....6.....7.....8
*AMURQENTER TRANSACTION CODE
.....9.....10.....11.....12.....13.....14.....15.....16.....17.....18.....19.....20.....21.....22.....23.....24
01 .....
02 .....
03 .....
04 .....
05 .....
06 .....
07 .....
08 .....
09 .....
10 .....
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .....
21 .....
22 .....
23 .....
24 .....

                                QENTER DATA:
                                QPART NO:J55555Q
                                QWHS NO:J200Q
                                QSTOCK STATUS:
                                QDESCRIPTION:_1/2 IN STEEL WASHER
                                QORDER UNITS:_GRS Q
                                QPRICE:_$505.0507Q
                                QSTOCK STATUS AT WAREHOUSE:
                                QLOCATION: MIAMI, FLA.
                                QON HAND:_6161506Q
                                QON ORDER:_4040617Q
                                QAS OF:_03/05/82Q
                                QAS OF:_10/11/82Q

ATTRIBUTE CHAR DECODING:
(40) = UNP,ALP,DIS/NOT
A (C1) = UNP,ALP,DIS/NDY,MDT
Y (C1) = UNP,NUM,DIS/NDY,MDT
Y (C8) = PRO,ALP,IDS/DET
0 (F0) = PRO,NUM,DIS/NOT

```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 7 of 22)

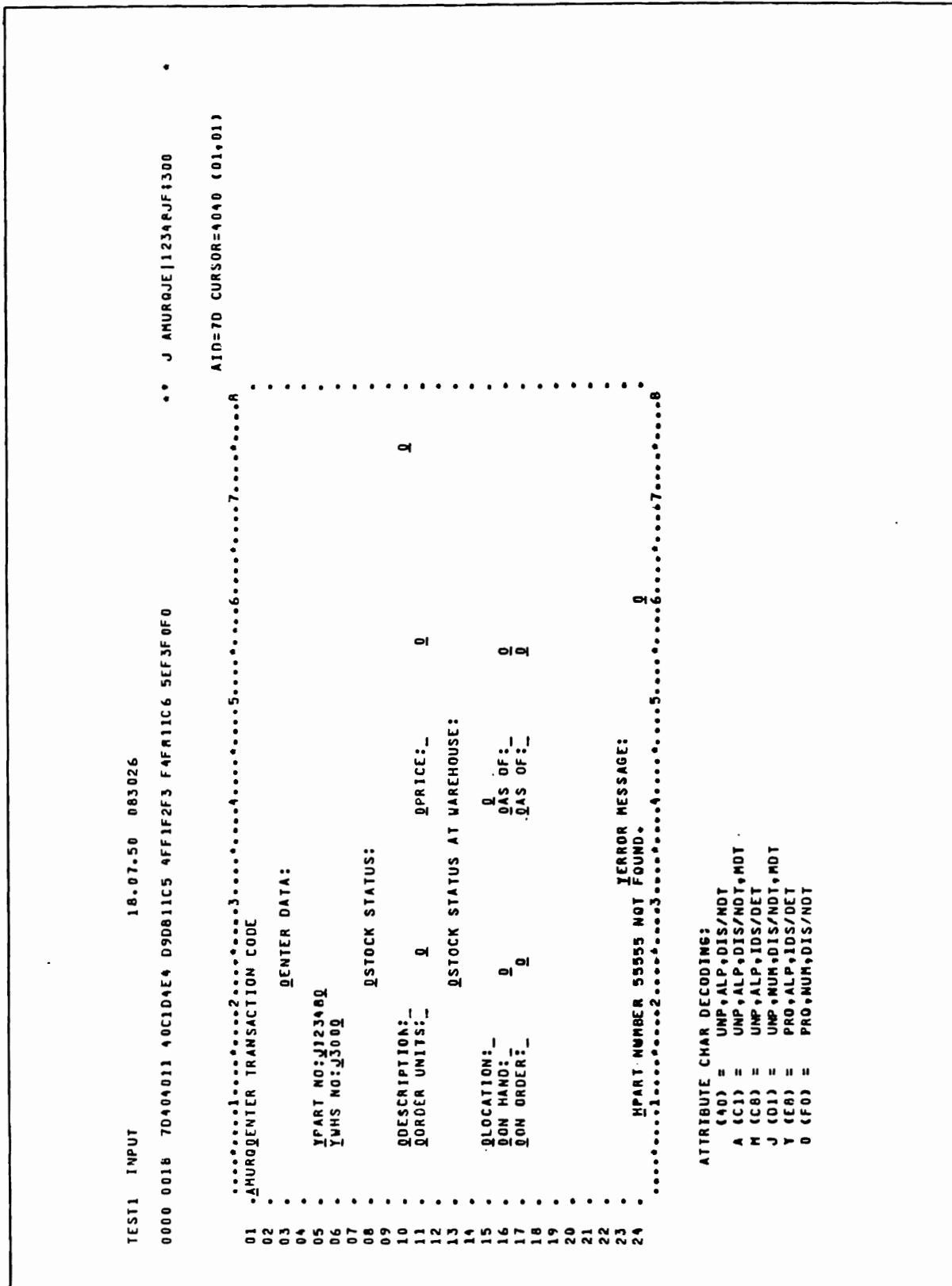


Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 9 of 22)

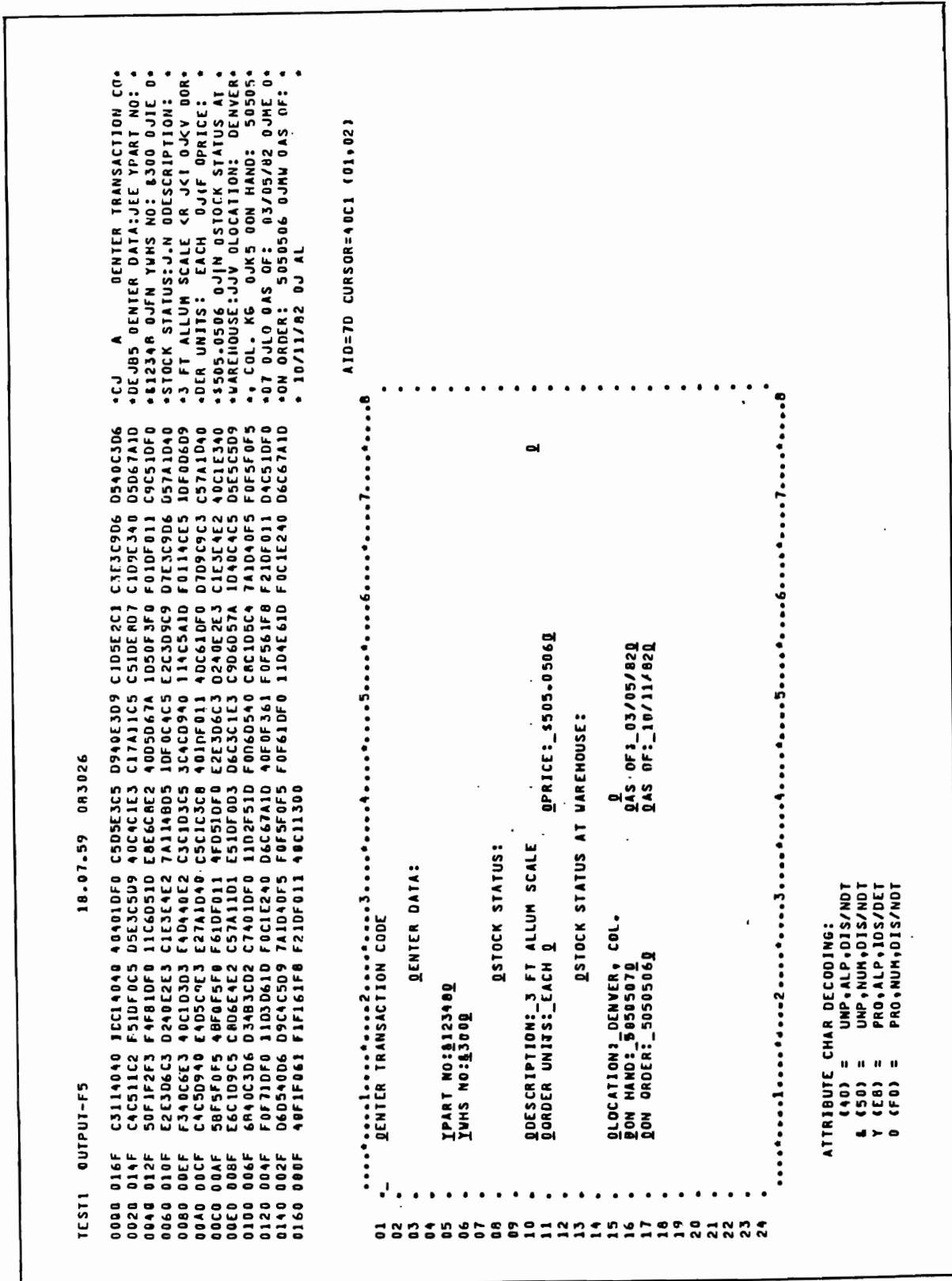


Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 10 of 22)

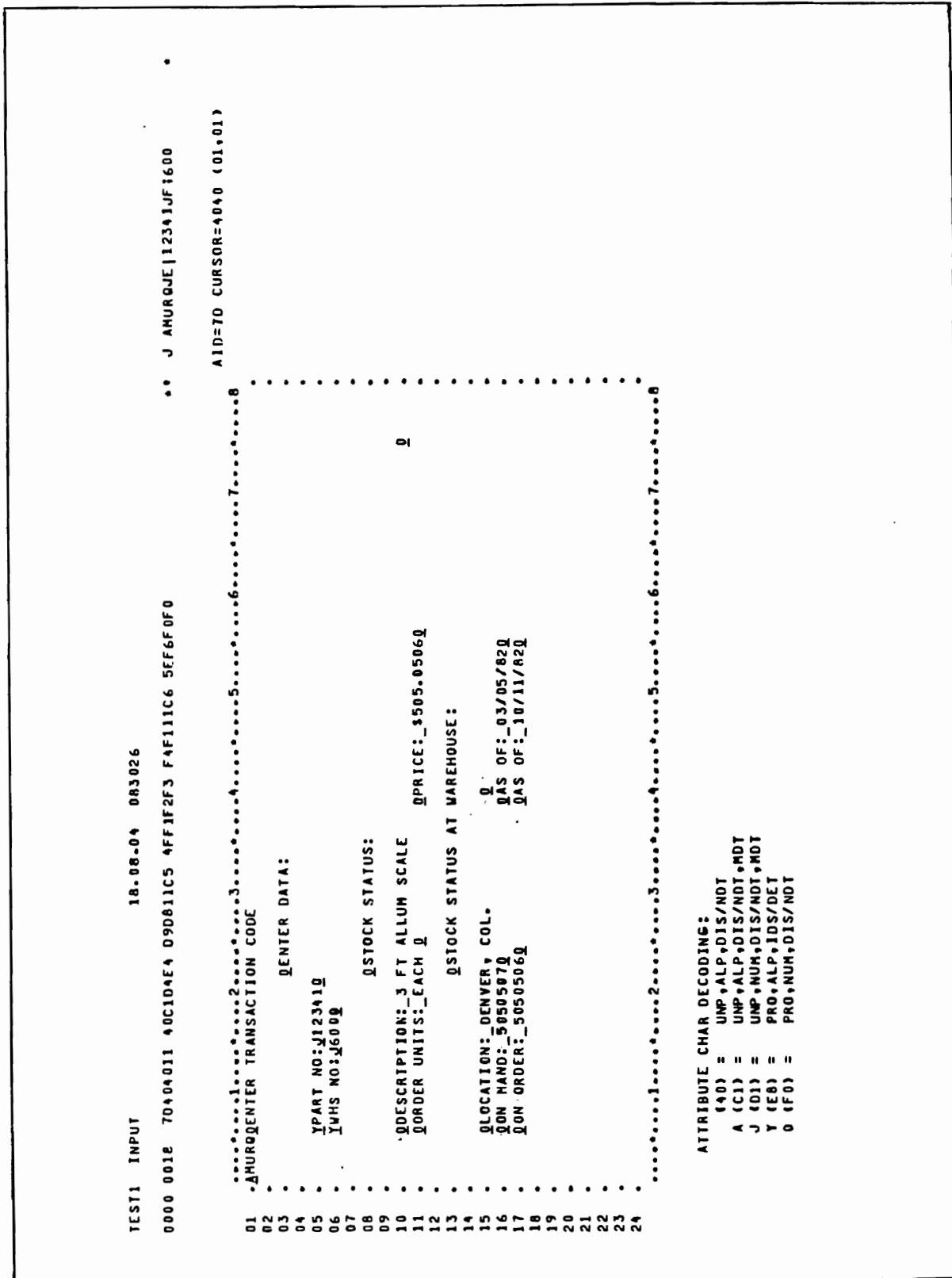


Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 11 of 22)

```

TEST1  OUTPUT-F1          14-08-09  0R3026
0000 004E C3114850 12585F11 5B7F10EA C5D9D9D6 0940D4C5 E2E2C1C7 C57A115C F81DC8D7 *CJ-8Ks-Js* YERROR MESSAGE:J*8 HP*
0020 002E C109E340 F1F2F3F4 F140D5D6 E340C6D6 E4D5C44D C9D540E6 C1D9C5C8 D6E4E2C5 *ART 12341 NOT FOUND IN WAREHOUSE*
0040 000E 40F6F0F0 3C5D6D40 1DF0115C F9130000 * 600 ), 0J*9L

```

AID=7D CURSOR=5CF9 (24,10)

```

.....1.....2.....3.....4.....5.....6.....7.....8
--_MURQENTER TRANSACTION CODE .....
01 .....
02 .....
03 .....
04 .....
05 .....
06 .....
07 .....
08 .....
09 .....
10 .....
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .....
21 .....
22 .....
23 .....
24 .....

```

QENTER DATA:

```

IPART NO:12341Q
YVHS NO:600Q

```

QSTOCK STATUS:

```

QDESCRIPTION:
QORDER UNITS: Q QPRICE: Q

```

QSTOCK STATUS AT WAREHOUSE:

```

QLOCATION: Q
QON HAND: Q QAS OF: Q
QON ORDER: Q QAS OF: Q

```

YERROR MESSAGE:

```

HPART 12341 NOT FOUND IN WAREHOUSE 600

```

ATTRIBUTE CHAR DECODING:
(40) = UNP,ALP,DIS/NOT
H (C8) = UNP,ALP,IDS/DET
& (50) = UNP,NUM,DIS/NOT
Y (E8) = PRO,ALP,IDS/DET
0 (F0) = PRO,NUM,DIS/NOT

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 12 of 22)


```

TEST1  OUTPUT-F1          10.08.19  083026
0000 0040 C3114850 12585F11 587F1DEA C5D7D9D6 094004C5 E2E2C1C7 C57A115C F81DC8C9 *CJ.EK3-JS" YERROR MESSAGE:J*8 HI*
0020 0020 D5E5C1D3 C9CA40C4 C1E3C17A 4007C1D9 E3D5D0640 DAE4E2E3 40C2C540 D5E4D4C5 *NVALID DATA: PARTNO MUST BE NUME*
0040 0080 D9C9C33C 5D6B401D F0115CF9 13000080 *RIC ), 0J*9L
                                AID=7D CURSOR=5CF9 (24,10)
01 .....1.....2.....3.....4.....5.....6.....7.....8.....
02 .....1.....2.....3.....4.....5.....6.....7.....8.....
03 .....1.....2.....3.....4.....5.....6.....7.....8.....
04 .....1.....2.....3.....4.....5.....6.....7.....8.....
05 .....1.....2.....3.....4.....5.....6.....7.....8.....
06 .....1.....2.....3.....4.....5.....6.....7.....8.....
07 .....1.....2.....3.....4.....5.....6.....7.....8.....
08 .....1.....2.....3.....4.....5.....6.....7.....8.....
09 .....1.....2.....3.....4.....5.....6.....7.....8.....
10 .....1.....2.....3.....4.....5.....6.....7.....8.....
11 .....1.....2.....3.....4.....5.....6.....7.....8.....
12 .....1.....2.....3.....4.....5.....6.....7.....8.....
13 .....1.....2.....3.....4.....5.....6.....7.....8.....
14 .....1.....2.....3.....4.....5.....6.....7.....8.....
15 .....1.....2.....3.....4.....5.....6.....7.....8.....
16 .....1.....2.....3.....4.....5.....6.....7.....8.....
17 .....1.....2.....3.....4.....5.....6.....7.....8.....
18 .....1.....2.....3.....4.....5.....6.....7.....8.....
19 .....1.....2.....3.....4.....5.....6.....7.....8.....
20 .....1.....2.....3.....4.....5.....6.....7.....8.....
21 .....1.....2.....3.....4.....5.....6.....7.....8.....
22 .....1.....2.....3.....4.....5.....6.....7.....8.....
23 .....1.....2.....3.....4.....5.....6.....7.....8.....
24 .....1.....2.....3.....4.....5.....6.....7.....8.....

QENTER DATA:
YPART NO:42345Q
YUHS NO:1200Q

QSTOCK STATUS:
QDESCRIPTION:
QORDER UNITS: Q QPRICE: Q
QLOCATION: Q QAS OF: Q Q
QON HAND: Q QAS OF: Q
QON ORDER: Q QAS OF: Q

QSTOCK STATUS AT WAREHOUSE:
YERROR MESSAGE:
MINVALID DATA: PARTNO MUST BE NUMERIC

ATTRIBUTE CHAR DECODING:
(40) = UNP,ALP,DIS/NOT
H (C8) = UNP,ALP,IDS/DET
& (50) = UNP,NUM,DIS/NOT
Y (E8) = PRO,ALP,IDS/DET
0 (F0) = PRO,NUM,DIS/NOT
    
```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 14 of 22)


```

TEST1  OUTPUT-F1          10.08.28  083026
0000 004C C3114850 12585F11 587F1DE8 C5D9D906 0940D4C5 E2E2C1C7 C57A115C F81DC8C9  *C-J,AKS-JS*  ERROR MESSAGE:J*8 HI*
0020 002C 05E5C1D3 C9C440C4 C1E3C17A 40E6C8E2 05D640D4 E4E2E340 C2C540D5 E4D4C5D9  *INVALID DATA: WHSNO MUST BE NUMER*
0040 000C C9C33C5D 68401DF0 115CF913                                     *IC ), 0J*9L
                                     A10=70 CURSOR=SCF9 (24,10)
.....1.....2.....3.....4.....5.....6.....7.....8
*_MURQENTER TRANSACTION CODE .....R
01 .....
02 .....
03 .....
04 .....
05 .....
06 .....
07 .....
08 .....
09 .....
10 .....
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .....
21 .....
22 .....
23 .....
24 .....

ENTER DATA:
IPART NO:123450
IWHM NO:18000

QSTOCK STATUS:
QDESCRIPTION:
QORDER UNITS:
QPRICE:
QSTOCK STATUS: AT WAREHOUSE:
QLOCATION:
QON HAND:
QON ORDER:
QAS OF:
QAS OF:

INVALID DATA: WHSNO MUST BE NUMERIC
ERROR MESSAGE:
ATTRIBUTE CHAR DECODING:
(40) = UNP,ALP,DIS/NOT
H (C8) = UNP,ALP,IOS/DET
B (50) = UNP,HUM,DIS/NOT
Y (E8) = PRO,ALP,IOS/DET
O (F0) = PRO,HUM,DIS/NOT

```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 16 of 22)

```

TEST1 INPUT                18.08.32 0A3026
0000 0018 7DA0A011 40C1D4E4 09D811C5 4FF1F2F3 F4E711C6 5EF2F0E8 *
                                ** J AMUROJE|1234XJF:20Y *
                                AID=7D CURSOR=4040 (01,01)
.....1.....2.....3.....4.....5.....6.....7.....8
*AMURQENTER TRANSACTION CODE .....8
01 .....
02 .....
03 .....
04 .....
05 .....
06 .....
07 .....
08 .....
09 .....
10 .....
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .....
21 .....
22 .....
23 .....
24 .....

QENTER DATA:
YPART NO:J1234XQ
YVHS NO:J20YD

QSTOCK STATUS:
QDESCRIPTION:
QORDER UNITS:
QPRICE:
QSTOCK STATUS AT WAREHOUSE:

QLOCATION:
QON HAND:
QON ORDER:
QAS OF:
QAS OF:

HINVALTD DATA: WMSNO MUST BE NUMERIC
.....1.....2.....3.....4.....5.....6.....7.....8

ATTRIBUTE CHAR DECODING:
(Q) = UNP,ALP,DIS/NDT
A (C) = UNP,ALP,DIS/NDT,MDT
H (CB) = UNP,ALP,IDS/DET
J (DI) = UNP,NUM,DIS/NDT,MDT
Y (EB) = PRO,ALP,IDS/DET
O (FO) = PRO,NUM,DIS/NDT
    
```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 17 of 22)

```

TEST1  OUTPUT-F1                18.08.58  083026

0000 0057 C3114850 12585F11 587F10E0 C5D9D9D6 D940DAC5 E2E2C1C7 C57A115C F81DC8C9 *CJ-&KS-JS" YERROR MESSAGE:J*8 MI*
0020 0037 D5E5C103 C9C440C4 C1E3C17A 4DD7C1D9 C3D5D640 CID5C440 E6C8E2D5 D640D4C4 *INVALID DATA: PARTNO AND WHSNO MU*
0040 0017 E2E340C2 C540D5E4 D4C5D9C9 C3404040 401DF011 5CF91300 *ST BE NUMERIC 0J*9L *
```

AID=7D CURSOR=SCF9 (24,10)

```

.....1.....2.....3.....4.....5.....6.....7.....8
01  _MURQENTER TRANSACTION CODE .....8
02  .
03  .
04  .
05  .
06  .
07  .
08  .
09  .
10  .
11  .
12  .
13  .
14  .
15  .
16  .
17  .
18  .
19  .
20  .
21  .
22  .
23  .
24  .
```

ENTER DATA:

```

IPART NO: 11234XQ
IWHM NO: 120YQ
```

QSTOCK STATUS:

```

QDESCRIPTION: Q
QORDER UNITS: Q QPRICE: Q
QLOCATION: Q
QON HAND: Q QAS OF: Q
QON ORDER: Q QAS OF: Q
```

QSTOCK STATUS AT WAREHOUSE:

```

.....1.....2.....3.....4.....5.....6.....7.....8
HINVALID DATA: PARTNO AND WHSNO MUST BE NUMERIC Q
.....1.....2.....3.....4.....5.....6.....7.....8
```

ERROR MESSAGE:

ATTRIBUTE CHAR DECODING:

```

(40) = UMP,ALP,DIS/NDT
H (C8) = UMP,ALP,IOS/DET
B (50) = UMP,NUM,DIS/NDT
Y (ER) = PRO,ALP,IOS/DET
0 (F0) = PRO,NUM,DIS/NDT
```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 18 of 22)


```

TEST1  OUTPUT-F1                18.08.47  083026
0000 0045 C3114850 12585F11 5B7F1DE8 C5D9D9D6 D940D4C5 E2E2C1C7 C57A113C F81DC8D7 *CJ.8KS-JS"  YERROR MESSAGE:J*8 MP*
0020 0025 C1D9E340 05E4D4C2 C5D9A0F1 F2F3F4F9 40U506E3 40C6D6E4 D5C44B3C 5D6B401D *ART NUMBER 12349 NOT FOUND. ), *
0040 0005 F0115CF9 13060000 *0J*9L

```

AID=7D CURSOR=5CF9 (24,10)

```

01 .....1.....2.....3.....4.....5.....6.....7.....8
02  _MURQENTER TRANSACTION CODE
03
04  QENTER DATA:
05  IPART NO:Q12349Q
06  IWH5 NO:Q100Q
07
08  QSTOCK STATUS:
09
10  QDESCRIPTION:
11  QORDER UNITS:  Q QPRICE:  Q
12
13  QSTOCK STATUS AT WAREHOUSE:
14
15  QLOCATION:
16  QON HAND:  Q QAS OF:  Q
17  QON ORDER:  Q QAS OF:  Q
18
19
20
21
22
23  MPART NUMBER 12349 NOT FOUND.
24  YERROR MESSAGE:

```

ATTRIBUTE CHAR DECODING:
(A0) = UNP,ALP,DIS/NOT
M (CB) = UNP,ALP,IDS/DET
& (50) = UNP,NUM,DIS/NOT
Y (E8) = PRO,ALP,IDS/DET
O (F0) = PRO,NUM,DIS/NOT

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 20 of 22)

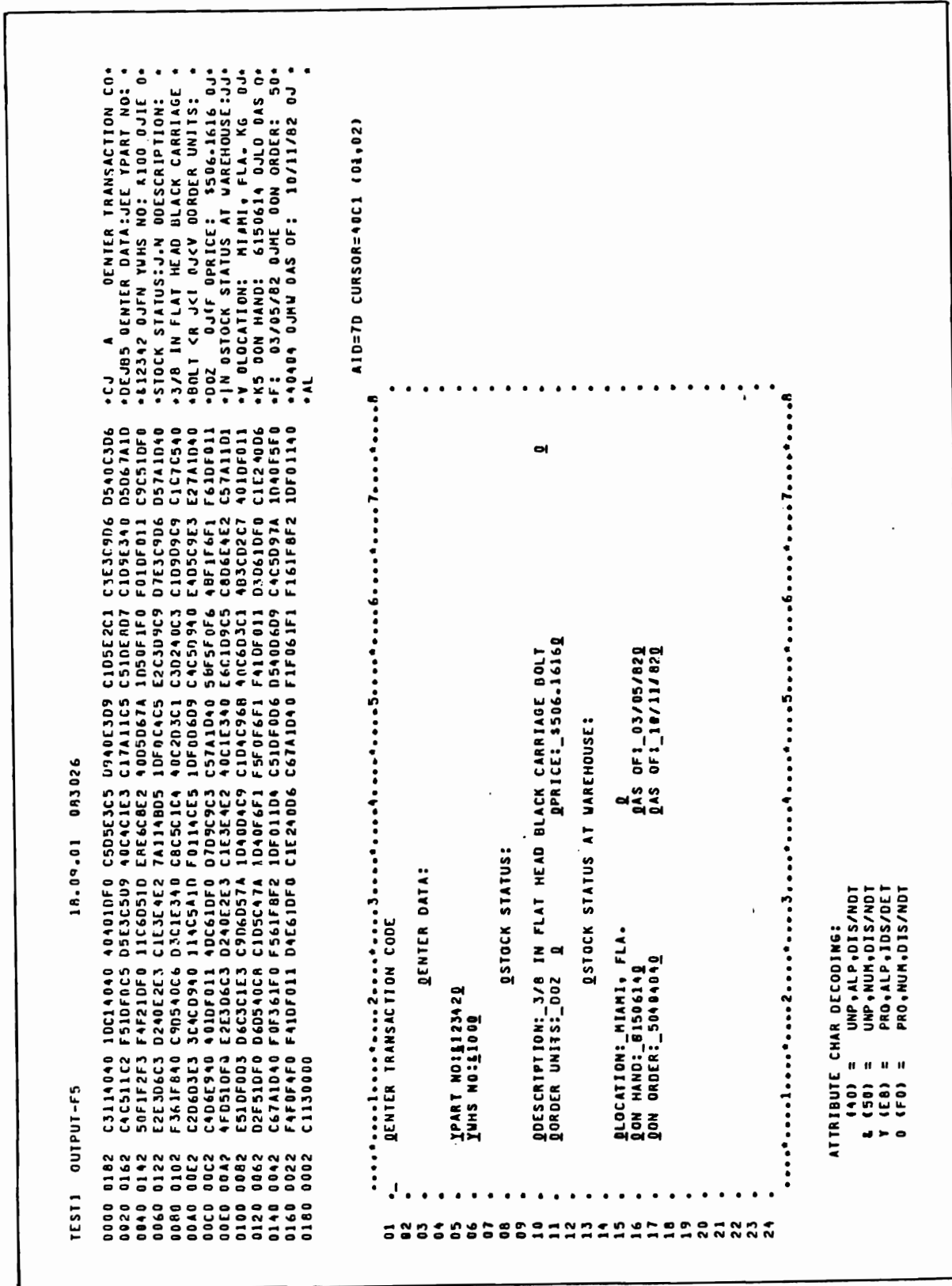


Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 22 of 22)

DATE	TIME	TYPE	18.24.20	****	INTERCOMM	LOG	DISPL	AY	****	PAGE					
MSGLN	THREAD	QPR	RSC	SSC	MMN	DATE	TIME	TID	PID	CO	USR	RMN	LOG	BLK	VMI
78	0	02	.. / 0000	.. / 0000	1	R3.026	18.06.5637	TOALL	0001	00	0	9F	00	00
000000	C9D5E3C5	D9C3D6DA	D4A0E2E3	C1D9E3E4		R3.026	18.06.5637	TOALL	0001	00	0	9F	00	00
000032	E5C8E7E7					R3.026	18.06.5637	TOALL	0001	00	0	9F	00	00
10A	0	02	.. / 0000	.. / 0000	1	R3.026	18.06.5637	TOALL	0001	00	0	01	00	50
000000	FF02002D	013C5C5C	5C40C7D6	06C440C5		R3.026	18.06.5637	TOALL	0001	00	0	01	00	50
000032	D9C3D6DA	D4A0C9E2	40D9C5C1	C4E8A07A		R3.026	18.06.5637	TOALL	0001	00	0	01	00	50
000064	F0FR					R3.026	18.06.5637	TOALL	0001	00	0	01	00	50
42	1	02	.. / 0000	.. / 0000	1	R3.026	18.06.5646	TOALL	0000	00	0	30	00	50
104	1	02	.. / 0000	.. / 0000	2	R3.026	18.06.5646	CNT01	0001	00	0	F2	00	50
000000	5C5C5C40	C7D6D6C4	40C5E5C5	05C9D5C7		R3.026	18.06.5646	CNT01	0001	00	0	F2	00	50
000032	C9E24009	C5C1C4E8	407A4040	40F0F160		R3.026	18.06.5646	CNT01	0001	00	0	F2	00	50
103	1	02	.. / 0000	.. / 0000	3	R3.026	18.06.5651	TEST1	0001	00	0	F2	00	50
000000	5C5C5C40	C7D6D6C4	40C5E5C5	05C9D5C7		R3.026	18.06.5651	TEST1	0001	00	0	F2	00	50
000032	C9E24009	C5C1C4E8	407A4040	40F0F160		R3.026	18.06.5651	TEST1	0001	00	0	F2	00	50
88	1	02	.. / 0000	.. / 0000	1	R3.026	18.06.5651	TOALL	0000	00	0	FA	00	50
000000	00000000	00000000	00000000	12780000		R3.026	18.06.5651	TOALL	0000	00	0	FA	00	50
000032	00000000	00000000	00000000	0000		R3.026	18.06.5651	TOALL	0000	00	0	FA	00	50
42	1	02	.. / 0000	.. / 0000	2	R3.026	18.06.5799	CNT01	0001	00	0	F3	00	50
42	1	02	.. / 0000	.. / 0000	3	R3.026	18.07.0160	TEST1	0001	00	0	F3	00	50
74	1	F2	.. / 0000	.. / 0000	5	R3.026	18.07.0610	TEST1	0001	00	1	01	00	57
000000	E2E3D9E3	40C3D6DA	D4C1D5C4	40C3D6DA		R3.026	18.07.0610	TEST1	0001	00	1	01	00	57
42	1	F2	.. / 0000	.. / 0000	5	R3.026	18.07.0610	TEST1	0000	00	1	30	00	57
74	1	02	.. / 0000	.. / 0000	6	R3.026	18.07.0610	TEST1	0001	00	1	F2	00	57
000000	E2E3D9E3	40C3D6DA	D4C1D5C4	40C3D6DA		R3.026	18.07.0610	TEST1	0001	00	1	F2	00	57
88	1	F2	.. / 0000	.. / 0000	5	R3.026	18.07.0610	TEST1	0000	00	1	FA	00	57
000000	00000000	00000000	00020000	07A80000		R3.026	18.07.0610	TEST1	0000	00	1	FA	00	57
000032	00000000	00000000	00000000	0000		R3.026	18.07.0610	TEST1	0000	00	1	FA	00	57
42	1	02	.. / 0000	.. / 0000	6	R3.026	18.07.1080	TEST1	0001	00	1	F3	00	57
67	0	F2	.. / 0000	.. / 0000	7	R3.026	18.07.1515	TEST1	0001	00	2	01	00	FF
000000	D4D4E4C3	68E2C8D6	E6B8DE2	E3D2E2E3		R3.026	18.07.1515	TEST1	0001	00	2	01	00	FF
42	1	F2	.. / 0000	.. / 0000	7	R3.026	18.07.1515	TEST1	0000	00	2	30	00	FF
350	1	02	.. / 0000	.. / 0000	8	R3.026	18.07.1542	TEST1	0000	00	2	F2	00	67
000000	F5C31140	401DC140	4040401D	F8C5D5E3		R3.026	18.07.1542	TEST1	0000	00	2	F2	00	67
000032	D6C4C511	C2F510F0	C5D5E3C5	D940C4C1		R3.026	18.07.1542	TEST1	0000	00	2	F2	00	67
000064	1D5011C5	D41DF011	C6D51DE8	E6C8E240		R3.026	18.07.1542	TEST1	0000	00	2	F2	00	67
000096	E3D6C3D2	40E2E3C1	E3E4E27A	1148D51D		R3.026	18.07.1542	TEST1	0000	00	2	F2	00	67

Figure 55. Simulation Mode Execution Log Printout (Page 1 of 6)

**** I N T E R C O M H L O G D I S P L A Y ****											PAGE				
DATE	TIME	TYP	RSC	SSC	MMN	DATE	TIME	TID	PID	CO	USR	RMN	LOG	BLK	VMI
42	1	02	0000	0000	12	83-026	18-07-1663	TEST1	0000	00	0000	00	4	F3	00 67
62	0	F2	0000	0000	13	83-026	18-07-5101	TEST1	0001	C6	0000	00	5	01	00 FF
000000	04E4D9D8	6811C54F	F1F2F3F4	F111C65E	F3F0F026										
42	1	F2	0000	0000	13	83-026	18-07-5101	TEST1	0000	00	0000	00	5	30	00 FF
411	1	02	0000	0000	14	83-026	18-07-5122	TEST1	0000	00	0000	00	5	F2	00 67
000000	F5C31140	401DC140	000000	000000	C5D940E3	09C1D5E2	C1C3E3C9	06D540C3	0000	00	0000	00	5	F2	00 67
000032	06C4C511	C2F51D0F	C5D5E3C5	D940C4C1	E3C37A11	C5C51D0E	07C1D9E3	40D5067A	0000	00	0000	00	5	F2	00 67
000064	1050F1F2	F3F4F810	F011C6D5	10E8E6C8	E2400506	7A1D50F3	F0F010F0	11C9C51D	0000	00	0000	00	5	F2	00 67
000096	F0E2E306	C3D240E2	E3C1E3E4	E27A1148	0510F0C4	C5E2C3D9	C907E3C9	06D57A1D	0000	00	0000	00	5	F2	00 67
000128	40F340C6	E340C1D3	03E4D490	E2C3C1D3	C53C4CD9	40114C5A	10F0114C	E510F006	0000	00	0000	00	5	F2	00 67
000160	09C4C5D9	40E4D5C9	E3E27A10	40C5C1C3	C8A01D0F	114DC61D	F007D9C9	C3C57A1D	0000	00	0000	00	5	F2	00 67
000192	4058F5F0	F548F0F5	F0F610F0	114F051D	F0E2E3D6	C3D240E2	E3C1E3E4	E240C1E3	0000	00	0000	00	5	F2	00 67
000224	40E6C109	C5C8D6E4	E2C57A11	D1E51D0F	D3D6C3C1	E3C9D6D5	7A1D40C4	C5D5E3C5	0000	00	0000	00	5	F2	00 67
000256	096840C3	D6D3483C	D2C7401D	F011D2F5	10F006D5	40C8C1D5	C47A1D40	F5F0F5F0	0000	00	0000	00	5	F2	00 67
000288	F5F0F71D	F01103D6	10F0C1E2	40D6C67A	10A0F0F3	61F0F561	F8F210F0	11D4C51D	0000	00	0000	00	5	F2	00 67
000320	F0D6D540	D6D9C4C5	D97A1D40	F5F0F5F0	F5F0F61D	F01104E6	10F0C1E2	40D6C67A	0000	00	0000	00	5	F2	00 67
000352	1D40F1F0	61F1F161	F8F210F0	1140C113	03				0000	00	0000	00	5	F2	00 67
88	1	F2	0000	0000	13	83-026	18-07-5122	TEST1	0000	00	0000	00	5	FA	00 FF
000000	00000000	00010000	00110000	19F00000	00630000	00010000	00050000	00000000	0000	00	0000	00	5	FA	00 FF
000032	00000000	00000000	00000000	0000											
42	1	02	0000	0000	14	83-026	18-08-0012	TEST1	0000	00	0000	00	5	F3	00 67
62	0	F2	0000	0000	15	83-026	18-08-0454	TEST1	0001	C6	0000	00	6	01	00 FF
000000	D4E4D9D8	6811C54F	F1F2F3F4	F111C65E	F6F0F026										
42	1	F2	0000	0000	15	83-026	18-08-0454	TEST1	0000	00	0000	00	6	30	00 FF
122	1	02	0000	0000	16	83-026	18-08-0475	TEST1	0000	00	0000	00	6	F2	00 67
000000	F1C31140	5012585F	11587F1D	E8C5D9D9	D6D9A0D4	C5E2E2C1	C7C57A11	5CF81D0C	0000	00	0000	00	6	F2	00 67
000032	D7C1D9E3	40F1F2F3	F4F140D5	D6E340C6	D6F4D5C4	40C9D540	E6C1D9C5	CRD6E4E2	0000	00	0000	00	6	F2	00 67
000064	C540F8F0	F03C5D68	401D0F11	5CF913D3					0000	00	0000	00	6	F2	00 67
88	1	F2	0000	0000	15	83-026	18-08-0475	TEST1	0000	00	0000	00	6	FA	00 FF
000000	00000000	00010000	000E0000	1A500000	004D0000	00010000	00050000	00000000	0000	00	0000	00	6	FA	00 FF
000032	00000000	00000000	00000000	0000											
42	1	02	0000	0000	16	83-026	18-08-0903	TEST1	0000	00	0000	00	6	F3	00 67
62	0	F2	0000	0000	17	83-026	18-08-1413	TEST1	0001	C6	0000	00	7	01	00 FF
000000	D4E4D9D8	6811C54F	F1F2F3F4	F511C65E	F2F0F026										
42	1	F2	0000	0000	17	83-026	18-08-1414	TEST1	0000	00	0000	00	7	30	00 FF
121	1	02	0000	0000	18	83-026	18-08-1414	TEST1	0000	00	0000	00	7	F2	00 67
000000	F1C31140	5012585F	11587F1D	E8C5D9D9	D6D9A0D4	C5E2E2C1	C7C57A11	5CF81D0C	0000	00	0000	00	7	F2	00 67
MSGLEN	THREAD	QPR	RSC	SSC	MMN	DATE	TIME	TID	PID	CO	USR	BMN	LOG	BLK	VMI

Figure 55. Simulation Mode Execution Log Printout (Page 3 of 6)


```

DATE 83.026   TIME 18.24.20   ***** I N T E R C O M M   L O G   D I S P L A Y *****   PAGE   6
MSGLEN  THREAD  GPR  RSC      SSC      MMN      DATE      TIME      TIO  PID  CO  USR  BMN  LOG  BLK  VMI
-----
103  1  02  ../0000  .U/00E4  30  83.026  18.09.0668  TEST1  ..... 0001  00  0  F2  00  50
000000  5C5C5C40 C706D6C4 40C5E5C5 05C9D5C7 405C5C5C 4040C9D5 E3C5D9C3 D6D4D440 ..... 600D  EVENING  .... INTERCOMM *
000032  C9E240C3 D3D6E2C5 C47A4040 40F0F160 F2F660FB F34040F1 F840F0F9 37 ..... IS  CLOSED:  01-26-83  18.09.
-----
88  1  02  .U/00E4  ../0000  28  83.026  18.09.0668  TOALL  ..... 0000  00  0  FA  00  50
000000  00000000 00000000 00080000 12780000 00240000 00020000 00020000 00000000 .....
000032  00000000 00000000 00000000 00000000 00000000 .....
-----
42  1  02  ../0000  .U/00E4  29  83.026  18.09.0786  CNT01  ..... 0001  00  0  F3  00  50
-----
47  0  F2  .J/0001  ../0000  31  83.026  18.09.1056  CNT01  ..... 0001  00  12  01  00  FC
000000  05D9C3C4 26 ..... NRCD.
-----
42  1  F2  .J/0001  ../0000  31  83.026  18.09.1056  CNT01  ..... 0000  00  12  30  00  FC
-----
42  1  02  ../0000  .U/00E4  30  83.026  18.09.1503  TEST1  ..... 0001  00  0  F3  00  50
-----
78  0  00  ../0000  ../0000  0  83.026  18.09.5060  ..... 0000  00  0  AA  00  00
000000  C9D5E3C5 D9C3D6D4 D440C3D3 06E2C5C4 06E6D540 D4C5E2E2 C1C7C940 E6E2C4C3 ..... INTERCOMM  CLOSEDOWN  MESSAGE  WSDC *
000032  ESCRE7E7 ..... VHXX
-----
    
```

Figure 55. Simulation Mode Execution Log Printout (Page 6 of 6)

Chapter 12

SUBSYSTEM TESTING IN TEST MODE

12.1 INTRODUCTION

All of the testing functions may be performed using the Intercomm Test Mode of operation without a Front End defined. Rather than receiving messages from a terminal, the Test Monitor reads messages into the system from a card-image data set. Snaps of input (snap id=15) and output (snap id=20) messages constitute a history of Test Mode execution. Essentially, the Front End is replaced by the Test Monitor (PMITEST) to drive the Back End as usual. In this way, subsystem testing can be going on in one or more regions or address spaces without affecting the on-line system. Figure 56 illustrates a sample reentrant COBOL subsystem (SQCOBOL) designed for the same purpose as SQCOBOLA, but using the Edit, Output and Change/Display Utilities. Note that SQCOBOL was compiled using the Release 9 version of COPY members (see Appendix B).

12.2 TESTING A SUBSYSTEM IN TEST MODE

To add and test an application subsystem in Test Mode, do the following:

NOTE: Steps preceded by an asterisk (*) may often be performed for the application programmer by an installation's Intercomm System Manager. Appendix C summarizes the Intercomm Table entries.

1. Compile and linkedit the application program. Appendix A describes Intercomm-supplied COBOL JCL procedures.
- *2. Create or add to a USRSCTS member on a user test library to contain a Subsystem Control Table Entry (SYCTTBL macro) which describe the subsystem. Reassemble and link INTSCT which copies the USRSCTS member from the test library (see Figure 57).
- *3. Create or add to a USRVERBS member on the user test library to contain an Edit Control Table (VERBTBL) entry for editing of input test messages by the Edit Utility. Reassemble and link PMIVERBS which copies the USRVERBS member from the test library (see Figure 57).
- *4. If a Fixed Format output message (VMI=X'72') is created for processing by the Change/Display Utility, code an entry for the CHNGTB (see Figure 57) to define the DES000 data set entry number for the File Description Record (DES00001--see Figure 58). The PMIEXLD utility must be used to load the FDR to the DES000 file (see the Utilities Users Guide and the Operating Reference Manual).

5. Code, assemble and link and add an INCLUDE statement for the OFT load module RPTnnnnn (RPT00100 and RPT00501--see Figure 58) to the Output Format Table (PMIRCNTB) in the Test Mode Intercomm linkedit for output message formatting by the Output Utility.
6. Prepare test messages via the SIMCRTA utility or as direct card-image input data (SYSIN data set). An input test message consists of a header card, detail cards, and a trailer card, grouped together as illustrated in Figure 60. Figure 59 details the required card formats. The message area in the Test Monitor will accommodate a message text up to 958 bytes long. Longer messages would require a modification to the Test Monitor (PMITEST), as described in the Operating Reference Manual.
- *7. Add control cards to the linkedit deck for the user program, unless the subsystem is dynamically loadable (see Figure 61).
- *8. Linkedit to create an Intercomm Test Mode load module (see Figure 61).
9. Create test data sets and add DD statements for them to the execution JCL.
10. Execute in Test Mode with test messages in card-image format:
 - a. Single-thread test the subsystem; to test a reentrant subsystem, initially specify MNCL=1 in the subsystem's SYCTTBL macro.
 - b. Multithread test a reentrant subsystem (change MNCL) using several test messages.

Test Mode execution is activated by the parameter 'TEST' on the Intercomm EXEC statement. Figure 61 illustrates a sample execution deck with test message input (DD statement SYSIN) for the sample inquiry program and JCL to print the system log.

The resulting snaps for the test mode execution of the sample inquiry subsystem are illustrated in Figure 62.

The System Log printed after executing in Test Mode with the sample inquiry subsystem is shown in Figure 63.

11. Test the subsystem concurrently with other application subsystems.

Note: to implement the sample subsystem for on-line execution, it would be necessary to code a BTVERB macro (in USRBTVRB--see Chapter 11) as follows:

```
BTVERB VERB=RTRQ,SSCH=R,SSC=Q,CONV=18000,EDIT=YES
```

```

PP 5734-CB2 V4 RELEASE 1.5 10NOV77          IBM OS AMERICAN NATIONAL STANDARD COBOL

1

00001 000100 IDENTIFICATION DIVISION.
00002 000200 PROGRAM-ID. SQC0BCL.
00003 000300 REMARKS. SUBSYSTEM CODE IS 'RQ', VERB IS 'RTRQ'.
00004 000400 ENVIRONMENT DIVISION.
00005 000500 DATA DIVISION.
00006 000600 WORKING-STORAGE SECTION.
00007 000700 77 FIXED-FORMAT-NAME PIC X(12) VALUE 'SSR000010 *'.
00008 000800 77 SLASH PIC X VALUE '/'.
00009 000900 77 BDAM-READ-VALUE PIC X VALUE 'D'.
00010 001000 77 DD-PART PIC X(8) VALUE 'PARTFILE'.
00011 001100 77 DD-STOCK PIC X(P) VALUE 'STOKFILE'.
00012 001200 01 ERROR-FORMAT.
00013 001300 05 FILLER PIC X(7) VALUE '255003N'.
00014 001400 05 FILLER PIC S9(3) VALUE '+501 COMP'.
00015 001500 05 FILLER PIC X(7) VALUE '249051N'.
00016 001600 01 MESSAGE-TABLE.
00017 001700 04 MSG-A PIC X(12) VALUE 'PART NUMBER *'.
00018 001800 04 MSG-B PIC X(11) VALUE 'NOT FOUND.'.
00019 001900 04 MSG-C PIC X(5) VALUE 'PART *'.
00020 002000 04 MSG-D PIC X(24) VALUE 'NOT FOUND IN WAREHOUSE *'.
00021 002100 04 MSG-E PIC X(20) VALUE 'MESSAGE CANCELLED.'.
00022 002200 01 HEXCODES COPY ICOMHEX.
00023 C 01 HEXCODES.
00024 C 05 HEX-00 PIC X VALUE '*'.
00025 C 05 CODE-00 REDEFINES HEX-00 PIC X.
00026 C 05 HEX-15 PIC X VALUE 'N'.
00027 C 05 CODE-21 REDEFINES HEX-15 PIC X.
00028 C 05 HEX-37 PIC X VALUE '*'.
00029 C 05 CODE-55 REDEFINES HEX-37 PIC X.
00030 C 05 HEY-50 PIC X VALUE '%'.
00031 C 05 CODE-60 REDEFINES HEX-50 PIC X.
00032 C 05 HEX-51 PIC X VALUE 'J'.
00033 C 05 CODE-91 REDEFINES HEX-51 PIC X.
00034 C 05 HEX-52 PIC X VALUE 'K'.
00035 C 05 CODE-82 REDEFINES HEX-52 PIC X.
00036 C 05 HEX-53 PIC X VALUE 'L'.
00037 C 05 CODE-83 REDEFINES HEY-53 PIC X.
00038 C 05 HEX-54 PIC X VALUE 'M'.
00039 C 05 CODE-84 REDEFINES HEX-54 PIC X.
00040 C 05 HEX-55 PIC X VALUE 'N'.
00041 C 05 CODE-85 REDEFINES HEX-55 PIC X.
00042 C 05 HEX-56 PIC X VALUE 'O'.
00043 C 05 CODE-86 REDEFINES HEX-56 PIC X.
00044 C 05 HEX-57 PIC X VALUE 'P'.
00045 C 05 CODE-87 REDEFINES HEX-57 PIC X.
00046 C 05 HEX-72 PIC X VALUE '2'.
00047 C 05 CODE-114 REDEFINES HEX-72 PIC X.
00048 C 05 HEX-FF PIC X VALUE '*'.
00049 C 05 CODE-255 REDEFINES HEX-FF PIC X.
00000100
00000200
00000300
00000400
00000500
00000600
00000700
00000800
00000900
00001000
00001100
00001200
00001300
00001400
00001500
00001600
00001700
00001800
00001900
00002000
00002100
00002200
00002300
00002400
00002500
00002600
00002700
00002800
00002900
00003000
00003100
00003200
00003300
00003400
00003500
00003600
00003700
00003800
00003900
00004000
00004100
00004200
00004300
00004400
00004500
00004600
00004700
00004800
00004900
00005000
00005100
00005200
00005300
00005400
00005500
00005600
00005700
00005800
00005900
00006000
00006100
00006200
00006300
00006400
00006500
00006600
00006700
00006800
00006900
00007000
00007100
00007200
00007300
00007400
00007500
00007600
00007700
00007800
00007900
00008000
00008100
00008200
00008300
00008400
00008500
00008600
00008700
00008800
00008900
00009000
00009100
00009200
00009300
00009400
00009500
00009600
00009700
00009800
00009900
00010000
00010100
00010200
00010300
00010400
00010500
00010600
00010700
00010800
00010900
00011000
00011100
00011200
00011300
00011400
00011500
00011600
00011700
00011800
00011900
00012000
00012100
00012200
00012300
00012400
00012500
00012600
00012700
00012800
00012900
00013000
00013100
00013200
00013300
00013400
00013500
00013600
00013700
00013800
00013900
00014000
00014100
00014200
00014300
00014400
00014500
00014600
00014700
00014800
00014900
00015000
00015100
00015200
00015300
00015400
00015500
00015600
00015700
00015800
00015900
00016000
00016100
00016200
00016300
00016400
00016500
00016600
00016700
00016800
00016900
00017000
00017100
00017200
00017300
00017400
00017500
00017600
00017700
00017800
00017900
00018000
00018100
00018200
00018300
00018400
00018500
00018600
00018700
00018800
00018900
00019000
00019100
00019200
00019300
00019400
00019500
00019600
00019700
00019800
00019900
00020000
00020100
00020200
00020300
00020400
00020500
00020600
00020700
00020800
00020900
00021000
00021100
00021200
00021300
00021400
00021500
00021600
00021700
00021800
00021900
00022000
00022100
00022200
00022300
00022400
00022500
00022600
00022700
00022800
00022900
00023000
00023100
00023200
00023300
00023400
00023500
00023600
00023700
00023800
00023900
00024000
00024100
00024200
00024300
00024400
00024500
00024600
00024700
00024800
00024900
00025000
00025100
00025200
00025300
00025400
00025500

```

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 1 of 11)

00051	002300 01	COBREENT-CODES COPY ICOMSBS.	
00052 C	01	COBREENT-CODES.	00001000
00053 C	*	THESE CODES REPRESENT OFFSETS FOR ROUTINE ADDRESSES IN THE	00002000
00054 C	*	TABLE NAMED REENTSDS. ONLY THE MOST COMMONLY USED VALUES	00003000
00055 C	*	ARE INCLUDED HERE; THE USERS MANUAL HAS A COMPLETE LIST.	00004000
00056 C	*	IF OFFSET ODD, THEN TRUE OFFSET=-(OFFSET+1)	00005000
00057 C	05	MAPFREE PIC 99 COMP VALUE 91.	00005500
00058 C	05	FECMRLSE PIC 99 COMP VALUE 87.	00006000
00059 C	05	FECEND PIC 99 COMP VALUE 83.	00007000
00060 C	05	FECENDC PIC 99 COMP VALUE 79.	00008000
00061 C	05	DYN-ALLOCATE PIC 99 COMP VALUE 75.	00009000
00062 C	05	DYN-ACCESS PIC 99 COMP VALUE 71.	00010000
00063 C	05	MAPURGE PIC 99 COMP VALUE 67.	00011000
00064 C	05	MAPCLR PIC 99 COMP VALUE 63.	00012000
00065 C	05	MAPEND PIC 99 COMP VALUE 59.	00013000
00066 C	05	MAPOUT PIC 99 COMP VALUE 55.	00014000
00067 C	05	MAPIN PIC 99 COMP VALUE 51.	00015000
00068 C	05	INTUNSTO PIC 99 COMP VALUE 47.	00016000
00069 C	05	INTSTORE PIC 99 COMP VALUE 43.	00017000
00070 C	05	INTFETCH PIC 99 COMP VALUE 39.	00018000
00071 C	05	FECMFCBK PIC 99 COMP VALUE 35.	00019000
00072 C	05	FECMGDD PIC 99 COMP VALUE 31.	00020000
00073 C	05	DQ-WRITE PIC 99 COMP VALUE 27.	00021000
00074 C	05	DQ-READX PIC 99 COMP VALUE 23.	00022000
00075 C	05	DQ-WRITE PIC 99 COMP VALUE 19.	00023000
00076 C	05	DQ-READ PIC 99 COMP VALUE 15.	00024000
00077 C	05	DQ-CLOSE PIC 99 COMP VALUE 11.	00025000
00078 C	05	DQ-OPEN PIC 99 COMP VALUE 07.	00026000
00079 C	05	DQ-BUILD PIC 99 COMP VALUE 03.	00027000
00080 C	05	FH-SELECT PIC 99 COMP VALUE 4.	00028000
00081 C	05	FH-RELEASE PIC 99 COMP VALUE 8.	00029000
00082 C	05	FH-READ PIC 99 COMP VALUE 12.	00030000
00083 C	05	FH-WRITE PIC 99 COMP VALUE 16.	00031000
00084 C	05	FH-GET PIC 99 COMP VALUE 20.	00032000
00085 C	05	FH-PUT PIC 99 COMP VALUE 24.	00033000
00086 C	05	FH-RELEX PIC 99 COMP VALUE 28.	00034000
00087 C	05	FH-FEOV PIC 99 COMP VALUE 32.	00035000
00088 C	05	COBPUT PIC 99 COMP VALUE 68.	00036000
00089 C	05	MSGCOL PIC 99 COMP VALUE 72.	00037000
00090 C	05	COBSTORF PIC 99 COMP VALUE 76.	00038000
00091 C	05	CONVERSE PIC 99 COMP VALUE 80.	00039000
00092 C	05	DBINT PIC 99 COMP VALUE 84.	00040000
00093 C	05	LOGPUT PIC 99 COMP VALUE 88.	00041000
00094 C	05	PAGE-FILE PIC 99 COMP VALUE 92.	00042000
00095 C	05	FH-GETV PIC 99 COMP VALUE 96.	00043000
00096 C	05	FH-PUTV PIC 999 COMP VALUE 100.	00044000
00097 C	*	CODES 104 AND UP INDICATE USER ADDITIONS TO THE TABLE	00045000
00099	002400 01	FILLER PIC X(22) VALUE 'END OF WORKING STORAGE'.	

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 2 of 11)

```

3

00100 002500 LINKAGE SECTION.
00101 002600 01 INPUT-MESSAGE COPY ICOMINMG.
00102 C      01 INPUT-MESSAGE.                                00000100
00103 C      04 MESSG-HDR.                                    00000200
00104 C      06 MSGH-LENGTH PIC S9999 COMP.                 00000300
00105 C      06 MSGH-QPR PIC X.                             00000400
00106 C      06 MSGH-RSCH PIC X.                             00000500
00107 C      06 MSGH-RSC PIC X.                             00000600
00108 C      06 MSGH-SSC PIC X.                             00000700
00109 C      06 MSGH-MMN PIC XXX.                           00000800
00110 C      06 MSGH-DATE.                                    00000900
00111 C      08 MSGH-YR PIC 99.                              00001000
00112 C      08 MSGH-PERIOD PIC X.                          00001100
00113 C      08 MSGH-JULIAN-DAY PIC 999.                   00001200
00114 C      06 MSGH-TIME.                                    00001300
00115 C      08 MSGH-HH PIC 99.                              00001400
00116 C      08 MSGH-MM PIC 99.                              00001500
00117 C      08 MSGH-SS PIC 99.                              00001600
00118 C      08 MSGH-TM PIC 99.                              00001700
00119 C      06 MSGH-TID.                                    00001800
00120 C      08 MSGH-TI1 PIC X.                              00001900
00121 C      08 MSGH-TI2-3 PIC XX.                           00002000
00122 C      08 MSGH-TI4-5 PIC 99.                           00002100
00123 C      06 MSGH-CON PIC S9999 COMP.                    00002200
00124 C      06 MSGH-PID PIC X(5).                           00002300
00125 C      06 MSGH-SSCH PIC X.                             00002400
00126 C      06 MSGH-USR PIC X.                              00002500
00127 C      06 MSGH-PMN PIC XX.                             00002600
00128 C      06 MSGH-LGG PIC X.                              00002700
00129 C      06 MSGH-RLK PIC Y.                              00002750
00130 C      06 MSGH-VMI PIC X.                              00002800

00132 002700 02 INPUT-TEXT-EDITED.
00133 002800 05 PNO-IN PIC X(5).
00134 002900 05 WMS-IN PIC 9(3).
00135 003000 01 SPA-ADDR PIC X(4).
00136 003100 01 SCT-ADDR PIC X(4).
00137 003200 01 ICOM-RETURN PIC S9(7) COMP.
00138 003300 01 DYNAMIC-WORK-SPACE COPY ICOMDWS.
00139 C      01 DYNAMIC-WORK-SPACE.                            00000100
00140 C      02 OUTPUT-MESSAGE.                                00000200
00141 C      04 OMESSG-HDR.                                    00000300
00142 C      06 OMSGH-LENGTH PIC S9999 COMP.                 00000400
00143 C      06 OMSGH-QPR PIC X.                             00000500
00144 C      06 OMSGH-RSCH PIC X.                             00000600
00145 C      06 OMSGH-RSC PIC X.                             00000700
00146 C      06 OMSGH-SSC PIC X.                             00000800
00147 C      06 OMSGH-MMN PIC XXX.                           00000900
00148 C      06 OMSGH-DATE.                                    00001000
00149 C      08 OMSGH-YR PIC 99.                              00001100
00150 C      08 OMSGH-PERIOD PIC X.                          00001200
00151 C      08 OMSGH-JULIAN-DAY PIC 999.                   00001300

```

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 3 of 11)

00152 C		06	OMSGH-TIME.		00001400
00153 C		08	OMSGH-HM	PIC 99.	00001500
00154 C		08	OMSGH-MM	PIC 99.	00001600
00155 C		08	OMSGH-SS	PIC 99.	00001700
00156 C		08	OMSGH-TM	PIC 99.	00001800
00157 C		06	OMSGH-TID.		00001900
00158 C		08	OMSGH-TI1	PIC X.	00002000
00159 C		08	OMSGH-TI2-3	PIC XX.	00002100
00160 C		08	OMSGH-TI4-5	PIC 99.	00002200
00161 C		06	OMSGH-COM	PIC S9999 COMP.	00002300
00162 C		06	OMSGH-PID	PIC X(5).	00002400
00163 C		06	OMSGH-SSCH	PIC X.	00002500
00164 C		06	OMSGH-USR	PIC X.	00002600
00165 C		06	OMSGH-PMN	PIC XX.	00002700
00166 C		06	OMSGH-LOG	PIC X.	00002750
00167 C		06	OMSGH-PLK	PIC X.	00002800
00168 C		06	OMSGH-VMI	PIC X.	00002900
00170	003400	04	TEXT-OUT	PIC X(150).	
00171	003500	02	FIXED-TEXT.		
00172	003600	04	FIXED-FORMAT-DATA	PIC X(12).	
00173	003700	04	OUT-PART-DATA	PIC X(64).	
00174	003800	04	OUT-PART-PRC	PIC \$\$\$9.9999.	
00175	003900	04	WHS-OUT	PIC ZZZZ9.	
00176	004000	04	OUT-STOCK-DATA.		
00177	004100	06	OUT-STOCK-WLC	PIC X(23).	
00178	004200	06	OUT-STOCK-LEV	PIC Z,ZZZ,999.	
00179	004300	06	OUT-STOCK-LDT	PIC X(8).	
00180	004400	06	OUT-STOCK-ORD	PIC Z,ZZZ,999.	
00181	004500	06	OUT-STOCK-ODT	PIC X(8).	
00182	004600	02	ERROR-TEXT.		
00183	004700	04	ERROR-MSG-FORMAT-ID	PIC X(16).	
00184	004800	04	ERROR-MESSAGE	PIC X(50).	
00185	004900	02	RECORD-AREA.		
00186	005000	04	PART-RECORD.		
00187	005100*		NOTE 100 CHARACTER BDAM RECORD WITHOUT KEYS.		
00188	005200	06	P-REC-PART-DATA.		
00189	005300	08	P-REC-PIN	PIC X(5).	
00190	005400	08	P-REC-DES	PIC X(54).	
00191	005500	08	P-REC-UNT	PIC X(5).	
00192	005600	06	P-REC-PRC	PIC 99V9(4) COMP-3.	
00193	005700	06	P-REC-MFR-NUM	PIC X(15).	
00194	005800	06	FILLER	PIC X(17).	
00195	005900	04	STOCK-RECORD.		
00196	006000*		NOTE 80 CHARACTER VSAM RECORD.		
00197	006100	06	DELETE-CHARACTER	PIC X.	
00198	006200	06	S-REC-KEY-FIELD.		
00199	006300	08	S-REC-WHS	PIC 9(3).	
00200	006400	08	S-REC-PNO	PIC 9(5).	
00201	006500	06	FILLER	PIC X(28).	
00202	006600	06	S-REC-STOCK-DATA.		
00203	006700	08	S-REC-WLC	PIC X(23).	

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 4 of 11)

```

5

00204 006800      08 S-REC-LEV      PIC 9(7)          COMP-3.
00205 006900+    NOTE S-REC-LEV IS 4 CHARACTERS LONG.
00206 007000      08 S-REC-LDT      PIC X(6).
00207 007100      08 S-REC-ORD      PIC 9(7)          COMP-3.
00208 007200+    NOTE S-REC-ORD IS 4 CHARACTERS LONG.
00209 007300      08 S-REC-ODT      PIC X(6).
00210 007400+
00211 007500      02 STATWD          PIC S9(7)          COMP SYNC.
00212 007600+    NOTE THIS PUTS US ONTO A FULLWORD BOUNDARY ALIGNMENT.
00213 007700      02 FHR-STATUS REDEFINES STATWD.
00214 007800      04 FHR-STAT1      PIC X.
00215 007900      88 IOK                      VALUE 0.
00216 008000      88 IOERROR                    VALUE 1.
00217 008100      88 NOT-FOUND                  VALUE 2.
00218 008200      88 NO-DD                      VALUE 9.
00219 008300      04 F-H-STAT2      PIC X.
00220 008400      04 FILLER          PIC X(2).
00221 008500      02 EXTOSCT      PIC X(48).
00222 008600+    NOTE WE ARE STILL ALIGNED HERE.
00223 008700      02 RBN-WORD      PIC S9(7)          COMP.
00224 008800      02 RBN-FILLER REDEFINES RBN-WORD.
00225 008900      04 FILLER          PIC X.
00226 009000      04 RBN          PIC X(3).
00227 009100      02 CP-RETURN      PIC 9(2).
00228 009200      88 SUCCESSFUL-COBSUT      VALUE 0.
00229 009300      02 CURRENT-FILE      PIC X(8).
00230 009400      02 WORD          PIC S9(7)          COMP SYNC.
00231 009500      02 FHR-STATUS REDEFINES WORD.
00232 009600      04 FHR-STAT1      PIC X.
00233 009700      88 NO-DDR                      VALUE 9.
00234 009800      04 FILLER          PIC X(3).
00235 009900      02 FHR-READ-FLAG      PIC X.
00236 010000      88 BDAM                      VALUE 'D'.
00237 010100      02 KEY-FIELD      PIC 9(8).
00238 010200      02 PARTNUMBER.
00239 010300      04 FILLER          PIC 9(4).
00240 010400      04 P-DIGIT        PIC 9.
00241 010500      02 DATE-EDIT.
00242 010600      04 D-E-MO          PIC X(2).
00243 010700      04 D-E-DAY        PIC X(2).
00244 010800      04 D-E-YEAR      PIC X(2).
00245 010900      04 FILLER          PIC X(2).
00246 011000      02 DATE-MOVE.
00247 011100      04 D-M-MO          PIC X(2).
00248 011200      04 SLASH2        PIC X.
00249 011300      04 D-M-DAY      PIC X(2).
00250 011400      04 SLASH1        PIC X.
00251 011500      04 D-M-YEAR      PIC X(2).
00252 011600      02 NO-PART-MESSAGE.
00253 011700      04 MSG-1          PIC X(12).
00254 011800      04 NOPART-PNO      PIC X(5).
00255 011900      04 MSG-2          PIC X(11).

```

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 5 of 11)

6

```
00256 012000 02 NOWARES-MESSAGE.
00257 012100 04 MSG-3 PIC X(5).
00258 012200 04 NOWARES-PNO PIC X(5).
00259 012300 04 MSG-4 PIC X(24).
00260 012400 04 NOWARES-WMS PIC X(3).
00261 012500 02 CANCEL-MESSAGE.
00262 012600 04 CAN-CODE PIC X(13) JUST RIGHT.
00263 012700 04 CAN-FILE-NAME PIC X(8).
00264 012800 04 MSG-5 PIC X(20).
```

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 6 of 11)

```

7

00266 013000 PRCCEDURE DIVISION USING INPUT-MESSAGE
00267 013100 SPA-ADDR
00268 013200 SCT-ADDR
00269 013300 ICOM-RETURN
00270 013400 DYNAMYC-WORK-SPACE.

00272 013600 0100-MAIN-LINE.
00273 013700 MOVE +0 TO ICOM-RETURN.
00274 013800 MOVE MESSG-HDR TO OMESSG-HDR.
00275 013900 MOVE OMSGH-RSCH TO OMSGH-SSCH.
00276 014000 MOVE OMSGH-RSC TO OMSGH-SSC.
00277 014100 PERFORM 1000-READ-PART-FILE.
00278 014200 PERFORM 1100-FH-BDAM-READ.
00279 014300 IF IOK
00280 014400 AND NOT NO-ODR
00281 014500 AND P-REC-PIN EQUAL PNO-IN
00282 014600 PERFORM 2000-READ-STOCK-FILE
00283 014700 PERFORM 2100-FH-VSAM-READ
00284 014800 IF IOK AND NOT NO-ODR
00285 014900 PERFORM 4000-SEND-REPLY-MESSAGE.
00286 015000 PERFORM 4100-COBPUT-CALL.
00287 015100 GOBACK.

00289 015300 1000-READ-PART-FILE.
00290 015400 MOVE PNO-IN TO PARTNUMBER.
00291 015500 MOVE P-DIGIT TO R9N-WORD.
00292 015600 MOVE DD-PART TO CURRENT-FILE.

00294 015800 1100-FH-BDAM-READ.
00295 015900 MOVE BDAM-READ-VALUE TO FH-READ-FLAG.
00296 016000 PERFORM 3000-FH-SELECT-ROUTINE.
00297 016100 IF NO-DD
00298 016200 PERFORM 9900-NO-DD-ROUTINE
00299 016300 ELSE
00300 016400 PERFORM 1200-FH-BDAM-READ-CONTINUE
00301 016500 IF NO-ODR
00302 016600 PERFORM 9000-NO-DD-ROUTINE
00303 016700 ELSE
00304 016800 IF IOERROR
00305 016900 PERFORM 9100-IO-ERROR-ROUTINE
00306 017000 ELSE
00307 017100 IF NOT-FOUND
00308 017200 PERFORM 9150-NOT-FOUND-RTN
00309 017300 ELSE
00310 017400 IF P-REC-PIN NOT EQUAL PNO-IN
00311 017500 PERFORM 9150-NOT-FOUND-RTN
00312 017600 ELSE
00313 017700 MOVE P-REC-PART-DATA TO OUT-PART-DATA
00314 017800 MOVE P-REC-PRC TO OUT-PART-PRC.

```

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 7 of 11)

```

      8

00316 018000 1200-FH-SDAM-READ-CONTINUE.
00317 018100     MOVE SPACES TO FH-STATUS.
00318 018200     CALL 'COBREENT' USING FH-READ
00319 018300             EXTDSCT
00320 018400             FH-STATUS
00321 018500             PART-RECORD
00322 018600             RBN.
00323 018700     PERFORM 3100-FH-RELEASE-ROUTINE.

00325 018900 2000-READ-STOCK-FILE.
00326 019000     MOVE DD-STOCK TO CURRENT-FILE.
00327 019100     MOVE WHS-IN TO S-REC-WHS.
00328 019200     MOVE PNO-IN TO S-REC-PNO.
00329 019300     MOVE S-REC-KEY-FIELD TO KEY-FIELD.

00331 019500 2100-FH-VSAM-READ.
00332 019600     MOVE ZERO TO FH-READ-FLAG.
00333 019700     MOVE LOW-VALUES TO EXTDSCT.
00334 019800     PERFORM 3000-FH-SELECT-ROUTINE.
00335 019900     IF NO-DD
00336 020000         PERFORM 9000-NO-DD-ROUTINE
00337 020100     ELSE
00338 020200         PERFORM 2200-FH-VSAM-READ-CONTINUE
00339 020300         IF NO-DDR
00340 020400             PERFORM 9000-NO-DD-ROUTINE
00341 020500         ELSE
00342 020600             IF IOERROR
00343 020700                 PERFORM 9100-IO-ERROR-ROUTINE
00344 020800             ELSE
00345 020900                 IF NOT-FOUND
00346 021000                     MOVE MSG-C TO MSG-3
00347 021100                     MOVE MSG-D TO MSG-4
00348 021200                     MOVE PNO-IN TO NOWARES-PNO
00349 021300                     MOVE WHS-IN TO NOWARES-WHS
00350 021400                     MOVE NOWARES-MESSAGE TO ERROR-MESSAGE
00351 021500                     PERFORM 9200-SEND-ERROR-MESSAGE
00352 021600             ELSE
00353 021700                 MOVE S-REC-WLC TO OUT-STOCK-WLC
00354 021800                 MOVE S-REC-LEV TO OUT-STOCK-LEV
00355 021900                 MOVE S-REC-LDT TO DATE-EDIT
00356 022000                 PERFORM 2300-DATE-EDITING
00357 022100                 MOVE DATE-MOVE TO OUT-STOCK-LDT
00358 022200                 MOVE S-REC-ORD TO OUT-STOCK-ORD
00359 022300                 MOVE S-REC-ODT TO DATE-EDIT
00360 022400                 PERFORM 2300-DATE-EDITING
00361 022500                 MOVE DATE-MOVE TO OUT-STOCK-ODT.

```

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 8 of 11)

```

          9

00363 022700 2200-FH-VSAM-READ-CONTINUE.
00364 022800      MOVE SPACES TO FH-STATUS.
00365 022900      CALL *COBREENT* USING FH-GETV
00366 023000                      EXTDSCT
00367 023100                      FH-STATUS
00368 023200                      STOCK-RECORD
00369 023300                      KEY-FIELD.
00370 023400      PERFORM 3100-FH-RELEASE-ROUTINE.

00372 023600 2300-DATE-EDITING.
00373 023700      MOVE D-E-YEAR TO D-M-YEAR.
00374 023800      MOVE SLASH TO SLASH1.
00375 023900      MOVE D-E-DAY TO D-M-DAY.
00376 024000      MOVE SLASH TO SLASH2.
00377 024100      MOVE D-E-MO TO D-M-MO.

00379 024300 3000-FH-SELECT-ROUTINE.
00380 024400      MOVE SPACES TO FH-STATUS.
00381 024500      CALL *COBREENT* USING FH-SELECT
00382 024600                      EXTDSCT
00383 024700                      FH-STATUS
00384 024800                      CURRENT-FILE.

00386 025000 3100-FH-RELEASE-ROUTINE.
00387 025100      MOVE SPACES TO FHR-STATUS.
00388 025200      CALL *COBREENT* USING FH-RELEASE
00389 025300                      EXTDSCT
00390 025400                      FHR-STATUS.

00392 025600 4000-SEND-REPLY-MESSAGE.
00393 025700      MOVE VMS-IN TO VMS-OUT.
00394 025800      MOVE FIXED-FORMAT-NAME TO FIXED-FORMAT-DATA.
00395 025900      MOVE +192 TO OMSGH-LENGTH.
00396 026000      MOVE HEX-00 TO OMSGH-RSCH.
00397 026100      MOVE *M* TO OMSGH-RSC.
00398 026200      MOVE HEX-72 TO OMSGH-VMI.
00399 026300      MOVE FIXED-TEXT TO TEXT-OUT.

00401 026500 4100-COINPUT-CALL.
00402 026600      CALL *COBREENT* USING COINPUT
00403 026700                      OUTPUT-MESSAGE
00404 026800                      CP-RETURN.
00405 026900      IF SUCCESSFUL-COINPUT
00406 027000          NEXT SENTENCE
00407 027100      ELSE
00408 027200          MOVE CP-RETURN TO ICOM-RETURN.

```

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 9 of 11)


```

10

00410 027400 9000-NO-DD-ROUTINE.
00411 027500     MOVE MSG-E TO MSG-5.
00412 027600     MOVE *CODE 9 ON * TO CAN-CODE.
00413 027700     MOVE CURRENT-FILE TO CAN-FILE-NAME.
00414 027800     MOVE CANCEL-MESSAGE TO ERROR-MESSAGE.
00415 027900* NOTE THIS RETURN CODE CAN CANCEL THE SUBSYSTEM TO SUBSEQUENT
00416 028000* MESSAGES IF SYCTTBL CANCEL=STOP.
00417 028100     PERFORM 9200-SEND-ERROR-MESSAGE.

00419 028300 9100-IO-ERROR-ROUTINE.
00420 028400     MOVE MSG-E TO MSG-5.
00421 028500     MOVE *IO ERROR ON * TO CAN-CODE.
00422 028600     MOVE CURRENT-FILE TO CAN-FILE-NAME.
00423 028700     MOVE CANCEL-MESSAGE TO ERROR-MESSAGE.
00424 028800     PERFORM 9200-SEND-ERROR-MESSAGE.

00426 029000 9150-NOT-FOUND-RTN.
00427 029100     MOVE MSG-A TO MSG-1.
00428 029200     MOVE MSG-B TO MSG-2.
00429 029300     MOVE PNO-IN TO NOPART-PNO.
00430 029400     MOVE NO-PART-MESSAGE TO ERROR-MESSAGE.
00431 029500     PERFORM 9200-SEND-ERROR-MESSAGE.

00433 029700 9200-SEND-ERROR-MESSAGE.
00434 029800     MOVE +108 TO OMSGH-LENGTH.
00435 029900     MOVE HEX-00 TO OMSGH-RSCH.
00436 030000     MOVE *U* TO OMSGH-RSC.
00437 030100     MOVE HEX-50 TO OMSGH-VMI.
00438 030200     MOVE ERROR-FORMAT TO ERROR-MSG-FORMAT-ID.
00439 030300     MOVE ERROR-TEXT TO TEXT-OUT.

```

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL)
(Page 10 of 11)

DNM=5-336	04	MSGH-T11	BLL=3	017	DNM=5-336	DS 1C	DISP
DNM=5-354	04	MSGH-T12-3	BLL=3	018	DNM=5-354	DS 2C	DISP
DNM=5-374	04	MSGH-T14-5	BLL=3	01A	DNM=5-374	DS 2C	DISP-NH
DNM=5-397	03	MSGH-COIN	BLL=3	01C	DNM=5-397	DS 2C	COMP
DNM=5-415	03	MSGH-PID	BLL=3	01E	DNM=5-415	DS 5C	DISP
DNM=5-433	03	MSGH-SSCH	BLL=3	023	DNM=5-433	DS 1C	DISP
DNM=5-452	03	MSGH-USR	BLL=3	024	DNM=5-452	DS 1C	DISP
DNM=5-473	03	MSGH-RMN	BLL=3	025	DNM=5-473	DS 2C	DISP
DNM=5-491	03	MSGH-LOG	BLL=3	027	DNM=5-491	DS 1C	DISP
DNM=6-000	03	MSGH-BLK	BLL=3	02A	DNM=6-000	DS 1C	DISP
DNM=6-018	03	MSGH-VHI	BLL=3	02R	DNM=6-018	DS 1C	DISP
DNM=6-039	02	INPUT-TEXT-EDITED	BLL=3	029	DNM=6-039	DS 1C	DISP
DNM=6-069	03	PND-IN	BLL=3	02A	DNM=6-069	DS 0CL8	GROUP
DNM=6-085	03	VHS-IN	BLL=3	02A	DNM=6-085	DS 5C	DISP
DNM=6-101	01	SPA-ADDR	BLL=4	02F	DNM=6-0A5	DS 3C	DISP-NH
DNM=6-119	01	SCT-ADDR	BLL=5	000	DNM=6-101	DS 4C	DISP
DNM=6-140	01	ICOM-RETURN	BLL=5	000	DNM=6-119	DS 4C	DISP
DNM=6-161	01	DYNAMIC-WORK-SPACE	BLL=6	000	DNM=6-140	DS 4C	COMP
DNM=6-192	02	OUTPUT-MESSAGE	BLL=7	000	DNM=6-161	DS 0CL192	GROUP
DNM=6-219	03	OMESG-HDR	BLL=7	000	DNM=6-192	DS 0CL192	GROUP
DNM=6-242	04	OMSGH-LENGTH	BLL=7	008	DNM=6-219	DS 0CL42	GROUP
DNM=6-267	04	OMSGH-QPR	BLL=7	002	DNM=6-242	DS 2C	COMP
DNM=6-289	04	OMSGH-RSCH	BLL=7	002	DNM=6-267	DS 1C	DISP
DNM=6-309	04	OMSGH-RSC	BLL=7	003	DNM=6-289	DS 1C	DISP
DNM=6-331	04	OMSGH-SSC	BLL=7	004	DNM=6-309	DS 1C	DISP
DNM=6-350	04	OMSGH-MMN	BLL=7	005	DNM=6-331	DS 1C	DISP
DNM=6-372	04	OMSGH-DATE	BLL=7	006	DNM=6-350	DS 3C	DISP
DNM=6-398	05	OMSGH-YR	BLL=7	009	DNM=6-372	DS 0CL6	GROUP
DNM=6-419	05	OMSGH-PERIOD	BLL=7	009	DNM=6-398	DS 2C	DISP-NH
DNM=6-441	05	OMSGH-JULIAN-DAY	BLL=7	008	DNM=6-419	DS 1C	DISP
DNM=6-467	04	OMSGH-TIME	BLL=7	00C	DNM=6-441	DS 3C	DISP-NH
DNM=6-490	05	OMSGH-MM	BLL=7	00F	DNM=6-467	DS 0CL8	GROUP
DNM=7-000	05	OMSGH-SS	BLL=7	011	DNM=6-490	DS 2C	DISP-NH
DNM=7-018	05	OMSGH-TH	BLL=7	013	DNM=7-000	DS 2C	DISP-NH
DNM=7-036	05	OMSGH-TD	BLL=7	015	DNM=7-018	DS 2C	DISP-NH
DNM=7-054	04	OMSGH-T11	BLL=7	017	DNM=7-036	DS 2C	DISP-NH
DNM=7-076	05	OMSGH-T12-3	BLL=7	017	DNM=7-054	DS 0CL5	GROUP
DNM=7-116	05	OMSGH-T14-5	BLL=7	018	DNM=7-076	DS 1C	DISP
DNM=7-137	04	OMSGH-COIN	BLL=7	018	DNM=7-116	DS 2C	DISP-NH
DNM=7-159	04	OMSGH-PID	BLL=7	01C	DNM=7-137	DS 2C	COMP
DNM=7-178	04	OMSGH-SSCH	BLL=7	01E	DNM=7-159	DS 5C	DISP
DNM=7-198	04	OMSGH-USR	BLL=7	023	DNM=7-178	DS 1C	DISP
DNM=7-220	04	OMSGH-RMN	BLL=7	024	DNM=7-198	DS 1C	DISP
DNM=7-242	04	OMSGH-LOG	BLL=7	025	DNM=7-220	DS 2C	DISP
DNM=7-261	04	OMSGH-BLK	BLL=7	028	DNM=7-242	DS 1C	DISP
DNM=7-283	04	OMSGH-VHI	BLL=7	029	DNM=7-261	DS 1C	DISP
DNM=7-302	03	TEXT-OUT	BLL=7	029	DNM=7-283	DS 1C	DISP
DNM=7-320	02	FIXED-TEXT	BLL=7	0C0	DNM=7-302	DS 150C	DISP
DNM=7-343	03	FIXED-FURNAT-DATA	BLL=7	0C0	DNM=7-320	DS 0CL146	GROUP
DNM=7-370	03	OUT-PART-DATA	BLL=7	0C0	DNM=7-343	DS 12C	DISP
DNM=7-393	03	OUT-PART-PRC	BLL=7	10C	DNM=7-370	DS 64C	DISP
					DNM=7-393	DS 8C	NH-EDIT

Figure 56. Sample Inquiry Subsystem; Reentrant (IBM ANS COBOL) (Page 11 of 11)

```

//TABLES      JOB
//*
//*           DEFINE SYCTTBL FOR SUBSYSTEM
//*
//STEP1       EXEC  LIBELINK,Q=TEST,NAME=INTSCT,LMOD=INTSCT
//LIB.SYSIN   DD    *
./ ADD NAME=USRSCTS
./ NUMBER     NEW1=100,INCR=100
USRSCTS       DS    OH
RQ            SYCTTBL SUBH=R,SUBC=Q,SBSP=SQCOBOL,LANG=RCOB,OVLV=0,      X
              NUMCL=10,MNCL=1,TCTV=60,GET=792

/*
//ASM.SYSIN   DD    DSN=INT.SYMREL(INTSCT),DISP=SHR
//*
//*           DEFINE EDIT CONTROL TABLE ENTRY
//*
//STEP2       EXEC  LIBELINK,Q=TEST,NAME=PMIVERBS,LMOD=PMIVERBS
//LIB.SYSIN   DD    *
./ ADD NAME=USRVERBS
./ NUMBER     NEW1=100,INCR=100
USRVERBS      DS    OH
RTRQECT       VERB  RTRQ,D9,256,2,FIX=YES
              PARM  P/N,1,7,5,10000111
              PARM  WHS,2,7,3,10000111

/*
//ASM.SYSIN   DD    DSN=INT.SYMREL(PMIVERBS),DISP=SHR
//*
//*           DEFINE CHANGE/DISPLAY TABLE
//*
//STEP3       EXEC  LIBELINK,Q=TEST,NAME=CHNGTB,LMOD=CHNGTB
//LIB.SYSIN   DD    *
./ ADD NAME=CHNGTB
./ NUMBER     NEW1=100,INCR=100
CHTB          TITLE 'CHNGTB - FIXED FORMAT OUTPUT-DESCRIPTOR NAME TABLE'
CHNGTB        CSECT
              DC    CL8'SSRQ0001' USED ONLY TO TEST COBOL PGM. GUIDE S/S
              DC    F'0'
              PMISTOP
              END
//

```

Figure 57. Table Updates to Implement Test Mode Testing

```

* OUTPUT FORMAT TABLE FOR SAMPLE INQUIRY SUBSYSTEM
*
OFT100  REPORT NUM=100,LINES=8
        LINE NUM=1, ITEMS=1
        ITEM CODE=255,DATA=STOCK STATUS REQUEST, FROM=6, TO=25
        LINE NUM=2, ITEMS=2
        ITEM CODE=255,DATA=PART NUMBER, FROM=1, TO=11
        ITEM CODE=12, FROM=13, TO=17
        LINE NUM=3, ITEMS=2
        ITEM CODE=255,DATA=DESCRIPTION, FROM=1, TO=11
        ITEM CODE=21, FROM=13, TO=66
        LINE NUM=4, ITEMS=4
        ITEM CODE=255,DATA=ORDER UNITS, FROM=1, TO=11
        ITEM CODE=18, FROM=13, TO=17
        ITEM CODE=255,DATA=PRICE, FROM=19, TO=23
        ITEM CODE=19, FROM=25, TO=32
        LINE NUM=5, ITEMS=2
        ITEM CODE=255,DATA=STOCK STATUS AT WAREHOUSE, FROM=1, TO=25
        ITEM CODE=8, FROM=27, TO=31
        LINE NUM=6, ITEMS=2
        ITEM CODE=255,DATA=LOCATION, FROM=4, TO=11
        LINE NUM=7, ITEMS=4
        ITEM CODE=255,DATA=ON HAND, FROM=6, TO=12
        ITEM CODE=13, FROM=15, TO=23
        ITEM CODE=14, FROM=38, TO=45
        ITEM CODE=255,DATA=AS OF, FROM=31, TO=35
        LINE NUM=8, ITEMS=4
        ITEM CODE=255,DATA=ON ORDER, FROM=6, TO=13
        ITEM CODE=15, FROM=15, TO=23
        ITEM CODE=255,DATA=AS OF, FROM=31, TO=35
        ITEM CODE=16, FROM=38, TO=45
        END
00000100
00000200
00000300
00000400
00000500
00000600
00000700
00000800
00000900
00001000
00001100
00001200
00001300
00001400
00001500
00001600
00001700
00001800
00001900
00002000
00002100
00002200
00002300
00002400
00002500
00002600
00002700
00002800
00002900
00003000
00003100
00003200
00003300

```

Figure 58. Utilities Table Coding for Test Mode Subsystem (Page 1 of 2)

<pre> * OUTPUT FORMAT TABLE FOR ERROR MESSAGES FROM INQUIRY SUBSYSTEM * OFT501 REPORT NUM=501,LINES=1 LINE NUM=1,ITEMS=2 ITEM CODE=255,FROM=1,TO=10,DATA=***ERROR** ITEM CODE=249,FROM=12,TO=62 END </pre>	<pre> 00000100 00000200 00000300 00000400 00000500 00000600 00000700 </pre>
<pre> * FILE DESCRIPTION RECORD FOR FIXED FORMAT OUTPUT * FROM SAMPLE INQUIRY SUBSYSTEM DES00001 CSECT SSRQ100 FDHDR NAME=SSRQ0001,RPTNO=100,FIELDS=10 PNQ12 FDETL OFSET=0,LEN=5,NAME=P/NXX,CODE=12 DES21 FDETL OFSET=5,LEN=54,NAME=DESEX,CODE=21 UNT18 FDETL OFSET=59,LEN=5,NAME=UNTX,CODE=18 PRC19 FDETL OFSET=64,LEN=8,NAME=PRCXX,CODE=19 WHS08 FDETL OFSET=72,LEN=5,NAME=WHSXX,CODE=8 WLC10 FDETL OFSET=77,LEN=23,NAME=WLCXX,CODE=10 LEV13 FDETL OFSET=100,LEN=9,NAME=LEVXX,CODE=13 LDT14 FDETL OFSET=109,LEN=8,NAME=LDTXX,CODE=14 ORD15 FDETL OFSET=117,LEN=9,NAME=ORDXX,CODE=15 ODT16 FDETL OFSET=126,LEN=8,NAME=ODTXX,CODE=16 END </pre>	<pre> 00000100 00000200 00000300 00000400 00000500 00000600 00000700 00000800 00000900 00001000 00001100 00001200 00001300 00001400 00001500 </pre>

Figure 58. Utilities Table Coding for Test Mode Subsystem (Page 2 of 2)

Card		Contents										
HEADER	1-3	MSG										
	*6-8	Low-order byte of S/S code (MSGHRSC) (or 8)										
	*9-11	Hi-order byte of S/S code (MSGHRSCH) (or 11)										
	20-24	Sending terminal ID (MSGHTID)										
	50-53	Front-end Message Number (MSGHBMN)										
	*55-57	VMI value (MSGHVMI); leave blank if EDIT required; code 255 if no editing by Edit Utility (or 57).										
DETAIL(s)	1-64**	Data for one line of input message. If VMI in header card is left blank, a new line character is inserted at end of text on every card except last one. If the last non-blank character is a \$ sign (X'5B'), it will be replaced by a NL; the preceding character (usually a blank) is kept as part of the input. All NL's are suppressed if editing is not required.										
TRAILER	1-3	Generates End of Transmission character following the last non-blank character of the previous detail card. <table border="0"> <thead> <tr> <th><u>Contents of Card</u></th> <th><u>Ending Character</u></th> </tr> </thead> <tbody> <tr> <td>EMS</td> <td>EOT (X'37')</td> </tr> <tr> <td>EOT</td> <td>EOT (X'37')</td> </tr> <tr> <td>ETX</td> <td>ETX (X'03')</td> </tr> <tr> <td>ETB</td> <td>ETB (X'26')</td> </tr> </tbody> </table>	<u>Contents of Card</u>	<u>Ending Character</u>	EMS	EOT (X'37')	EOT	EOT (X'37')	ETX	ETX (X'03')	ETB	ETB (X'26')
<u>Contents of Card</u>	<u>Ending Character</u>											
EMS	EOT (X'37')											
EOT	EOT (X'37')											
ETX	ETX (X'03')											
ETB	ETB (X'26')											
<p>*3-digit integer values (from 000 to 255) or a corresponding single alphanumeric character in the low-order field position.</p> <p>**64 is default maximum. See the <u>Operating Reference Manual</u> if necessary to alter this specification.</p>												

Figure 59. Test Mode Message Card Formats

MSG	Q	R	TEST1	0001
RTRQ				
P/N	12345			
WHS	200			
EMS				
MSG	Q	R	TEST1	0002
RTRQ				
P/N	55555			
WHS	200			
EMS				
MSG	Q	R	TEST1	0003
RTRQ				
P/N	12345			
WHS	300			
EMS				
MSG	Q	R	TEST1	0004
RTRQ				
P/N	12349			
WHS	200			
EMS				
MSG	Q	R	TEST1	0005
RTRQ				
P/N	12341			
WHS	100			
EMS				
MSG	Q	R	TEST1	0006
RTRQ				
P/N	A2345			
WHS	400			
EMS				

Figure 60. Sample Input Test Messages for Test Mode

```

//EXECUTEST JOB (ICOMTEST,,20),'ICOM TEST SQCOBOL',CLASS=A,
//  RESTART=(GENLINK.ASM)
//PROCLIB DD DSN=INT.PROCLIB,DISP=SHR          (AS NEEDED)
//*****
//* THE RESTART PARM IN THE JOB STATEMENT RESTARTS THE TEST AT THE *
//* BEGINNING. IF YOU WISH TO RESTART AT A DIFFERENT STEP, CODE *
//* RESTART=STEPNAME OR RESTART=STEPNAME.PROCSTEPNAME *
//* *
//* NOTE: WHEN USING A VSAM FILE, IT MAY BE NECESSARY TO EXECUTE *
//* IDCAMS TO VERIFY THE FILE IF A PREVIOUS EXECUTION ABENDED. *
//*****
//*
//*****
//* STEP GENLINK GENERATES A STANDARD TEST MODE LINKEDIT DECK *
//* VIA ASSEMBLY OF THE ICOMLINK MACRO. *
//* THE GENERATED DECK (TESTLINK) IS PLACED ON INT.SYMTEST. *
//*****
//GENLINK EXEC ASMPQ,Q=LIB,U=REL,DECK=DECK
//ASM.SYSIN DD *
//          ICOMLINK TEST=YES,MMU=NO,STORFCH=NO,COBOL=YES,RECOBOL=YES
//          END
//SYSPUNCH DD DSN=INT.SYMTEST(TESTLINK),DISP=SHR
//*
//*****
//* STEPS SCRSCR AND ALLOCSCR DELETE AND RE-ALLOCATE THE LOAD *
//* MODULE LIBRARY USED IN THE TEST (ALSO USED FOR DYNLLIB) *
//*****
//SCRSCR EXEC PGM=IEFBRI4
//FILE1 DD DSN=INT.MODSCR,DISP=(OLD,DELETE)
//ALLOCSCR EXEC PGM=IEFBRI4
//A DD DSN=INT.MODSCR,DISP=(,CATLG),UNIT=SYSDA,
// DCB=INT.MODREL,VOL=SER=INT001,SPACE=(CYL,(3,,7))

```

NOTE: JCL requirements vary by installation requirements. The above example illustrates representative JCL. The installation System Manager should verify JCL to use.

Figure 61. Linkedit and Execution JCL for Test Mode (Page 1 of 3)


```

//*****
//* STEP GENINCL CREATES INCLUDE CARDS USED BY THE LINK EDIT STEP *
//* THE ADDED INCLUDE STATEMENTS ARE FOR THE SAMPLE SUBSYSTEM AND *
//* THE REFERENCED OFTS (INCLUDE AFTER PMIRCNTB). *
//* IF THE TEST1 TERMINAL IS NOT IN THE SYSTEM PMISTATB TABLE, USE: *
//* INCLUDE MODREL(PMISTATB) *
//* INCLUDE MODREL(PMIDEVTB) *
//* INCLUDE MODREL(PMIBROAD) *
//* THE ABOVE ASSUMES THE CONTROL TERMINAL IS NAMED CNT01. *
//* *** BEFORE THIS STEP, SEQUENCE NUMBER THE TESTLINK SOURCE. ***** *
//*****
//GENINCL EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=INT.SYMTTEST,DISP=SHR
//SYSUT2 DD DSN=&&INCL,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(1,1,1)),
// DCB=(BLKSIZE=80,LRECL=80)
//SYSIN DD *
./ CHANGE NAME=TESTLINK,LIST=ALL
        INCLUDE SYSLIB(SQCOBOL)          SAMPLE SUBSYSTEM          00000010
        INCLUDE SYSLIB(RPT00100)        DISPLAY OFT FOR SUBSYSTEM  01981000
        INCLUDE SYSLIB(RPT00501)        ERROR MESSAGES OFT       01982000
//*****
//* LINK EDIT THE TEST INTERCOMM SYSTEM *
//* NOTE: THE INTERCOMM PROC 'LKEDT' LINKEDITS MODULES FROM THE *
//* SYSLIB CONCATENATION STREAM AS FOLLOWS - *
//* THE LOAD LIBRARY SPECIFIED BY THE Q= PARAMETER, *
//* FOLLOWED BY MODULES FOUND IN MODUSR, MODLIB, THEN MODREL. *
//* THEN, SYS1.COBLIB (FOR OS/V5 COBOL). *
//* THE INTERCOMM LOAD MODULE IS PLACED ON INT.MODSCR. *
//* IT IS NOT NECESSARY TO RE-DO THE WHOLE LINK TO REPLACE 1 MODULE *
//* IN THIS CASE, ALL YOU SHOULD DO IS: *
//* 1) REASSEMBLE OR RECOMPILE THE CHANGED/NEW MODULE INTO A *
//* SEPARATE LOAD LIBRARY *
//* 2) OVERRIDE THE SYSLIN DD STMT TO //LKED.SYSLIN DD *
//* FOLLOW IT WITH INCLUDE CARDS *
//* FOR THE MODULES YOU WISH TO REPLACE *
//* 3) FOLLOW THOSE INCLUDES WITH THE FOLLOWING 3 CARDS: *
//* INCLUDE SYSLMOD(TESTICOM) *
//* ENTRY PMISTUP *
//* NAME TESTICOM(R) *
//* 4) INSERT A DD STMT FOR THE LOAD LIBRARY ON WHICH THE *
//* REPLACEMENT MODULES RESIDE *
//* 5) CHANGE THE RESTART PARM ON THE JOB STATEMENT *
//* TO POINT TO THE LKED STEP *
//*****
//LKED EXEC LKEDT,LMOD=TESTICOM,Q=TEST,
// PARM.LKED='LIST,LET,XREF,NCAL,SIZE=(250K,100K)'
//LKED.SYSLIN DD DSN=&&INCL(TESTLINK),DISP=(OLD,PASS)
//MODREL DD DSN=INT.MODREL,DISP=SHR

```

Figure 61. Linkedit and Execution JCL for Test Mode (Page 2 of 3)

```

//*****
//* EXECUTE INTERCOMM IN TESTMODE
//*****
//GO EXEC PGM=TESTICOM,PARM='TEST',TIME=(,30)
//STEPLIB DD DSN=INT.MODSCR,DISP=(OLD,PASS) (DYNLLIB)
// DD DSN=INT.MODUSR,DISP=SHR (USER LOAD LIBRARY)
// DD DSN=INT.MODLIB,DISP=SHR (SYSTEM UPDATE LIBRARY)
// DD DSN=INT.MODREL,DISP=SHR (SYSTEM RELEASE LIBRARY)
// DD DSN=SYS1.COBLIB,DISP=SHR (COBOL LOAD LIBRARY)
//INTERLOG DD DSN=INTLOG,DISP=(NEW,PASS),
// SPACE=(TRK,(10,5)),VOL=SER=INT001,UNIT=SYSDA,
// DCB=(DSORG=PS,RECFM=VB,BLKSIZE=4096,LRECL=4092,NCP=8,OPTCD=C)
//STSLOG DD SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=120,RECFM=FA)
//SMLOG DD SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=120,RECFM=FA)
//SYSPRINT DD SYSOUT=A,DCB=(DSORG=PS,RECFM=VA,BLKSIZE=141,LRECL=137)
//RCT000 DD DSN=INT.RCT000,DISP=SHR,
// DCB=(DSORG=DA,OPTCD=RF) OUTPUT FORMATS
//PMIQUE DD DISP=OLD,DSN=INT.PMIQUE,
// DCB=(DSORG=DA,OPTCD=R) SUBSYSTEM DISK QUEUE
//STOKFILE DD DSN=VSAMSD1.STCKFILE.CLUSTER,DISP=OLD,
// AMP=(AMORG,'RECFM=F') VSAM TEST FILE
//PARTFILE DD DSN=INT.TEST.PARTFILE,DISP=OLD,
// DCB=(DSORG=DA,OPTCD=R) BDAM TEST FILE
//DES000 DD DSN=INT.DES000,DISP=SHR,
// DCB=(DSORG=DA,OPTCD=RF) FILE DESCRIPTION RECORDS
//SYSIN DD DSN=INT.SYMTEST(TESTMSG),DISP=SHR,
// DCB=DSORG=PS TEST MODE INPUT MESSAGES
//PMISTOP DD DUMMY
//ICOMIN DD DUMMY (ADD FAR PARMS IF DESIRED)
//*
//STEPCAT DD DSN=VSAMSD1,DISP=SHR VSAM CATALOG (IF NEEDED)
//* DYNAMIC LINKEDIT DATA SETS (IF NEEDED)
//DYNLPRNT DD SYSOUT=A
//DYNLLIB DD DSN=INT.MODSCR,DISP=(OLD,PASS)
//DYNLWORK DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(,PASS) (REL 9 ONLY)
//*
//SNAPDD DD SYSOUT=A
//SYSSNAP DD SYSOUT=A SNAP INPUT TEST MESSAGES
//SYSSNAP2 DD SYSOUT=A SNAP OUTPUT TEST MESSAGES
//SYSUDUMP DD SYSOUT=A
//*
//ABNLIGNR DD DUMMY FORCE ABEND-AID TO IGNORE DUMP (PRODUCE IBM DUMP)
//*****
//* PRINT INTERCOMM LOG FROM TEST MODE RUN
//*****
//INTERLOG EXEC PGM=LOGPRINT,COND=EVEN
//STEPLIB DD DSN=INT.MODREL,DISP=SHR
//SYSPRINT DD SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=121)
//INTERLOG DD DSN=INTLOG,DISP=SHR,DCB=BLKSIZE=5000
//SYSIN DD DUMMY,DCB=BLKSIZE=80
//

```

Figure 61. Linkedit and Execution JCL for Test Mode (Page 3 of 3)

JOB WSDCVHIT	STEP 60	TIME 174531	DATE 63032	ID = 015	CPUID = 000204333033	PAGE 0001
-STORAGE						
1173E0	00000000	00000000	00000000	00000000	004102D9 D8000000RO.....
117400	000009E3	D9D815D7	61D540F1	F2F3F4F5	C5E2E3F1 00010000 00000000 00000100TEST1.....
117420	000009E3	D9D815D7	61D540F5	F5F5F5F5	15E6C8E2 40F2F0F0 37000000	...RTRQ.P.N 12345.WHS 200.....
-STORAGE						
1173E0	00000000	00000000	00000000	000000E3	004102D9 D8000000RO.....
117400	000009E3	D9D815D7	61D540F5	F5F5F5F5	C5E2E3F1 00010000 00000000 00000200TEST1.....
117420	000009E3	D9D815D7	61D540F1	F2F3F4F5	15E6C8E2 40F2F0F0 37000000	...RTRQ.P.N 55555.WHS 200.....
-STORAGE						
1173E0	00000000	00000000	00000000	000000E3	004102D9 D8000000RO.....
117400	000009E3	D9D815D7	61D540F1	F2F3F4F5	C5E2E3F1 00010000 00000000 00000300TEST1.....
117420	000009E3	D9D815D7	61D540F1	F2F3F4F5	15E6C8E2 40F2F0F0 37000000	...KTRQ.P.N 12345.WHS 300.....
-STORAGE						
1173E0	00000000	00000000	00000000	000000E3	004102D9 D8000000RO.....
117400	000009E3	D9D815D7	61D540F1	F2F3F4F5	C5E2E3F1 00010000 00000000 00000400TEST1.....
117420	000009E3	D9D815D7	61D540F1	F2F3F4F5	15E6C8E2 40F2F0F0 37000000	...RTRQ.P.N 12345.WHS 200.....
-STORAGE						
1173E0	00000000	00000000	00000000	000000E3	004102D9 D8000000RO.....
117400	000009E3	D9D815D7	61D540C1	F2F3F4F5	C5E2E3F1 00010000 00000000 00000500TEST1.....
117420	000009E3	D9D815D7	61D540F1	F2F3F4F5	15E6C8E2 40F1F0F0 37000000	...RTRQ.P.N 12341.WHS 100.....
-STORAGE						
1173E0	00000000	00000000	00000000	000000E3	004102D9 D8000000RO.....
117400	000009E3	D9D815D7	61D540C1	F2F3F4F5	C5E2E3F1 00010000 00000000 00000600TEST1.....
117420	000009E3	D9D815D7	61D540F1	F2F3F4F5	15E6C8E2 40F4F0F0 37000000	...RTRQ.P.N A2345.WHS 400.....

Figure 62. Sample Test Mode Execution Snaps (Page 1 of 3)

JOB WSDCVHTT	STEP 60	TIME 174553	DATE P3032	ID = 020	CPUID = 000204333033	PAGE 0001
-STORAGE						
149720	07ERF300 F0F3F2F1 F7F4F5F5 F3F6F5C3	00680200	E4000000	07F8F300	F0F3F2F1U.....
149740	00505C5C 5C40C7D6 D6C4A0C1 C6E3C5D9	00000000	00000000	00000000	00000000	*.83.03217455365CNT01.....2.....
149760	00505C5C 5C40C7D6 D6C4A0C1 C6E3C5D9	05060605	5C5C4040	C9D5E3C5	D9C3D6D4	*.... GOOD AFTERNOON.. INTERCOM*
149780	D4A0C9E2 40D9C5C1 C4E9A07A 4A4A04F0	F260F0F1	60F8F3A0	40F1F7A8	F4F52637	*M IS READY . 02.01.83 17.45..*
1497A0	40A040A0					
-STORAGE						
14CCA0	F7F4F5F5 F3F6F5E3 C5E2E3F1 00010000	00670200	E4000000	07F8F300	F0F3F2F1	*U.....83.0321*
14CC00	D6C4A0C1 C6E3C5D9 D5D6D6D5 5C5C4040	00000000	000000F2	00505C5C	5C40C7D6	*7455365TEST1.....2..... 60*
14C000	C4E9A07A 40A040F0 F260F0F1 60F8F3A0	C9D5E3C5	D9C3D6D4	D4A0C9F2	40D9C5C1	*0D AFTERNOON.. INTERCOMM IS REA*
		40F1F7A0	F4F537A0			*DY . 02.01.83 17.45. *
-STORAGE						
14CCA0	F7F4F5F5 FAF2FAE3 C5E2E3F1 00010000	01290200	E4C80000	09F8F301	F0F3F2F1	*UH.....R3.0321*
14CC00	E3D6C3D2 40E2E3C1 E3E4E2A0 D9C5D9E4	00000000	000001F2	005040A0	40A040E2	*7455424TEST1.....2.....S*
14C000	F1F2F3F4 F515C4C5 E2C3D9C9 D7E3C906	C5E2E315	D7C1D9L3	40D5E4D4	C2C5D9A0	*T0CK STATUS REQUEST.PART NUMBER *
14C020	C1E2C8C5 D915D6D9 C4C5D9A0 E4D5C9E3	D540F161	F240C9D5	40E2E3C5	C5D3A0F6	*12345-DESCRIPTION 1-2 IN STEEL W*
14C040	F54RFF05 F0F715E2 E3D6C3D2 40E2E3C1	E240C7D9	E24040A0	D7D9C9C3	C5A058F0	*ASHER-ORDER UNITS 6RS PRICE .0*
14C060	40F2F0F0 15A040A0 D3D6C3C1 E3C9D6D5	E3E4E2A0	C1E3A0E6	C1D9C5C8	D6E4E2C5	*5.0507.STOCK STATUS AT WAREHOUSE*
14C080	40A0D6D5 40C8C1D5 C4A040F6 60F1F6F1	40D4C9C1	04C948A0	C6D3C1A8	15A040A0	* 200. LOCATION MIAMI. FLA. *
14C0A0	40A0D6D5 40C8C1D5 C4A040F6 60F1F6F1	68F5F0F6	40A040A0	40A040C1	E2A0D6C6	* ON HAND 6.161.506 AS OF*
14C0C0	F1F740A0 40A040A0 40C1E2A0 D6C640A0	D6D5A0D6	D9C4C5D9	40F46RF0	F4F068F6	* 03.05.82. ON ORDER 4.040.6*
		F1F061F1	F161F8F2	37A040A0		*17 AS OF 10.11.82. *
-STORAGE						
14CCA0	F7F4F5F5 F4F3F2E3 C5E2E3F1 00010000	00520200	E4D80000	0AF8F302	F0F3F2F1UO.....83.0321*
14CC00	D95C5C40 40D7C1D9 E3A0D5E4 D4C2C5D9	00000009	000002E2	00505C5C	C5D9D9D6	*7455432TEST1.....R...2.....ERR0*
14C000	483740A0	40F5F5F5	F5F540D5	06E340C6	D6E4D5C4	*R... PART NUMBER 55555 NOT FOUND*
-STORAGE						
14CCA0	F7F4F5F5 FAF3F6E3 C5E2E3F1 00010000	005B0200	E4D00000	08F8F301	F0F3F2F1UO.....R3.0321*
14CC00	D95C5C40 40D7C1D9 E3A0D5E4 D4C2C5D9	00000009	000003F2	00505C5C	C5D9D9D6	*7455436TEST1.....K....2.....ERR0*
14C000	C5C8D6E4 E2C5A0F3 F0F037A0	D5D6E3A0	C6D6E4D5	C4A0C9D5	40E6C1D9	*R.. PART 12345 NOT FOUND IN WAR*
						*EHOUSE 300. *
-STORAGE						
14CCA0	F7F4F5F5 FAF4F5E3 C5E2E3F1 00010000	00520200	E4D80000	0CF8F301	F0F3F2F1UO.....R3.0321*
14CC00	D95C5C40 40D7C1D9 E3A0D5E4 D4C2C5D9	00000009	00000AF2	00505C5C	C5D9D9D6	*7455445TEST1.....K....2.....ERR0*
14C000	4R3740A0	40F1F2F3	F4F940D5	D6E340C6	D6E4D5C4	*R.. PART NUMNER 12349 NOT FOUND*

Figure 62. Sample Test Mode Execution Snaps (Page 2 of 3)

JOB WSDCVHTT	STEP GO	TIME	DATE	8X03P	ID = 020	CPUID = 000204333033	PAGE 0001
-STORAGE							
14CCA0	F7F4F5F5	F4F4F9E3	C5E2E3F1	00010000	00850200	D5E80000	0EF8F302 F0F3F2F1
14CCCO	F0F0F2F9	1505D6D5	60D5E4D4	C5D9C9C3	00000000	000006F2	0050F0F0 C5F4A0F0
14CCCO	F0F0F2F2	1505D6D5	60D5E4D4	C5D9C9C3	40C3C8C1	D9C1C3E3	C5D9A0C7 C9E5C5D5
14CD00	40D6D5A0	D761D5A0	D7C1D9C1	D4C5E3C5	D9A0C6D6	D9A0D9E3	D9D0A0E5 C5D9C2A8
14CD00	40C1D3D3	40C3C8C1	D9C1C3E3	C5D9E2A0	E2C8D6E4	D3C4A0C2	C5A0D5E4 D4C5D9C9
14CD40	C3A015A0	D4C5E2E2	C1C7C5A0	D5D6A8A0	F0F0F0F6	40C6D9D6	D4A0E307 E4A0E3C5
14CD60	E2E3F1A8	37A0A0A0					
-STORAGE							
14CCA0	F7F4F5F5	F4F4F9E3	C5E2E3F1	00010000	00820200	D5E80000	0FF8F302 F0F3F2F1
14CCCO	F0F0F2F2	1505D6D5	E4C9D9C5	C4A0D7C1	00000000	000006F2	0050F0F0 C5F4A0F0
14CCCO	F0F0F2F2	1505D6D5	E4C9D9C5	C4A0D7C1	D9C1D9C3	E3C5D9A0	D761D5A0 E6C1E2A0
14CD00	D6D4C9E3	E3C5C4A0	D6D9A0C7	C9E5C5D5	40C9D5A0	C5D9D9D6	D95D0D6D5 A0E3C8C5
14CD20	40A0D9E3	D9D8A0E5	C5D9C2A8	A0E5C5D9	C2A0E6C1	E2A0C3C1	D5C3C5D3 D3C5C415
14CD40	40D4C5E2	E2C1C7C5	40D5D6A8	40F0F0F0	F6A0C6D9	D6D4A0E3	D7E4A0E3 C5E2E3F1
14CD60	4B37A0A0						
-STORAGE							
14CCA0	F7F4F5F5	F4F4F9E3	C5E2E3F1	00010000	013D0200	EACR0000	10F8F301 F0F3F2F1
14CCCO	E3D6C3D2	A0E2E3C1	E3E4E2A0	D9C5D9E4	00000000	000005F2	0050A0A0 40A0A0E2
14CCCO	F1F2F3FA	F115C4C5	E2C3D9C9	D7E3C9D6	C5E2E315	D7C1D9E3	40D5E4D4 C2C5D9A0
14CD00	C3C5D9D9	C1E3C5CA	A0D3D6C3	D2A0D5E4	D5A0F161	F4A0C9D5	40D3C8D9 D6D4C5A0
14CD00	E9A0A0A0	D7D9C9C3	C5A0D5D1	F6A8F1F6	E315D6D9	C4C5D9A0	E4D5C9E3 E2A0C4D6
14CD60	C1E3A0E6	C1D9C5C8	D6E4E2C5	40F1F0F0	F1F615E2	E3D6C3D2	A0E2E3C1 E3E4E2A0
14CD80	40E8A0D9	D2A0C3C9	E3E8A0A0	D5A0E8A8	15A0A0A0	D3D6C3C1	E3C9D6D5 40D5C5E6
14CD40	68F0F5F0	68F5F0FA	40A0A0A0	40A0A0C1	E2A0D6C6	40A0D6D5	40C8C1D5 C4A0A0F5
14CD00	40A0A0A0	D6D5A0D6	D9C4C5D9	40F568F0	F5F068F5	F0F4A0A0	40A0A0A0 40C1E2A0
14CD00	D6C6A0A0	F1F061F1	F161F8F2	37A0A0A0			
-STORAGE							
149720	12F8F301	F0F3F2F1	F7F4F5F5	F9F0F5C3	00680200	E4000000	
149740	0D50C5C5	5C40C7D6	D6C4A0C1	C6E3C5D9	D5E3F0F1	00010000	00000000 000000F2
149780	D4A0C9E2	40C3D3D6	E2C5C47A	40A0A0F0	D5D6D6D5	5C5C40A0	C9D5E3C5 D9C3D6D4
1497A0	40A0A0A0				F26D0F01	60F8F3A0	40F1F7A8 F4F52637
-STORAGE							
14CCA0	F7F4F5F5	F9F0F5E3	C5E2E3F1	00010000	00670200	E4000000	12F8F301 F0F3F2F1
14CCCO	D6C4A0C1	C6E3C5D9	D5D6D6D5	5C5C40A0	00000000	000000F2	0050C5C5 5C40C7D6
14CCCO	E2C5C47A	40A0A0F0	F26D0F01	60F8F3A0	C9D5E3C5	D9C3D6D4	D4A0C9F2 40C3D3D6
14CD00					40F1F7A0	F4F537A0	

Figure 62. Sample Test Mode Execution Snaps (Page 3 of 3)

DATE	THREAD	MSGLEN	TIME	QPR	RSC	SSC	MMN	DATE	TIME	TID	PID	CO	USR	BMN	LOG	BLK	VMI	PAGE
83-032	0	02	17-45-3149	02	0000	0000	1	83-032	17-45-3149	TEST1	0001	00	1	9F	00	00	1
000000	C905E3C5	D9C3D6DA	D440E2E3	C1D9E3E4	D740DAC5	F2E2C1C7	C5A060A0	E6E2C4C3	*INTERCOMM STARTUP MESSAGE - WSDC*									
000032	E5C6E3E3								*VHTT									
000000	0	02	17-45-3149	02	0000	0000	1	83-032	17-45-3149	TEST1	0001	00	1	01	00	00	1
000000	D9E3D9D8	15D761D5	40F1F2F3	F4F515E6	C8E240F2	F0F037			*ATRO.P/N 12345.WHS 200.									
000000	0	02	17-45-3150	02	0000	0000	2	83-032	17-45-3150	TFST1	0001	00	2	01	00	00	2
000000	D9E3D9D8	15D761D5	40F5F5F5	F5F515E6	C8E240F2	F0F037			*ATRO.P/N 55555.WHS 200.									
000000	0	02	17-45-3153	02	0000	0000	3	83-032	17-45-3153	TEST1	0001	00	3	01	00	00	3
000000	D9E3D9D8	15D761D5	40F1F2F3	F4F515E6	C8E240F3	F0F037			*ATRO.P/N 12345.WHS 300.									
000000	0	02	17-45-3153	02	0000	0000	4	83-032	17-45-3153	TEST1	0001	00	4	01	00	00	4
000000	D9E3D9D8	15D761D5	40F1F2F3	F4F915E6	C8E240F2	F0F037			*ATRO.P/N 12349.WHS 200.									
000000	0	02	17-45-3154	02	0000	0000	5	83-032	17-45-3154	TEST1	0001	00	5	01	00	00	5
000000	D9E3D9D8	15D761D5	40F1F2F3	F4F115E6	C8E240F1	F0F037			*ATRO.P/N 12341.WHS 100.									
000000	0	02	17-45-3154	02	0000	0000	6	83-032	17-45-3154	TEST1	0001	00	6	01	00	00	6
000000	D9E3D9D8	15D761D5	40C1F2F3	F4F515E6	C8E240F4	F0F037			*ATRO.P/N A2345.WHS 400.									
000000	0	02	17-45-5365	02	0000	0000	7	83-032	17-45-5365	TOALL	0001	00	0	01	00	50	7
000000	FF0202D	013C5C5C	5C40C7D6	D6C4A0C1	C6E3C5D9	D5D6D6D5	5C5C40A0	C9D5E3C5	*GOOD AFTERNOON** INTE*									
000032	D9C3D6DA	D440C9E2	40D9C5C1	C4E8A07A	40A040F0	F260F0F1	60F8F340	40F1F74B	*RCOMM IS READY : 02-01-83 17.45									
000064	F4F5								*45									
000000	0	02	17-45-5380	02	0000	0000	7	83-032	17-45-5380	TOALL	0000	00	0	30	00	50	8
000000	5C5C5C40	C7D6D6C4	40C1C6E3	C5D9D5D6	D6D55C5C	40A0C9D5	E3C5D9C3	D6D4D440	*GOOD AFTERNOON** INTERCOMM *									
000032	C9E240D9	C5C1C4E8	407A4040	40F0F260	F0F160F8	F34040F1	F74BF4F5	2637	*IS READY : 02-01-83 17.45**									
000000	0	02	17-45-5398	02	0000	0000	7	83-032	17-45-5398	TEST1	0001	00	0	40	00	50	9
000000	5C5C5C40	C7D6D6C4	40C1C6E3	C5D9D5D6	D6D55C5C	40A0C9D5	E3C5D9C3	D6D4D440	*GOOD AFTERNOON** INTERCOMM *									
000032	C9E240D9	C5C1C4E8	407A4040	40F0F260	F0F160F8	F34040F1	F74BF4F5	37	*IS READY : 02-01-83 17.45**									
000000	0	02	17-45-5398	02	0000	0000	7	83-032	17-45-5398	TOALL	0000	00	0	FA	00	50	10
000032	00000000	00000000	00180000	0F500000	00070000	00000000	00020000	00000000	*GOOD AFTERNOON** INTERCOMM *									
000000	0	02	17-45-5398	02	0000	0000	7	83-032	17-45-5398	TOALL	0000	00	0	FA	00	50	11
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*GOOD AFTERNOON** INTERCOMM *									
000000	0	02	17-45-5412	02	0000	0000	1	83-032	17-45-5412	TEST1	0000	00	1	30	00	00	12
000000	E2E2D9D8	F0F0F0F1	F040A040	F1F2F3F4	F540F161	F240C9D5	40E2E3C5	C5D340E6	*SSR000010 12345 1/2 IN STEEL V*									
000032	C1E2C8C5	D9A04040	40A0A040	40A0A040	40A0A040	40A0A040	40A0A040	40A0A040	*ASHER GRS 305.0507 200MIAMI, *									
000064	40A0A040	40A0A040	40A0A040	40A0A040	40A0A040	40A0A040	40A0A040	40A0A040	*FLA. 6.161.50603/05/8*									
000096	C6D3C1A8	40A0A040	40A0A040	40A0A040	F660F1F6	F168F5F0	F6F0F361	F0F561F8	*24.040.61710/11/82									
000128	F2F468F0	F4F068F6	F1F7F1F0	61F1F161	FRF24040	4040			*24.040.61710/11/82									
000000	0	02	17-45-5412	02	0000	0000	1	83-032	17-45-5412	TEST1	0000	00	1	FA	00	00	13
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*GOOD AFTERNOON** INTERCOMM *									

Figure 63. Test Mode Execution Log Printout (Page 1 of 6)

DATE	TIME	TYPE	18.01.27	*****	INTERCOM	LOG	DISPLAY	*****	PAGE						
MSGLEN	THREAD	QPR	RSC	SSC	MMN	DATE	TIME	TID	PID	CO	USR	BMN	LOG	BLK	VMI
000000	00000000	00010000	000E0000	0A100000	00190000	00010000	00000000	00010000	000000	000000	000000	000000	000000	000000	000000
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000000	000000	000000	000000	000000	000000	000000
42	1	02	.H/00CR	RQ/D9DR	8	83.032	17.45.5412	TEST1	0000	00	1	30	00	72
42	2	02	RQ/D9DR	..7000	2	83.032	17.45.5421	TEST1	0000	00	2	30	00	00
200	1	02	.U/00E4	.H/00CR	9	83.032	17.45.5424	TEST1	0001	00	1	01	00	50
000000	0C05F1F2	F3F4F515	35F161F2	40C9D5A0	E2E3C5C5	D3A0E6C1	E2C8C5D9	40A0A0A0	0001	00	1	01	00	50
000032	40A0A0A0	40A0A0A0	40A0A0A0	40A0A0A0	40A0A0A0	40A0A0A0	40A0A0A0	40A01205	0001	00	1	01	00	50
000064	C7D9E2A0	40130858	F0F548F0	F5F0F708	03F20F00	0A1704C9	C104C96B	40C6D3C1	0001	00	1	01	00	50
000096	40A0A0A0	40A0A0A0	40A0A0A0	400D09F6	68F1F6F1	68F5F0F6	0E08F0F3	61F0F561	0001	00	1	01	00	50
000128	F8F20F09	F468F0F4	F068F6F1	F71008F1	F061F1F1	61F8F2FF	02006A08	0000	0001	00	1	01	00	50
88	1	02	.H/00CR	RQ/D9DR	8	83.032	17.45.5424	TEST1	0000	00	1	FA	00	72
000000	00000000	00010000	00080000	0E880000	00080000	00010000	00010000	00010000	0000	00	1	FA	00	72
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	0000	00	1	FA	00	72
42	1	02	.U/00E4	.H/00CR	9	83.032	17.45.5424	TEST1	0000	00	1	30	00	50
297	1	02	.U/00E4	.H/00CR	9	83.032	17.45.5425	TEST1	0001	00	1	40	00	50
000000	40A0A0A0	40E2E3D6	C3D2A0E2	E3C1E3E4	E2A0D9C5	08E4C5E2	E315D7C1	D9E3A0D5	0001	00	1	40	00	50
000032	E4D4C2C5	D940F1F2	F3F4F515	C4C5E2E3	D9C9D7E3	C9D6D5A0	F161F2A0	C9D5A0E2	0001	00	1	40	00	50
000064	E3C5C5D3	40E6C1F2	C8C5D915	D6D9C4C5	D940E4D5	C9E3E2A8	C7D9E2A0	40A0D7D9	0001	00	1	40	00	50
000096	C9C3C5A0	58F0F5A8	F0F5F0F7	15E2E3D6	C3D2A0E2	E3C1E3E4	E2A0C1E3	40E6C1D9	0001	00	1	40	00	50
000128	C5C8D6E4	E2C5A8F2	F0F15A0	40A0D3D6	C3C1E3C9	D6D5A0D4	C9C1D4C9	68A0C6D3	0001	00	1	40	00	50
000160	C1A815A0	40A0A0A0	D6D5A0C8	C1D5C4A0	40F66D0F1	F6F168F5	F0F6A0A0	40A0A0A0	0001	00	1	40	00	50
000192	40C1E2A0	D6C6A0A0	F0F361F0	F561FR2	15A0A0A0	40A0D6D5	40D6D9C4	C5D9A0F4	0001	00	1	40	00	50
000224	68F0F4F0	68F6F1F7	40A0A0A0	40A0A0C1	E2A0D6C6	40A0F1F0	61F1F161	F8F237	0001	00	1	40	00	50
88	1	02	.U/00E4	.H/00CR	9	83.032	17.45.5426	TEST1	0000	00	1	FA	00	50
000000	00000000	00000000	00080000	0E880000	000A0000	00000000	00010000	80000000	0000	00	1	FA	00	50
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	0000	00	1	FA	00	50
104	2	02	.U/00E4	RQ/D9DR	10	83.032	17.45.5432	TEST1	0001	00	2	01	00	50
000000	FF0201F5	F932D7C1	D9E3A0D5	E4D4C2C5	D940F5F5	F5F5F5A0	05D6E3A0	C6D6E4D5	0001	00	2	01	00	50
000032	C4A8A0A0	40A0A0A0	40A0A0A0	40A0A0A0	40A0A0A0	40A0A0A0	40A0A0A0	00000000	0001	00	2	01	00	50
88	2	02	RQ/D9DR	..7000	2	83.032	17.45.5432	TEST1	0000	00	2	FA	00	00
000000	00000000	00010000	00030000	08D80000	00110000	00010000	00000000	00000000	0000	00	2	FA	00	00
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	0000	00	2	FA	00	00
42	1	02	.U/00E4	RQ/D9DR	10	83.032	17.45.5433	TEST1	0000	00	2	30	00	50
82	1	02	.U/00E4	RQ/D9DR	10	83.032	17.45.5433	TEST1	0001	00	2	40	00	50
000000	5C5CC5D9	D9D6D95C	5C4A0D07	C1D9E3A0	D5E4D4C2	C5D9A0F5	F5F5F5F5	40D5D6E3	0001	00	2	40	00	50
000032	40C6D6E4	D5C4A837							0001	00	2	40	00	50
88	1	02	.U/00E4	RQ/D9DR	10	83.032	17.45.5433	TEST1	0000	00	2	FA	00	50
000000	00000000	00000000	00070000	0EE80000	000A0000	00000000	00010000	00000000	0000	00	2	FA	00	50
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	0000	00	2	FA	00	50

Figure 63. Test Mode Execution Log Printout (Page 2 of 6)

DATE	TIME	THREAD	QPR	RSC	SSC	MMN	DATE	TIME	TID	PID	CO	USR	RMN	LOG	BLK	VMI	PAGE	
42	1	02	RQ/D9D8	.. / 0000		3	83.032	17.45.5434	TEST1	0000	00	3	30	00	00	3	
104	1	02	-U/00E4	RQ/D9D8		11	83.032	17.45.5436	TEST1	0001	00	3	01	00	50	50	
000000	FF0201F5	F93207C1	D9E340F1	F2F3F4F5		40D5D6E5	40C6D6E4	D5C440C9	D540E6C1									
000032	D9C5C8D6	EAE2C5A0	F3F0F40	40A0A040		40A0A040	40A0A040	00000000	0000									
88	1	02	RQ/D9D8	.. / 0000		3	83.032	17.45.5436	TEST1	0000	00	3	FA	00	00	00	
000000	00000000	00010000	00040000	09A80000		00180000	00010000	00000000	00000000									
000032	00000000	00000000	00000000	0000														
42	1	02	-U/00F4	RQ/D9D8		11	83.032	17.45.5438	TEST1	0000	00	3	30	00	50	50	
91	1	02	-U/00E4	RQ/D9D8		11	83.032	17.45.5439	TEST1	0001	00	3	40	00	50	50	
000000	5C5CC5D9	D9D6D95C	5C4D4007	C1D9E340		F1F2F3F4	F540D5D6	E340C6D6	E405C440									
000032	C9D540E6	C1D9C5C8	D6E4E2C5	40F3F0F0		37												
88	1	02	-U/00E4	RQ/D9D8		11	83.032	17.45.5439	TEST1	0000	00	3	FA	00	50	50	
000000	00000000	00000000	00060000	0EE80000		00040000	00000000	00010000	00000000									
000032	00000000	00000000	00000000	0000														
42	1	02	RQ/D9D8	.. / 0000		4	83.032	17.45.5439	TEST1	0000	00	4	30	00	00	00	
104	1	02	-U/00E4	RQ/D9D8		12	83.032	17.45.5445	TEST1	0001	00	4	01	00	50	50	
000000	FF0201F5	F93207C1	D9E340F5	E404C2C5		D9A0F1F2	F3F4F94D	D5D6E340	C6D6E4D5									
000032	C4A8A040	40A0A040	40A0A040	40A0A040		40A0A040	40A0A040	00000000	0000									
88	1	02	RQ/D9D8	.. / 0000		4	83.032	17.45.5445	TEST1	0000	00	4	FA	00	00	00	
000000	00000000	00010000	00030000	08D00000		00110000	00010000	00000000	00000000									
000032	00000000	00000000	00000000	0000														
42	1	02	-U/00E4	RQ/D9D8		12	83.032	17.45.5445	TEST1	0000	00	4	30	00	50	50	
82	1	02	-U/00E4	RQ/D9D8		12	83.032	17.45.5445	TEST1	0001	00	4	40	00	50	50	
000000	5C5CC5D9	D9D6D95C	5C4D4007	C1D9E340		D5E4D4C2	C5D940F1	F2F3F4F9	40D5D6E3									
000032	40C6D6E4	D5C440B37																
88	1	02	-U/00E4	RQ/D9D8		12	83.032	17.45.5445	TEST1	0000	00	4	FA	00	50	50	
000000	00000000	00000000	00060000	0EE80000		00040000	00000000	00010000	00000000									
000032	00000000	00000000	00000000	0000														
42	1	02	RQ/D9D8	.. / 0000		5	83.032	17.45.5446	TEST1	0000	00	5	30	00	00	00	
192	1	02	-M/00C8	RQ/D9D8		13	83.032	17.45.5448	TEST1	0001	00	5	01	00	72	72	
000000	E2E2D9D8	F0F0F0F1	F0404040	F1F2F3F4		F140F161	F440C9D5	40C3C8D9	D6D4C540									
000032	C3C5D9D9	C1E3C5CA	40D3D6C3	D240D5E4		E3404040	40404040	40404040	40404040									
000064	40404040	40404040	D6E94040	58F1F648		F1F6F1F6	4040F1F0	F0D5C5E6	40E8D6D9									
000096	D240C3C9	E3E46840	D548E848	40404040		F56F0F5	F068F5F0	F4F0F361	F0F561F8									
000128	F2F568F0	F5F068F5	F0F4F1F0	61F1F161		F8E24040	4040											
88	1	02	RQ/D9D8	.. / 0000		5	83.032	17.45.5448	TEST1	0000	00	5	FA	00	00	00	
MSGLEN	THREAD	QPR	RSC	SSC	MMN	DATE <td>TIME</td> <td>TID</td> <td>PID</td> <td>CO</td> <td>USR</td> <td>RMN</td> <td>LOG</td> <td>BLK</td> <td>VMI</td> <td>PAGE</td>	TIME	TID	PID	CO	USR	RMN	LOG	BLK	VMI	PAGE		

Figure 63. Test Mode Execution Log Printout (Page 3 of 6)

DATE	TIME	TR.	01.27	****	INTER	COM	LOG	DIS	PLAY	****	PAGE				
MSGLEN	THREAD	OPR	RSC	SSC	MHN	DATE	TIME	TID	PID	CO	USR	RMN	LOG	BLK	VMI
000000	00000000	00010000	00040000	09A80000	00170000	00010000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
42	1	02	..N/00C8	RG/D9D8	13	83.032	17.45.5448	TEST1	0000	0000	00	5	30	00	72
42	2	02	RG/D9D8	..Y/0000	6	83.032	17.45.5449	TEST1	0000	0000	00	6	30	00	00
77	2	02	..N/0005	..Y/00E8	14	83.032	17.45.5449	TEST1	0000	0001	00	6	01	00	50
000000	0505C1F2	F3F4F504	03D761D5	0304D9E3	D9D80105	E3C5E2E3	F1FF0200	1D0204F0	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000032	F0F0F6								00000000	00000000	00000000	00000000	00000000	00000000	00000000
70	2	02	..N/0005	..Y/00E8	15	83.032	17.45.5449	TEST1	0000	0001	00	6	01	00	50
000000	0403D761	D50304D9	E3D9D801	05E3C5E2	E3F1FF02	00160204	F0F0F0F6		00000000	00000000	00000000	00000000	00000000	00000000	00000000
000032									00000000	00000000	00000000	00000000	00000000	00000000	00000000
88	2	02	RG/D9D8	..Y/0000	6	83.032	17.45.5449	TEST1	0000	0004	00	6	FA	00	00
000000	00000000	00000000	00010000	06F80000	00060000	00020000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
42	2	02	..N/0005	..Y/00E8	14	83.032	17.45.5449	TEST1	0000	0000	00	5	30	00	50
42	3	02	..N/0005	..Y/00E8	15	83.032	17.45.5462	TEST1	0000	0000	00	6	30	00	50
200	1	02	..U/00E4	..H/00C8	16	83.032	17.45.5464	TEST1	0000	0001	00	5	01	00	50
000000	0C05F1F2	F3F4F115	35F161F4	40C9D540	C3C8D9D6	D4C540C3	C5D9D9C1	E3C5C440	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000032	D3D6C3D2	40D5E4E3	40A0A040	40A0A040	40A0A040	40A0A040	40A0A040	40A01205	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000064	C4D6E9A0	4013085B	F1F648F1	F6F1F608	03F1F0F0	0A17D5C5	F6A0E8D6	D9D2A0C3	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000096	C9E3C868	40D548E8	48A0A040	4000D09F5	68F0F5F0	68F5F0F4	0C08F0F3	61F0F561	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000128	F8F20F09	F568F0F5	F068F5F0	F41008F1	F061F1F1	61F8F2FF	02006400	0000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
88	1	02	..H/00C8	RG/D9D8	13	83.032	17.45.5464	TEST1	0000	0000	00	5	FA	00	72
000000	00000000	00010000	00040000	0E000000	00080000	00010000	00010000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
42	1	02	..U/00E4	..H/00C8	16	83.032	17.45.5464	TEST1	0000	0000	00	5	30	00	50
181	2	02	..N/0005	..Y/00E8	14	83.032	17.45.5466	TEST1	0000	0001	00	6	40	00	50
000000	F0F0C5F8	40F0F0F0	F2F915D5	D6D560D5	E4D4C5D9	C9C340C3	C8C1D9C1	C3E3C5D9	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000032	40C7C9E5	C5D540D6	D540D761	D540D7C1	D9C1D4C5	E3C5D940	C6D6D940	D9E3D9D8	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000064	40E5C5D9	C248A0C1	D3D340C3	C8C1D9C1	C3E3C5D9	E240E2C8	D6E4D3C4	40C2C540	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000096	D5E4D4C5	D9C9C348	I340D4C5	E2E2C1C7	C540D5D6	48A0F0F0	F0F640C6	D9D6D440	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000128	E3D7E4A0	E3C5E2E3	F14837						00000000	00000000	00000000	00000000	00000000	00000000	00000000
88	2	02	..N/0005	..Y/00E8	14	83.032	17.45.5466	TEST1	0000	0000	00	6	FA	00	50
000000	00000000	00010000	00120000	15D00000	000C0000	00000000	00010000	00010000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000032	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
178	3	02	..N/0005	..Y/00E8	15	83.032	17.45.5473	TEST1	0000	0001	00	6	40	00	50
000000	F0F0C5F8	40F0F0F0	F2F215D9	C5D0E4C9	D9C5C440	D7C1D9C1	D4C5E3C5	D940D761	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000032	D540E6C1	E240D6D4	C9E3E3C5	C44DD6D9	40C7C9E5	C5D540C9	D4540C5D9	D9D6D9D0	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000064	D6D540E3	C8C54040	D9E3D9D8	40E5C5D9	C248A0E5	C8C54040	E6C1E240	C3C1D5C3	00000000	00000000	00000000	00000000	00000000	00000000	00000000
000096	C5D3D3C5	C41540D4	C5E2E2C1	C7C540D5	D648A0F0	F0F0F640	C6D9D6D4	40E3D7E4	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Figure 63. Test Mode Execution Log Printout (Page 4 of 6)


```

DATE R3.032    TIME 1R.01.27    **** I N T E R C O M M    L O G    D I S P L A Y ****    PAGE    6
MSGLEN THREAD GPR RSC    SSC    MMN    DATE    TIME    TID    PID    CO    USR    RMN    LOG    BLK    VMT
-----
42  1  02  .J/0001  .. /0000  19  83.032  17.46.0003  CNT01  ..... 0000  00  0  30  00  FC
7R  0  00  .. /0000  .. /0000  0  83.032  17.46.000R  ..... 0000  00  0  AA  00  00
000000 C9D5E3C5 09C30604 D440C3D3 06E2C5C4  06E6D590 04C5E2E2 C1C7C590 E6E2C4C3  *INTERCOMM CLOSEDOWN MESSAGE *SUDC*
000032 E5C8E3E3  *VHTT
    
```

Figure 63. Test Mode Execution Log Printout (Page 6 of 6)

Chapter 13

VS COBOL II TESTING

This chapter illustrates testing of a VS COBOL II subsystem under Release 10 and using the BTAM simulator, as previously described in Chapter 11.

The VS COBOL II subsystem, SQCOBOLV (see Figure 64) is a reorganized version of SQCOBOLA (see Figure 48 in Chapter 10), in that all the DWS fields (except the output message header area) have been moved to, and reorganized in, the VS COBOL II subsystem's Working-Storage Section. The one-character 77 level fields were changed to literals in program MOVE instructions, and the field areas were organized into 01 level groups. Note that the COBOL COPY members are at the post SM level 2240 of Release 10 (see Appendix B).

To test this subsystem, the same MMU map definitions (see Figure 51), symbolic MMU maps (copied directly into Working-Storage), and test input messages (see Figure 52) are used. However, the STRT\$DWSCK command is omitted as DWS overflow checking is not needed. Also the (recompiled under VS COBOL II) subroutine SQCOBOLB (see Figure 49 in Chapter 10) is called via COBREENT, as before. The test setup described in Section 11.3 is the same except that: Figure 65 replaces the table additions illustrated in Figure 50 (note that there are no COBOL II parameters to add, the only changes are the subsystem name and its DWS size to 48); and Figure 66 replaces Figure 53 to illustrate execution JCL in the VS COBOL II environment. The SIM3270 output printout from the test would look the same as in Figure 54. A Release 10 log printout is provided in Figure 67.

```

PP 5668-958 IBM VS COBOL II Release 3.0 09/13/88          SQCBOBLV Date 05/26/94
LineID  PL SL  -----*A-1-8-----2-----3-----4-----5-----6-----7-----
000001**          IDENTIFICATION DIVISION.
000002**          PROGRAM-ID. SQCBOBLV.
000003**          ENVIRONMENT DIVISION.
000004**          DATA DIVISION.
000005**          WORKING-STORAGE SECTION.
000006**          77 DD-STOCK          PIC X(8)    VALUE 'STOKFILE'.
000007**          77 DD-PART          PIC X(8)    VALUE 'PARTFILE'.
000008**          01 IO-GROUP-NAME    PIC X(8)    VALUE 'STKSTAT'.
000009**          01 IO-MAP-NAME      PIC X(8)    VALUE 'MAP1'.
000010**          01 ERR-MAP-NAME     PIC X(8)    VALUE 'ERRMAP'.
000011**          01 MESSAGE-TABLE.
000012**              04 MSG-A          PIC X(12)   VALUE 'PART NUMBER '.
000013**              04 MSG-B          PIC X(11)   VALUE ' NOT FOUND.'.
000014**              04 MSG-C          PIC X(5)    VALUE 'PART '.
000015**              04 MSG-D          PIC X(24)   VALUE
000016**                  ' NOT FOUND IN WAREHOUSE '.
000017**              04 MSG-E          PIC X(20)   VALUE '. MESSAGE CANCELLED.'.
000018**              04 MSG-F          PIC X(17)   VALUE 'MAP ERROR MCH IS '.
000019**              04 MSG-G          PIC X(36)   VALUE
000020**                  'INVALID DATA: PARTNO MUST BE NUMERIC'.
000021**              04 MSG-H          PIC X(35)   VALUE
000022**                  'INVALID DATA: WHSNO MUST BE NUMERIC'.
000023**              04 MSG-I          PIC X(46)   VALUE
000024**                  'INVALID DATA: PARTNO AND WHSNO MUST BE NUMERIC'.
000025**          01 INVALID-INPUT-MESSAGE.
000026**              04 MSG-7          PIC X(50).
000027**          01 NO-PART-MESSAGE REDEFINES INVALID-INPUT-MESSAGE.
000028**              04 MSG-1          PIC X(12).
000029**              04 NOPART-PNO     PIC X(5).
000030**              04 MSG-2          PIC X(11).
000031**          01 NOWARES-MESSAGE REDEFINES INVALID-INPUT-MESSAGE.
000032**              04 MSG-3          PIC X(5).
000033**              04 NOWARES-PNO     PIC X(5).
000034**              04 MSG-4          PIC X(24).
000035**              04 NOWARES-WHS     PIC X(3).
000036**          01 CANCEL-MESSAGE REDEFINES INVALID-INPUT-MESSAGE.
000037**              04 CAN-CODE       PIC X(15)   JUST RIGHT.
000038**              04 CAN-FILE-NAME  PIC X(8).
000039**              04 MSG-5          PIC X(20).
000040**          01 MAPPING-ERR-MESSAGE REDEFINES INVALID-INPUT-MESSAGE.
000041**              04 MSG-6          PIC X(17).
000042**              04 ERRDR-TAG      PIC X(4).

```

Figure 64. Sample VS COBOL II Subsystem (Page 1 of 16)

```

PP 5608-958 IBM VS COBOL II Release 3.0 04/13/88          SQC080LV Date 05/26/94
LineID  PL SL  ----+---A-1-B-----2-----3-----4-----5-----6-----7-
000044**          01 RECORD-AREA.
000045**              04 PART-RECORD.
000046**          *          NOTE 100 CHARACTER HDAM RECORD WITHOUT KEYS.
000047**              06 P-REC-PART-DATA.
000048**                  08 P-REC-PIN          PIC X(5).
000049**                  08 P-REC-DES          PIC X(54).
000050**                  08 P-REC-UNT          PIC X(5).
000051**              06 P-REC-PRC          PIC 9999(4)          COMP-3.
000052**              06 P-REC-MFR-NUM      PIC X(15).
000053**              06 FILLER            PIC X(17).
000054**          04 STOCK-RECORD.
000055**          *          NOTE 80 CHARACTER VSAM RECORD.
000056**              06 DELETE-CHARACTER  PIC X.
000057**              06 S-REC-KEY-FIELD.
000058**                  08 S-REC-WMS          PIC 9(3).
000059**                  08 S-REC-PNO          PIC 9(5).
000060**              06 FILLER            PIC X(28).
000061**              06 S-REC-STOCK-DATA.
000062**                  08 S-REC-WLC          PIC X(23).
000063**                  08 S-REC-LEV          PIC 9(7)          COMP-3.
000064**          *          NOTE S-REC-LEV IS 4 CHARACTERS LONG.
000065**                  08 S-REC-LDT          PIC X(6).
000066**                  08 S-REC-ORD          PIC 9(7)          COMP-3.
000067**          *          NOTE S-REC-ORD IS 4 CHARACTERS LONG.
000068**                  08 S-REC-ODT          PIC X(6).

000070**          01 FILE-AREAS.
000071**              02 STATWD          PIC S9(7)          COMP SYNC.
000072**          *          NOTE THIS PUTS US ONTO A FULLWORD BOUNDARY ALIGNMENT.
000073**              02 FH-STATUS REDEFINES STATWD.
000074**                  04 Fm-STAT1          PIC X.
000075**                      88 IOK          VALUE '0'.
000076**                      88 IDERROR        VALUE '1'.
000077**                      88 NOT-FOUND      VALUE '2'.
000078**                      88 NO-OP         VALUE '9'.
000079**                  04 F-H-STAT2          PIC X.
000080**                  04 FILLER            PIC X(2).
000081**              02 EXTDSCT          PIC X(48).
000082**          *          NOTE WE ARE STILL ALIGNED HERE.
000083**              02 RBN-WORD          PIC S9(7)          COMP.
000084**              02 RBN-FILLER REDEFINES RBN-WORD.
000085**                  04 FILLER            PIC X.
000086**                  04 RBN              PIC X(3).
000087**              02 CURRENT-FILE      PIC X(8).

```

Figure 64. Sample VS COBOL II Subsystem (Page 2 of 16)

```

PP 5668-958 IBM VS COBOL II Release 3.0 09/13/88                SQCOBOLV Date 05/26/94
LineID  PL SL  -----+---+A-1-8---+---2---+---3---+---4---+---5---+---6---+---7-
000089**          01 SYMBOLIC-MAP.
000090**          COPY STKSTATS.
000091C*          03 MAP1.
000092C*          05 VERBF.
000093C*          06 VERBL PIC 9(4) COMP.
000094C*          06 VERBT PIC X.
000095C*          06 VERB PIC X(4).
000096C*          04 PARTNUF.
000097C*          05 PARTNOL PIC 9(4) COMP.
000098C*          05 PARTNOT PIC X.
000099C*          05 PARTNO.
000100C*          06 FILLER PIC S9(4).
000101C*          06 RBNBYTE PIC S9.
000102C*          04 USEG1.
000103C*          05 WHSNOF.
000104C*          06 WHSNOL PIC 9(4) COMP.
000105C*          06 WHSNOT PIC X.
000106C*          06 WHSNO PIC S999.
000107C*          05 PRTDATAF.
000108C*          06 PRTDATAL PIC 9(4) COMP.
000109C*          06 PRTDATAT PIC X.
000110C*          06 PRTDATA PIC X(54).
000111C*          05 ORDUNTF.
000112C*          06 ORDUNTL PIC 9(4) COMP.
000113C*          06 ORDUNT PIC X.
000114C*          06 ORDUNT PIC X(5).
000115C*          05 PRTPRCF.
000116C*          06 PRTPRCL PIC 9(4) COMP.
000117C*          06 PRTPRCT PIC X.
000118C*          06 PRTPRC PIC S999V9(4) COMP-3.
000119C*          05 WHSLOCF.
000120C*          06 WHSLOCL PIC 9(4) COMP.
000121C*          06 WHSLOCT PIC X.
000122C*          06 WHSLOC PIC X(23).
000123C*          05 STKLEVF.
000124C*          06 STKLEVL PIC 9(4) COMP.
000125C*          06 STKLEYT PIC X.
000126C*          06 STKLEY PIC S9(7) COMP-3.
000127C*          05 LEYDATEF.
000128C*          06 LEYDATEL PIC 9(4) COMP.
000129C*          06 LEYDATET PIC X.
000130C*          06 LEYDATE PIC X(8).
000131C*          05 STKORDF.
000132C*          06 STKORDL PIC 9(4) COMP.
000133C*          06 STKORUT PIC X.
000134C*          06 STKORD PIC S9(7) COMP-3.
000135C*          05 ORDDATEF.
000136C*          06 ORDDATEL PIC 9(4) COMP.
000137C*          06 ORDDATET PIC X.
000138C*          06 ORDDATE PIC X(8).
000139C*          04 FILLER PIC X(7).
000140C*          03 ERRMAP.
000141C*          05 ERRMSGF.

```

Figure 64. Sample VS COBOL II Subsystem (Page 3 of 16)

```

PP 5668-958 IBM VS COBOL II Release 3.0 04/13/88          SQCOBOLV Date 05/26/94
LineID  PL SL  ----+---*A-1-8-+-----2-----3-----4-----5-----6-----7-
000142C*                06  ERRMSG  PIC 9(4) COMP.
000143C*                06  ERRMSGT PIC X.
000144C*                06  ERRMSG  PIC X(50).
000145C*                04  FILLER  PIC X(7).

000147**                01  MMU-AREAS.
000148**                02  MCW      PIC 9(8)      COMP SYNC.
000149**                02  MCW-CODE-BYTES REDEFINES MCW.
000150**                04  MCW-RETURN-CODE PIC X.
000151**                88  MAPPING-OK      VALUE ZERO.
000152**                88  MAPEND-SUCCESSFUL VALUE '8'.
000153**                04  MCW-OPTION-2    PIC X.
000154**                04  MCW-OPTION-3    PIC X.
000155**                04  MCW-OPTION-4    PIC X.
000156**                02  MCW-CODES-PART REDEFINES MCW.
000157**                04  MCW-CODES1-2    PIC X(2).
000158**                04  FILLER      PIC X(2).
000159**                02  MCB      PIC X(48).
000160**                02  KEY-FIELD   PIC 9(8).

000162**                01  DATE-EDIT-AND-FLAGS.
000163**                02  DATE-MOLD.
000164**                04  D-E-MO      PIC X(2).
000165**                04  D-E-DAY     PIC X(2).
000166**                04  D-E-YEAR    PIC X(2).
000167**                02  DATE-MOVE.
000168**                04  D-M-MO      PIC X(2).
000169**                04  SLASH2     PIC X.
000170**                04  D-M-DAY     PIC X(2).
000171**                04  SLASH1     PIC X.
000172**                04  D-M-YEAR    PIC X(2).
000173**                02  MAP-FLAG   PIC X.
000174**                88  MAP-GOOD    VALUE 'G'.
000175**                88  MAP-ERR     VALUE 'E'.
000176**                88  MAP-OUT-ABORT VALUE 'A'.
000177**                02  FH-READ-FLAG PIC X.
000178**                88  BDAM-READ-OK VALUE 'D'.
000179**                88  VSAM-READ-OK VALUE 'V'.

```

Figure 64. Sample VS COBOL II Subsystem (Page 4 of 16)

PP 5668-958 IBM VS COBOL II Release 3.0 09/13/88		SQCOBOLV Date 05/26/94	
LineID	PL SL	A-1-B	2-3-4-5-6-7
000181**			COPY COBLOGCH.
000182C*			01 COBLOGCH.
000183C*			02 UAN PICTURE X VALUE ' '.
000184C*			02 UANMDT PICTURE X VALUE ' '.
000185C*			02 UANSEL PICTURE X VALUE ' '.
000186C*			02 UANMSEL PICTURE X VALUE ' '.
000187C*			02 UAHSEL PICTURE X VALUE ' '.
000188C*			02 UAHMSEL PICTURE X VALUE ' '.
000189C*			02 UAX PICTURE X VALUE ' '.
000190C*			02 UAXMDT PICTURE X VALUE ' '.
000191C*			02 UNN PICTURE X VALUE ' '.
000192C*			02 UNNMDT PICTURE X VALUE ' '.
000193C*			02 UNNSEL PICTURE X VALUE ' '.
000194C*			02 UNNMSEL PICTURE X VALUE ' '.
000195C*			02 UNHSEL PICTURE X VALUE ' '.
000196C*			02 UNHMSEL PICTURE X VALUE ' '.
000197C*			02 UNX PICTURE X VALUE ' '.
000198C*			02 UNXMDT PICTURE X VALUE ' '.
000199C*			02 PAN PICTURE X VALUE ' '.
000200C*			02 PANMDT PICTURE X VALUE ' '.
000201C*			02 PANSEL PICTURE X VALUE ' '.
000202C*			02 PANMSEL PICTURE X VALUE ' '.
000203C*			02 PAHSEL PICTURE X VALUE ' '.
000204C*			02 PAHMSEL PICTURE X VALUE ' '.
000205C*			02 PAX PICTURE X VALUE ' '.
000206C*			02 PAXMDT PICTURE X VALUE ' '.
000207C*			02 PSN PICTURE X VALUE ' '.
000208C*			02 PSNMDT PICTURE X VALUE ' '.
000209C*			02 PSNSEL PICTURE X VALUE ' '.
000210C*			02 PSNMSEL PICTURE X VALUE ' '.
000211C*			02 PSHSEL PICTURE X VALUE ' '.
000212C*			02 PSHMSEL PICTURE X VALUE ' '.
000213C*			02 PSX PICTURE X VALUE ' '.
000214C*			02 PSXMDT PICTURE X VALUE ' '.
000215C*			02 SUPR PICTURE X VALUE ' '.
000216C*			02 WRITEL PICTURE X VALUE ' '.
000217C*			02 ERASWRIT PICTURE X VALUE ' '.
000218C*			02 ERASWRAL PICTURE X VALUE ' '.
000219C*			02 RMDT PICTURE X VALUE ' '.
000220C*			02 RKEYBD PICTURE X VALUE ' '.
000221C*			02 RMDTKEYB PICTURE X VALUE ' '.
000222C*			02 ALARM PICTURE X VALUE ' '.
000223C*			02 ALRRMDT PICTURE X VALUE ' '.
000224C*			02 ALRRKEY PICTURE X VALUE ' '.
000225C*			02 ALRRMKY PICTURE X VALUE ' '.
000226C*			02 PRNTNL PICTURE X VALUE ' '.
000227C*			02 PRNT40 PICTURE X VALUE ' '.
000228C*			02 PKNT64 PICTURE X VALUE ' '.
000229C*			02 PRNT80 PICTURE X VALUE ' '.
000230C*			02 PRNLRMDT PICTURE X VALUE ' '.
000231C*			02 PR40RMDT PICTURE X VALUE ' '.
000232C*			02 PR64RMDT PICTURE X VALUE ' '.
000233C*			02 PR80RMDT PICTURE X VALUE ' '.

Figure 64. Sample VS COBOL II Subsystem (Page 5 of 16)

```

PP 5668-958 IBM VS COBOL II Release 3.0 04/13/86                SQCOBOLV Date 05/26/94
LineID  PL SL  ----+*A-1-B-----2-----3-----4-----5-----6-----7-
000234C*      02  PRNLRKEY PICTURE X VALUE ' '.
000235C*      02  PR4ORKEY PICTURE X VALUE ' '.
000236C*      02  PR64RKEY PICTURE X VALUE ' '.
000237C*      02  PR8ORKEY PICTURE X VALUE ' '.
000238C*      02  PRNLRMKY PICTURE X VALUE ' '.
000239C*      02  PR4ORMKY PICTURE X VALUE ' '.
000240C*      02  PR64RMKY PICTURE X VALUE ' '.
000241C*      02  PR8ORMKY PICTURE X VALUE ' '.
000242C*      02  PRNLALRM PICTURE X VALUE ' '.
000243C*      02  PR4OALRM PICTURE X VALUE ' '.
000244C*      02  PR64ALRM PICTURE X VALUE ' '.
000245C*      02  PR8OALRM PICTURE X VALUE ' '.
000246C*      02  PRNLAKMD PICTURE X VALUE ' '.
000247C*      02  PR4OARMO PICTURE X VALUE ' '.
000248C*      02  PR64ARMO PICTURE X VALUE ' '.
000249C*      02  PR8OARMO PICTURE X VALUE ' '.
000250C*      02  PRNLRKY PICTURE X VALUE ' '.
000251C*      02  PR4OARKY PICTURE X VALUE ' '.
000252C*      02  PR64ARKY PICTURE X VALUE ' '.
000253C*      02  PR8OARKY PICTURE X VALUE ' '.
000254C*      02  PRNLRMKY PICTURE X VALUE ' '.
000255C*      02  PR4OAMKY PICTURE X VALUE ' '.
000256C*      02  PR64AMKY PICTURE X VALUE ' '.
000257C*      02  PR8OAMKY PICTURE X VALUE ' '.
000258C*      *02  NULL PICTURE X VALUE ' '.
000259C*      02  NL PICTURE X VALUE ' '.
000260C*      02  FF PICTURE X VALUE ' '.
000261C*      02  CR PICTURE X VALUE ' '.
000262C*      02  SI PICTURE X VALUE ' '.

000264**      COPY ICOMSBS.
000265C*      01  REENTSBS-CODES.
000266C*      * THESE CODES REPRESENT OFFSETS FOR ROUTINE ADDRESSES IN THE
000267C*      * TABLE NAMED REENTSBS. ONLY THE MOST COMMONLY USED VALUES
000268C*      * ARE INCLUDED HERE; THE USEKS MANUAL HAS A COMPLETE LIST.
000269C*      * IF OFFSET ODD, THEN TRUE OFFSET==(OFFSET+1)
000270C*      05  INITLUG          PIC 999 COMP VALUE 103.
000271C*      05  INTSORTC         PIC 99  COMP VALUE 99.
000272C*      05  DWS-SNAP         PIC 99  COMP VALUE 95.
000273C*      05  MAPFREE         PIC 99  COMP VALUE 91.
000274C*      05  FECHRLSE       PIC 99  COMP VALUE 87.
000275C*      05  FESEND         PIC 99  COMP VALUE 83.
000276C*      05  FESENDC       PIC 99  COMP VALUE 79.
000277C*      05  DYN-ALLOCATE   PIC 99  COMP VALUE 75.
000278C*      05  DYN-ACCESS     PIC 99  COMP VALUE 71.
000279C*      05  MAPURGE        PIC 99  COMP VALUE 67.
000280C*      05  MAPCLR         PIC 99  COMP VALUE 63.
000281C*      05  MAPEND         PIC 99  COMP VALUE 59.
000282C*      05  MAPOUT        PIC 99  COMP VALUE 55.
000283C*      05  MAPIN         PIC 99  COMP VALUE 51.
000284C*      05  INTUNSTO      PIC 99  COMP VALUE 47.

```

Figure 64. Sample VS COBOL II Subsystem (Page 6 of 16)

PP 5668-958 IBM VS COBOL II Release 3.0 04/13/88		SQC080LV		Date 05/26/94				
LineID	PL SL	A-1-B	2	3	4	5	6	7
000285C*		05	INTSTORE		PIC 99	COMP	VALUE 43.	
000286C*		05	INTFETCH		PIC 99	COMP	VALUE 39.	
000287C*		05	FECMFDBK		PIC 99	COMP	VALUE 35.	
000288C*		05	FECMDDQ		PIC 99	COMP	VALUE 31.	
000289C*		05	DQ-WRITE		PIC 99	COMP	VALUE 27.	
000290C*		05	DQ-READX		PIC 99	COMP	VALUE 23.	
000291C*		05	DQ-WRITE		PIC 99	COMP	VALUE 19.	
000292C*		05	DQ-READ		PIC 99	COMP	VALUE 15.	
000293C*		05	DQ-CLOSE		PIC 99	COMP	VALUE 11.	
000294C*		05	DQ-OPEN		PIC 99	COMP	VALUE 07.	
000295C*		05	DQ-BUILD		PIC 99	COMP	VALUE 03.	
000296C*		05	FH-SELECT		PIC 99	COMP	VALUE 4.	
000297C*		05	FH-RELEASE		PIC 99	COMP	VALUE 8.	
000298C*		05	FH-READ		PIC 99	COMP	VALUE 12.	
000299C*		05	FH-WRITE		PIC 99	COMP	VALUE 16.	
000300C*		05	FH-GET		PIC 99	COMP	VALUE 20.	
000301C*		05	FH-PUT		PIC 99	COMP	VALUE 24.	
000302C*		05	FH-RELEX		PIC 99	COMP	VALUE 28.	
000303C*		05	FH-FE0V		PIC 99	COMP	VALUE 32.	
000304C*		05	TABUILD		PIC 99	COMP	VALUE 36.	
000305C*		05	TABOPEN		PIC 99	COMP	VALUE 40.	
000306C*		05	TABPUT		PIC 99	COMP	VALUE 44.	
000307C*		05	TABGET		PIC 99	COMP	VALUE 48.	
000308C*		05	TABSORT		PIC 99	COMP	VALUE 52.	
000309C*		05	TABEND		PIC 99	COMP	VALUE 56.	
000310C*		05	COBPUT		PIC 99	COMP	VALUE 68.	
000311C*		05	MSGCOL		PIC 99	COMP	VALUE 72.	
000312C*		05	COBSTORF		PIC 99	COMP	VALUE 76.	
000313C*		05	CONVERSE		PIC 99	COMP	VALUE 80.	
000314C*		05	DBINT		PIC 99	COMP	VALUE 84.	
000315C*		05	LOGPUT		PIC 99	COMP	VALUE 88.	
000316C*		05	PAGE-FILE		PIC 99	COMP	VALUE 92.	
000317C*		05	FH-GETV		PIC 99	COMP	VALUE 96.	
000318C*		05	FH-PUTV		PIC 999	COMP	VALUE 100.	
000319C*		*	CODES 104 AND UP INDICATE USER ADDITIONS TO THE TABLE					
000321**			05	SQC080LB		PIC 999	COMP VALUE 104.	
000322**			02	FILLER		PIC X(22)	VALUE	
000323**							'END OF WORKING STORAGE'.	

Figure 64. Sample VS COBOL II Subsystem (Page 7 of 16)

```

PP 5668-958 Iam VS COBOL II Release 3.0 04/13/88                SQCOBOLV Date 05/26/94
LineID  PL SL  -----+*A-1-B-----2-----3-----4-----5-----6-----7-
000325**          LINKAGE SECTION.
000326**          COPY ICOMINMG.
000327C*          01 INPUT-MESSAGE.
000328C*              04 MESSG-HDR.
000329C*                  06 MSGH-LENGTH          PIC S9999  COMP.
000330C*                  06 MSGH-QPR           PIC X.
000331C*                  06 MSGH-RSCH          PIC X.
000332C*                  06 MSGH-RSC          PIC X.
000333C*                  06 MSGH-SSC          PIC X.
000334C*                  06 MSGH-MMN          PIC XXX.
000335C*                  06 MSGH-DATE.
000336C*                      08 MSGH-YR           PIC 99.
000337C*                      08 MSGH-PERIOD        PIC X.
000338C*                      08 MSGH-JULIAN-DAY     PIC 949.
000339C*                  06 MSGH-TIME.
000340C*                      08 MSGH-HH           PIC 99.
000341C*                      08 MSGH-MM           PIC 99.
000342C*                      08 MSGH-SS           PIC 99.
000343C*                      08 MSGH-TH           PIC 99.
000344C*                  06 MSGH-TID.
000345C*                      08 MSGH-TI1          PIC X.
000346C*                      08 MSGH-TI2-3         PIC XX.
000347C*                      08 MSGH-TI4-5         PIC 99.
000348C*                  06 MSGH-FLGS          PIC X(2).
000349C*                  06 MSGH-PID           PIC X(5).
000350C*                  06 MSGH-PIDX REDEFINES MSGH-PID.
000351C*                      08 FILLER             PIC X(2).
000352C*                      08 MSGH-BMN          PIC X(3).
000353C*                  06 MSGH-SSCH          PIC X.
000354C*                  06 MSGH-ADDR          PIC X(3).
000355C*                  06 MSGH-ADRX REDEFINES MSGH-ADDR.
000356C*                      08 MSGH-USR           PIC X.
000357C*                      08 FILLER             PIC X(2).
000358C*                  06 MSGH-LDG           PIC X.
000359C*                  06 MSGH-BLK           PIC X.
000360C*                  06 MSGH-VMI           PIC X.

000362**          02 INPUT-TEXT.
000363**              04 INPUT-VERB          PIC X(4).
000364**          01 ICOM-SPA             PIC X(500).
000365**          01 ICOM-SCT             PIC X(100).
000366**          01 ICOM-RETURN          PIC S9(7) COMP.
000367**          COPY ICOMDWS.
000368C*          01 DYNAMIC-WORKING-STORAGE.
000369C*              02 OUTPUT-MESSAGE.
000370C*                  04 OMESSG-HDR.
000371C*                      06 OMSGH-LENGTH          PIC S9999  COMP.
000372C*                      06 OMSGH-QPR           PIC X.
000373C*                      06 OMSGH-RSCH          PIC X.
000374C*                      06 OMSGH-RSC          PIC X.
000375C*                      06 OMSGH-SSC          PIC X.
000376C*                      06 OMSGH-MMN          PIC XXX.
000377C*                      06 OMSGH-DATE.

```

Figure 64. Sample VS COBOL II Subsystem (Page 8 of 16)

```

PP 5668-958 IBM VS COBOL II Release 3.0 09/13/88                SQCOBOLV Date 05/26/94
LineID  PL SL  -----*A-1-B-----2-----3-----4-----5-----6-----7-----
000378C*                08  MSGH-YR                PIC 99.
000379C*                08  MSGH-PERIOD           PIC X.
000380C*                08  MSGH-JULIAN-DAY      PIC 999.
000381C*                06  MSGH-TIME.
000382C*                08  MSGH-HH                PIC 99.
000383C*                08  MSGH-MM                PIC 99.
000384C*                08  MSGH-SS                PIC 99.
000385C*                08  MSGH-TM                PIC 99.
000386C*                06  MSGH-TID.
000387C*                08  MSGH-TI1                PIC X.
000388C*                08  MSGH-TI2-3           PIC XX.
000389C*                08  MSGH-TI4-5           PIC 99.
000390C*                06  MSGH-FLGS                PIC X(2).
000391C*                06  MSGH-PID                PIC X(5).
000392C*                06  MSGH-PIDX REDEFINES MSGH-PID.
000393C*                08  FILLER                PIC X(2).
000394C*                08  MSGH-BMN                PIC X(3).
000395C*                06  MSGH-SSCH                PIC X.
000396C*                06  MSGH-ADDR                PIC X(3).
000397C*                06  MSGH-ADRX REDEFINES MSGH-ADDR.
000398C*                08  MSGH-USR                PIC X.
000399C*                08  FILLER                PIC X(2).
000400C*                06  MSGH-LOG                PIC X.
000401C*                06  MSGH-BLK                PIC X.
000402C*                06  MSGH-VMI                PIC X.

```

Figure 64. Sample VS COBOL II Subsystem (Page 9 of 16)

```

PP 5668-958 IBM VS COBOL II Release 3.0 09/13/88          SQC080LV Date 05/26/94
LineID  PL SL  -----+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
000405**          PROCEDURE DIVISION USING INPUT-MESSAGE
000406**          ICOM-SPA
000407**          ICOM-SCT
000408**          ICOM-RETURN
000409**          DYNAMIC-WORKING-STORAGE.

000411**          0100-MAIN-LINE.
000412**          PERFORM 1000-HOUSEKEEPING.
000413**          PERFORM 2000-HEADER-MOVE.
000414**          PERFORM 3000-MAP-IN.
000415**          MOVE LOW-VALUES TO VERB.
000416**          IF PARTNOT NOT EQUAL TO LOW-VALUES
000417**          OR WMSNOT NOT EQUAL TO LOW-VALUES
000418**          1      PERFORM 8900-INVALID-INPUT-RTN
000419**          ELSE
000420**          1      IF NOT MAPPING-OK
000421**          2      PERFORM 8850-MAPPING-ERR-RTN
000422**          ELSE
000423**          2      PERFORM 3500-MAP-CLEAR-RTN
000424**          2      PERFORM 4000-READ-PART-FILE
000425**          2      PERFORM 5000-FH-8DAM-READ
000426**          2      IF 8DAM-READ-OK
000427**          3      PERFORM 6000-READ-STOCK-FILE
000428**          3      PERFORM 7000-FH-VSAM-READ
000429**          3      IF VSAM-READ-OK
000430**          4      PERFORM 8000-MAP-OUT
000431**          4      IF NOT MAPPING-OK
000432**          5      PERFORM 8850-MAPPING-ERR-RTN.
000433**          IF MAP-GOOD
000434**          1      PERFORM 8500-GOOD-MAP-END
000435**          ELSE
000436**          1      IF MAP-ERR
000437**          2      PERFORM 8600-ERR-MAP-END.
000438**          GOBACK.

```

Figure 64. Sample VS COBOL II Subsystem (Page 10 of 16)

```

PP 5668-958 IBM VS COBOL II Release 3.0 09/13/88                SQCOBOLV   Date 05/26/94
LineID  PL SL  ----+--*A-1-8--+---2---+---3---+---4---+---5---+---6---+---7---
000440**          1000-HOUSEKEEPING.
000441**             MOVE +0 TO ICM-RETURN.
000442**             MOVE 'G' TO MAP-FLAG.

000444**          2000-HEADER-MOVE.
000445**             MOVE MESSG-HDR TO OMESSG-HDR.

000447**          3000-MAP-IN.
000448**             MOVE SPACES TO MCW-CODE-BYTES.
000449**             CALL 'COBREENT' USING MAPIN
000450**                                     MCB
000451**                                     IO-GROUP-NAME
000452**                                     IO-MAP-NAME
000453**                                     INPUT-MESSAGE
000454**                                     MCW
000455**                                     MAP1.

000457**          3500-MAP-CLEAR-RTN.
000458**             MOVE SPACES TO MCW-CODE-BYTES.
000459**             MOVE 'A' TO MCW-OPTION-4.
000460**             CALL 'COBREENT' USING MAPCLR
000461**                                     MCB
000462**                                     IO-GROUP-NAME
000463**                                     IO-MAP-NAME
000464**                                     MAP1
000465**                                     OMSGH-TID.

000467**          4000-READ-PART-FILE.
000468**             MOVE RBNBYTE TO RBN-WORD.
000469**             MOVE DD-PART TO CURRENT-FILE.

000471**          5000-FH-BDAM-READ.
000472**             CALL 'COBREENT' USING SQCOBOLB
000473**                                     PART-RECORD
000474**                                     RBN.
000475**             IF ICM-RETURN EQUAL 1
000476**                 1          PERFORM 9600-IO-ERROR-ROUTINE
000477**             ELSE
000478**                 1          IF ICM-RETURN EQUAL 2
000479**                 2          PERFORM 9700-NOT-FOUND-RTN
000480**             ELSE
000481**                 2          IF ICM-RETURN EQUAL 9
000482**                 3          PERFORM 9500-NO-DD-ROUTINE
000483**             ELSE
000484**                 3          IF P-REC-PIN NOT EQUAL PARTNO
000485**                 4          PERFORM 9700-NOT-FOUND-RTN
000486**             ELSE
000487**                 4          MOVE P-REC-OES TO PRDData
000488**                 4          MOVE P-REC-UNT TO ORDUNT
000489**                 4          MOVE P-REC-PRC TO PRTPRC
000490**                 4          MOVE 'D' TO FH-READ-FLAG.

```

Figure 64. Sample VS COBOL II Subsystem (Page 11 of 16)

```

PP 5668-958 IBM VS COBOL II Release 3.0 09/13/88          SQCOBOLV Date 05/26/94
LineID  PL SL  -----A-1-B-----2-----3-----4-----5-----6-----7-----
000492**          6000-READ-STOCK-FILE.
000493**          MOVE DD-STOCK TO CURRENT-FILE.
000494**          MOVE WHSNO TO S-REC-WHS.
000495**          MOVE PARTNO TO S-REC-PNO.
000496**          MOVE S-REC-KEY-FIELD TO KEY-FIELD.

000498**          7000-FH-VSAM-READ.
000499**          MOVE ZERO TO FH-READ-FLAG.
000500**          MOVE LOW-VALUES TO EXTDSCT.
000501**          PERFORM 7400-FH-SELECT-ROUTINE.
000502**          IF NO-OD
000503**          1     PERFORM 9500-NO-OD-ROUTINE
000504**          ELSE
000505**          1     PERFORM 7200-FH-VSAM-READ-CONTINUE
000506**          1     IF IUERROR
000507**          2     PERFORM 9600-ID-ERROR-ROUTINE
000508**          1     ELSE
000509**          2     IF NOT-FOUND
000510**          3     MOVE MSG-C TO MSG-3
000511**          3     MOVE MSG-O TO MSG-4
000512**          3     MOVE PARTNO TO NOWARES-PNO
000513**          3     MOVE WHSNO TO NOWARES-WHS
000514**          3     MOVE NOWARES-MESSAGE TO EKRMMSG
000515**          3     PERFORM 9800-SEND-ERROR-MESSAGE
000516**          2     ELSE
000517**          3     MOVE 'V' TO FH-READ-FLAG
000518**          3     MOVE S-REC-WLC TO WMSLOC
000519**          3     MOVE S-REC-LEV TO STKLEV
000520**          3     MOVE S-REC-LDT TO DATE-HOLD
000521**          3     PERFORM 7300-DATE-EDITING
000522**          3     MOVE DATE-MOVE TO LEVDATE
000523**          3     MOVE S-REC-ORD TO STKORD
000524**          3     MOVE S-REC-ODT TO DATE-HOLD
000525**          3     PERFORM 7300-DATE-EDITING
000526**          3     MOVE DATE-MOVE TO ORDDATE.
000527**          PERFORM 7500-FH-RELEASE-ROUTINE.

000529**          7200-FH-VSAM-READ-CONTINUE.
000530**          MOVE SPACES TO FH-STATUS.
000531**          CALL 'COBREENT' USING FH-GETV
000532**          EXTDSCT
000533**          FH-STATUS
000534**          STOCK-RECORD
000535**          KEY-FIELD.

```

Figure 64. Sample VS COBOL II Subsystem (Page 12 of 16)


```

PP 5668-958 IBM VS COBOL II Release 3.0 09/13/88          SQCOBOLV Date 05/26/94
LineID  PL SL  -----A-1-8-----2-----3-----4-----5-----6-----7-----
000537**          7300-DATE-EDITING.
000538**             MOVE D-E-YEAR TO D-M-YEAR.
000539**             MOVE '/' TO SLASH1.
000540**             MOVE D-E-DAY TO D-M-DAY.
000541**             MOVE '/' TO SLASH2.
000542**             MOVE D-E-MO TO D-M-MO.

000544**          7400-FH-SELECT-ROUTINE.
000545**             MOVE SPACES TO FH-STATUS.
000546**             CALL 'COBREENT' USING FH-SELECT
000547**                                     EXTDSCT
000548**                                     FH-STATUS
000549**                                     CURRENT-FILE.

000551**          7500-FH-RELEASE-ROUTINE.
000552**             MOVE SPACES TO FH-STATUS.
000553**             CALL 'COBREENT' USING FH-RELEASE
000554**                                     EXTDSCT
000555**                                     FH-STATUS.

000557**          8000-MAP-OUT.
000558**             MOVE SPACES TO MCW-CODE-BYTES.
000559**             CALL 'COBREENT' USING MAPOUT
000560**                                     MCB
000561**                                     IO-GROUP-NAME
000562**                                     IO-MAP-NAME
000563**                                     MAP1
000564**                                     MCW
000565**                                     OMSGH-TID.

000567**          8500-GOOD-MAP-END.
000568**             MOVE ' Q ' TO MCW-CODE-BYTES.
000569**             PERFORM 8700-CALL-MAP-END.
000570**             IF NOT MAPEND-SUCCESSFUL
000571**                 1 PERFORM 8800-MAP-PURGE-RTN
000572**                 1 PERFORM 8850-MAPPING-ERR-RTN.

000574**          8600-ERR-MAP-END.
000575**             MOVE ' Q ' TO MCW-CODE-BYTES.
000576**             MOVE WRITE1 TO MCW-OPTION-3.
000577**             PERFORM 8700-CALL-MAP-END.
000578**             IF NOT MAPEND-SUCCESSFUL
000579**                 1 PERFORM 8800-MAP-PURGE-RTN
000580**                 1 MOVE +8 TO ICOM-RETURN.

```

Figure 64. Sample VS COBOL II Subsystem (Page 13 of 16)

```

PP 5668-958 IOM VS COBOL II Release 3.0 09/13/88                SQCOBOLV Date 05/26/94
LineID  PL SL  -----A-1-2-----3-----4-----5-----6-----7-----
000582**      8700-CALL-MAP-END.
000583**          CALL 'COBREENT' USING MAPEND
000584**                      MCB
000585**                      OUTPUT-MESSAGE
000586**                      MCW.

000588**      8800-MAP-PURGE-RTN.
000589**          CALL 'COBREENT' USING MAPPURGE
000590**                      MCB.

000592**      8850-MAPPING-ERR-RTN.
000593**          MOVE MSG-F TO MSG-6.
000594**          MOVE MCW-CODES1-2 TO ERROR-TAG.
000595**          MOVE MAPPING-ERR-MESSAGE TO ERRMSG.
000596**          PERFORM 9800-SEND-ERROR-MESSAGE.

000598**      8900-INVALID-INPUT-RTN.
000599**          IF PARTNOT NOT EQUAL TO LOW-VALUES
000600**          IF WHSNOT NOT EQUAL TO LOW-VALUES
000601**          1          MOVE MSG-I TO MSG-7
000602**          2          ELSE
000603**          1          MOVE MSG-G TO MSG-7
000604**          2          ELSE
000605**          1          IF WHSNOT NOT EQUAL TO LOW-VALUES
000606**          2          MOVE MSG-H TO MSG-7.
000607**          MOVE INVALID-INPUT-MESSAGE TO ERRMSG.
000608**          PERFORM 9800-SEND-ERROR-MESSAGE.

000610**      9500-NO-DD-ROUTINE.
000611**          MOVE MSG-E TO MSG-5.
000612**          MOVE 'NO DD FOR FILE ' TO CAN-CODE.
000613**          MOVE CURRENT-FILE TO CAN-FILE-NAME.
000614**          MOVE CANCEL-MESSAGE TO ERRMSG.
000615**          MOVE +0 TO ICOM-RETURN.
000616**          PERFORM 9800-SEND-ERROR-MESSAGE.

000618**      9600-IO-ERROR-ROUTINE.
000619**          MOVE MSG-E TO MSG-5.
000620**          MOVE 'IO ERROR ON ' TO CAN-CODE.
000621**          MOVE CURRENT-FILE TO CAN-FILE-NAME.
000622**          MOVE CANCEL-MESSAGE TO ERRMSG.
000623**          MOVE +0 TO ICOM-RETURN.
000624**          PERFORM 9800-SEND-ERROR-MESSAGE.

000626**      9700-NOT-FOUND-RTN.
000627**          MOVE +0 TO ICOM-RETURN.
000628**          MOVE MSG-A TO MSG-1.
000629**          MOVE MSG-B TO MSG-2.
000630**          MOVE PARTNO TO NOPART-PNO.
000631**          MOVE NO-PART-MESSAGE TO ERRMSG.
000632**          PERFORM 9800-SEND-ERROR-MESSAGE.

```

Figure 64. Sample VS COBOL II Subsystem (Page 14 of 16)

```

5608-958 IBM VS COBOL II Release 3.0 09/13/88                SQCOBOLV Date 05/26/94
LineID  PL SL  -----+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
000634**          9d00-SEND-ERROR-MESSAGE.
000635**          MOVE SPACES TO MCW-CODE-BYTES.
000636**          MOVE 'E' TO MAP-FLAG.
000637**          CALL 'COBREENT' USING MAPOUT
000638**                      MCB
000639**                      IO-GROUP-NAME
000640**                      ERR-MAP-NAME
000641**                      ERRMAP
000642**                      MCW
000643**                      OMSGH-TIO.
000644**          IF NOT MAPPING-OK
000645**          1      MOVE +8 TO ICOM-RETURN
000646**          1      MOVE 'A' TO MAP-FLAG.

```

Figure 64. Sample VS COBOL II Subsystem (Page 15 of 16)

S0C0B0LV Date 05/26/94 Time 14:39:38 Page 33

Source LineID	Hierarchy and Data Name	Base Locator	Hex-Displacement	Structure	Asmbif Definition	Data Type	Data Der Attributes
334	03 MSGH-MN.	BLL=0001	000	0 000 006	05 3C	Display	
335	03 MSGH-DATE	BLL=0001	004	0 000 004	05 OCL6	Group	
336	04 MSGH-YR	BLL=0001	009	0 000 009	05 2C	Disp-Num	
337	04 MSGH-PERIOD	BLL=0001	008	0 000 008	05 1C	Display	
338	04 MSGH-JULIAN-DAY	BLL=0001	00C	0 000 00C	05 3C	Disp-Num	
339	03 MSGH-TIME	BLL=0001	00F	0 000 00F	05 OCL8	Group	
340	04 MSGH-HH	BLL=0001	00F	0 000 00F	05 2C	Disp-Num	
341	04 MSGH-MM	BLL=0001	011	0 000 011	05 2C	Disp-Num	
342	04 MSGH-SS	BLL=0001	013	0 000 013	05 2C	Disp-Num	
343	04 MSGH-TH	BLL=0001	015	0 000 015	05 2C	Disp-Num	
344	03 MSGH-TID.	BLL=0001	017	0 000 017	05 OCL5	Group	
345	04 MSGH-T11.	BLL=0001	017	0 000 017	05 1C	Display	
346	04 MSGH-T12-3.	BLL=0001	018	0 000 018	05 2C	Display	
347	04 MSGH-T14-5.	BLL=0001	01A	0 000 01A	05 2C	Disp-Num	
348	03 MSGH-FLGS	BLL=0001	01C	0 000 01C	05 2C	Display	
349	03 MSGH-PID.	BLL=0001	01E	0 000 01E	05 5C	Display	
350	03 MSGH-PIDX	BLL=0001	01E	0 000 01E	05 OCL5	Group	R
351	04 FILLER.	BLL=0001	01E	0 000 01E	05 2C	Display	
352	04 MSGH-BMN	BLL=0001	020	0 000 020	05 3C	Display	
353	03 MSGH-SSCH	BLL=0001	023	0 000 023	05 1C	Display	
354	03 MSGH-ADDR	BLL=0001	024	0 000 024	05 3C	Display	
355	03 MSGH-ADRX	BLL=0001	024	0 000 024	05 OCL3	Group	
356	04 MSGH-USR.	BLL=0001	024	0 000 024	05 1C	Display	
357	04 FILLER.	BLL=0001	025	0 000 025	05 2C	Display	
358	03 MSGH-LOG.	BLL=0001	027	0 000 027	05 1C	Display	
359	03 MSGH-BL.	BLL=0001	028	0 000 028	05 1C	Display	
360	03 MSGH-YMI.	BLL=0001	029	0 000 029	05 1C	Display	
362	02 INPUT-TEXT.	BLL=0001	02A	0 000 02A	05 OCL4	Group	
363	03 INPUT-VERB.	BLL=0001	02A	0 000 02A	05 4C	Display	
364	01 ICDM-SPA.	BLL=0002	000	0 000 02A	05 500C	Display	
365	01 ICDM-SCI.	BLL=0003	000	0 000 02A	05 100C	Display	
366	01 ICDM-RETUN.	BLL=0004	000	0 000 02A	05 4C	Binary	
368	01 DYNAMIC-MARKING-STORAGE	BLL=0005	000	0 000 000	05 OCL42	Group	
369	02 OUTPUT-MESSAGE.	BLL=0005	000	0 000 000	05 OCL42	Group	
370	03 OMESSG-HDR.	BLL=0005	000	0 000 000	05 OCL42	Group	
371	04 OMSGH-LENGTH.	BLL=0005	000	0 000 000	05 2C	Binary	
372	04 OMSGH-OPR.	BLL=0005	002	0 000 002	05 1C	Display	
373	04 OMSGH-RSCH.	BLL=0005	003	0 000 003	05 1C	Display	
374	04 OMSGH-RSC.	BLL=0005	004	0 000 004	05 1C	Display	
375	04 OMSGH-SSC.	BLL=0005	005	0 000 005	05 1C	Display	
376	04 OMSGH-MN.	BLL=0005	006	0 000 006	05 3C	Display	
377	04 OMSGH-DATE.	BLL=0005	009	0 000 009	05 OCL6	Group	
378	05 OMSGH-YR.	BLL=0005	009	0 000 009	05 2C	Disp-Num	
379	05 OMSGH-PERIOD.	BLL=0005	008	0 000 008	05 1C	Display	
380	05 OMSGH-JULIAN-DAY.	BLL=0005	00C	0 000 00C	05 3C	Disp-Num	
381	04 OMSGH-TIME.	BLL=0005	00F	0 000 00F	05 OCL8	Group	
382	05 OMSGH-HH.	BLL=0005	00F	0 000 00F	05 2C	Disp-Num	
383	05 OMSGH-MM.	BLL=0005	011	0 000 011	05 2C	Disp-Num	
384	05 OMSGH-SS.	BLL=0005	013	0 000 013	05 2C	Disp-Num	
385	05 OMSGH-TH.	BLL=0005	015	0 000 015	05 2C	Disp-Num	
386	04 OMSGH-TID.	BLL=0005	017	0 000 017	05 OCL5	Group	

Figure 64. Sample VS COBOL II Subsystem (Page 16 of 16)

```

//TABLES      JOB
//*
//*           DEFINE SYCTTBL FOR SUBSYSTEM
//*
//STEP1       EXEC  LIBELINK,Q=TEST,NAME=INTSCT,LMOD=INTSCT
//LIB.SYSIN   DD    *
./ ADD NAME=USRSCTS
./ NUMBER     NEW1=100,INCR=100
USRSCTS      DS    OH
RQ           SYCTTBL SUBH=R,SUBC=Q,SBSP=SQCOBOLV,LANG=RCOB,OVLY=0,      X
              NUMCL=10,MNCL=2,TCTV=60,GET=48
/*
//ASM.SYSIN   DD    DSN=INT.SYMREL(INTSCT),DISP=SHR
//*
//*           DEFINE BTVERB FOR SUBSYSTEM
//*
//STEP2       EXEC  LIBELINK,Q=TEST,NAME=BTVRBTB,LMOD=BTVRBTB
//LIB.SYSIN   DD    *
./ ADD NAME=USRBTVRB
./ NUMBER     NEW1=100,INCR=100
USRBTVRB     DS    OH
              BTVERB VERB=MURQ,SSCH=R,SSC=Q,CONV=18000
/*
//ASM.SYSIN   DD    DSN=INT.SYMREL(BTVRBTB),DISP=SHR
//*
//*           DEFINE SUBMODS FOR SUBROUTINE
//*
//STEP3       EXEC  LIBELINK,Q=TEST,NAME=REENTSBS,LMOD=REENTSBS
//LIB.SYSIN   DD    *
./ ADD NAME=USRSUBS
./ NUMBER     NEW1=100,INCR=100
USRSUBS      DS    OH
              SUBMODS LNAME=SQCOBOLB,TYPE=COBOL,DELTIME=30,PARM=RC,      X
              GET=60
/*
//ASM.SYSIN   DD    DSN=INT.SYMREL(REENTSBS),DISP=SHR
//

```

Figure 65. Table Updates for Simulation Mode Testing

```

//EXEC TEST JOB (ICOMTEST,,20),'VS COBOL 2 TEST',CLASS=A,
//  RESTART=(GENLINK.ASM)
//PROCLIB DD DSN=INT.PROCLIB,DISP=SHR          (AS NEEDED)
//*****
//* THE RESTART PARM IN THE JOB STATEMENT RESTARTS THE TEST AT THE *
//* BEGINNING. IF YOU WISH TO RESTART AT A DIFFERENT STEP, CODE *
//* RESTART=STEPNAME OR RESTART=STEPNAME.PROCSTEPNAME *
//* *
//* NOTE: WHEN USING A VSAM FILE, IT MAY BE NECESSARY TO EXECUTE *
//* IDCAMS TO VERIFY THE FILE IF A PREVIOUS EXECUTION ABENDED. *
//*****
//*****
//* STEP GENLINK GENERATES A STANDARD BTAM FRONT END LINKEDIT DECK *
//* VIA ASSEMBLY OF THE ICOMLINK MACRO. IF ONLY A VTAM FRONT END IS *
//* USED ON-LINE, A SETGLOBE WITH THE BTAM GLOBAL SET TO 1 MUST BE *
//* IN THE LIBRARY SPECIFIED BY THE Q= PARM. ADD OR CHANGE PARMS FOR *
//* THE ICOMLINK MACRO BASED ON INTERCOMM FACILITIES USED. *
//* NOTE: COBOL2=YES GENERATES INCLUDE STATEMENTS FOR VS COBOL II *
//* SUPPORT - IGZEBST, IGZEOPT, DUMVCOB2, STVSCOB2; *
//* RECOBOL=NO SUPPRESSES INCLUDES FOR ILBO... MODULES. *
//* THE GENERATED DECK (SIMLINKV) IS PLACED ON INT.SYMTEST. *
//* NOTE: THE SPECIFIED FRONT END NETWORK TABLE (FENETWRK) THAT IS *
//* ON MODREL CONTAINS A DEFINITION FOR THE TEST TERMINAL *
//* TEST1 AS A LOCAL BTAM 3270 CRT. (COPY TO MODTEST) *
//* STEP NUM NUMBERS GENERATED LINK DECK IN INCREMENTS OF 1000 *
//* FOR ADDING INCLUDE STATEMENTS IN GENINCL STEP. *
//*****
//GENLINK EXEC ASMPC,DECK=DECK,Q=TEST
//ASM.SYSIN DD *
//          ICOMLINK MMU=YES,FETABLE=FENETWRK,RECOBOL=NO,COBOL2=YES
//          END
//SYSPUNCH DD DSN=INT.SYMTEST(SIMLINKV),DISP=SHR
//*          NUMBER GENERATED LINKEDIT DECK
//NUM          EXEC LIBE,Q=TEST
//LIB.SYSIN DD *
//  CHANGE NAME=SIMLINKV
//  NUMBER NEW1=1000,INCR=1000
//*
//*****
//* STEPS SCRSCR AND ALLOCSCR DELETE AND RE-ALLOCATE THE LOAD *
//* MODULE LIBRARY USED IN THE TEST (ALSO USED FOR DYNLLIB) *
//*****
//SCRSCR EXEC PGM=IEFBRI4
//FILE1 DD DSN=INT.MODSCR,DISP=(OLD,DELETE)
//ALLOCSCR EXEC PGM=IEFBRI4
//A DD DSN=INT.MODSCR,DISP=(,CATLG),UNIT=SYSDA,
// DCB=INT.MODREL,VOL=SER=INT001,
// SPACE=(TRK,(30,,7)) 7 RECORDS PER TRK/3380

```

Figure 66. Simulation Mode Linkedit and Execution JCL (Page 1 of 3)

```

/*****
/** STEP GENINCL CREATES INCLUDE DECK USED BY THE LINK EDIT STEP: *
/** THE ADDED INCLUDE STATEMENTS ARE FOR THE SAMPLE SUBSYSTEM *
/** (ASSUMED TO HAVE BEEN LINKED TO MODTEST), *
/** AND THE REQUIRED SIMULATION MODE MODULES. *
/** IF THE TEST1 TERMINAL IS NOT IN THE SYSTEM PMISTATB TABLE, USE: *
/** INCLUDE MODREL(PMISTATB) *
/** INCLUDE MODREL(PMIDEVTB) *
/** INCLUDE MODREL(PMIBROAD) *
/** THE ABOVE ASSUMES THE CONTROL TERMINAL IS NAMED CNT01. *
/*****
//GENINCL EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=A TO PRINT CHANGES
//SYSUT1 DD DSN=INT.SYMTST,DISP=SHR
//SYSUT2 DD DSN=&&INCL,DISP=(,PASS),UNIT=SYSDA,SPACE=(TRK,(8,1,1)),
// DCB=(BLKSIZE=80,LRECL=80)
//SYSIN DD *
./ CHANGE NAME=SIMLINKV,LIST=ALL
INCLUDE SYSLIB(SQCOBOLV) TEST SUBSYSTEM 00000010
INCLUDE SYSLIB(BTAMSIM) BTAM SIMULATOR 00000020
INCLUDE SYSLIB(SIM3270) SCREEN PRINTING 00000030
INCLUDE SYSLIB(IGZETUN) VS COBOL2 TUNING TABLE (IF NEEDED) 00000040
/*
/*****
/** LINK EDIT THE TEST INTERCOMM SYSTEM. *
/** NOTE THAT THE INTERCOMM LKEDT PROC PLACES THE LOAD MODULE ON *
/** THE MODSCR LOAD LIBRARY CREATED ABOVE. *
/*****
//LKED EXEC LKEDT,Q=TEST,LMOD=SIMICOM,
// PARM.LKED='LIST,LET,XREF,NCAL,SIZE=(250K,100K)'
//LKED.SYSLIB DD
// DD
// DD
// DD
// DD DSN=SYS1.COB2LIB,DISP=SHR (OVERRIDE SYS1.COBLIB)
//SYSIN DD DSN=&&INCL(SIMLINKV),DISP=(OLD,PASS)
//MODREL DD DSN=INT.MODREL,DISP=SHR
/*****
/** LINKEDIT THE DYNAMICALLY LOADABLE SUBROUTINE *
/** FROM MODTEST (Q= POINTS TO IT) TO MODSCR *
/*****
//LINKSQB EXEC LKEDT,Q=TEST,LMOD=SQCOBOLB
//SYSIN DD *
INCLUDE SYSLIB(SQCOBOLB)
/*
/**
/*****
/** ENSURE THE INTERCOMM VERSIONS OF VS COBOL II COBPACKS **
/** IGZCPCO AND IGZCPAC **
/** ARE ON ONE OF THE STEPLIB LIBRARIES BEFORE SYS1.COB2LIB **
/** FOR THE GO STEP (OTHERWISE RELINK THEM TO MODSCR). **
/*****

```

Figure 66. Simulation Mode Linkedit and Execution JCL (Page 2 of 3)

```

//*****
//** EXECUTE INTERCOMM IN SIMULATION MODE *
//*****
//GO EXEC PGM=SIMICOM, PARM='STARTUP', TIME=(,30)
//STEPLIB DD DSN=INT.MODSCR, DISP=(OLD,PASS)
// DD DSN=INT.MODLIB, DISP=SHR
// DD DSN=INT.MODREL, DISP=SHR
// DD DSN=SYS1.COB2LIB, DISP=SHR VS COBOL II LOAD LIBRARY
//INTERLOG DD DSN=INTLOG, DISP=(NEW,PASS),
// DCB=(DSORG=PS, RECFM=VB, BLKSIZE=4096, LRECL=4092, NCP=8, OPTCD=C),
// SPACE=(TRK,(10,5)), VOL=SER=INT100, UNIT=SYSDA
//SMLOG DD SYSOUT=A, DCB=(DSORG=PS, BLKSIZE=120, RECFM=FA)
//STSLOG DD SYSOUT=A, DCB=(DSORG=PS, BLKSIZE=120, RECFM=FA)
//SYSPRINT DD SYSOUT=A, DCB=(DSORG=PS, BLKSIZE=141, LRECL=137, RECFM=VA)
//RCT000 DD DSN=INT.RCT000, DISP=SHR, DCB=(DSORG=DA, OPTCD=RF)
//PMIQUE DD DSN=INT.PMIQUE, DCB=(DSORG=DA, OPTCD=R), DISP=SHR
//BTAMQ DD DSN=INT.BTAMQ, DCB=(DSORG=DA, OPTCD=R), DISP=SHR
//INTSTOR2 DD DSN=INTSTOR2, DCB=(DSORG=DA, OPTCD=EF, LIMCT=3), DISP=SHR
//INTSTOR3 DD DSN=INTSTOR3, DCB=(DSORG=DA, OPTCD=EF, LIMCT=3), DISP=SHR
//** TEST DATA SETS FOR SAMPLE SUBSYSTEM
//STOKFILE DD DSN=VSAMSD1.STCKFILE.CLUSTER, DISP=OLD,
// AMP=(AMORG, 'RECFM=F')
//PARTFILE DD DSN=INT.BETA.PARTFILE, DISP=OLD,
// DCB=(DSORG=DA, OPTCD=R)
//** DATA SETS FOR SIMULATED TERMINAL -- TEST1
//TEST1 DD DSN=INT.TTEST1, DCB=DSORG=PS, DISP=OLD
//SCRTEST1 DD SYSOUT=A, DCB=(DSORG=PS, RECFM=FA, BLKSIZE=121)
//SIMCARDS DD *
TEST1,002
//PMISTOP DD DUMMY DELIMIT INTERCOMM FILES
//** FAR PARAMETERS
//** (TO USE, CHANGE ICOMIN TO DD *, FOLLOW WITH FARs INLINE)
//ICOMIN DD DUMMY
//** DYNAMIC LINKEDIT DATA SETS
//DYNLLIB DD DSN=INT.MODSCR, DISP=(OLD,PASS)
//DYNLPRNT DD SYSOUT=A
//**
//SNAPDD DD SYSOUT=A, SPACE=(CYL,5), FREE=CLOSE
//SYSUDUMP DD SYSOUT=A
//SYSOUT DD SYSOUT=A FOR DISPLAY VERB OUTPUT
//**
//ABNLIGNR DD DUMMY FORCE ABEND-AID TO IGNORE DUMP (PRODUCE IBM DUMP)
//*****
//** PRINT INTERCOMM LOG GENERATED BY THE TEST *
//*****
//INTERLOG EXEC PGM=LOGPRINT, COND=EVEN
//STEPLIB DD DSN=INT.MODREL, DISP=SHR
//SYSPRINT DD SYSOUT=A, DCB=(DSORG=PS, BLKSIZE=121)
//INTERLOG DD DSN=INTLOG, DISP=OLD, DCB=BLKSIZE=5000
//SYSIN DD DUMMY
//

```

Figure 66. Simulation Mode Linkedit and Execution JCL (Page 3 of 3)

DATE	TIME	15.30.24	**** I N T E R C O M M L U G D I S P L A Y ****										PAGE	
MSGLEN	THREAD	QPR	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	PMN	LOG	BLK	YMI
78	0	02	0000	0000	1	94.146	15.29.1062	TDALL	0000	00	0	0	00	00
000000	C9D5E3C5	D4C3D6D4	U440E2E3	C1D9E4E4		0740D4C5	E2E2C1C7	C5406040	C9D5E3E3					INTERCOMM STARTUP MESSAGE - INTT*
000032	F0F0F1E7													*001X
108	0	02	0000	0000	1	94.146	15.29.1062	TDALL	0000	00	0	01	00	50
000000	FF02002D	013C5C5C	5C40C7D6	D6C440C1		C6E3C5D9	D5D6D6D5	5C5C4040	C9D5E3C5					GOOD AFTERNOON** INTE*
000032	D9C3D6D4	D440C9E2	40D9C5C1	C4E8407A		404040F0	F560F2F6	60F9F440	40F1F548					*RCOMM IS READY : 05-26-94 15.*
000064	F2F9													*29
42	1	02	0000	0000	1	94.146	15.29.1062	TDALL	0000	00	0	30	00	50
42	1	02	0000	0000	1	94.146	15.29.1062	TDALL	0000	00	0	FA	00	50
108	0	02	0000	0000	2	94.146	15.29.1062	CNT01	0000	00	0	01	00	50
000000	FF02002D	013C5C5C	5C40C7D6	D6C440C1		C6E3C5D9	D5D6D6D5	5C5C4040	C9D5E3C5					GOOD AFTERNOON** INTE*
000032	D9C3D6D4	D440C9E2	40D9C5C1	C4E8407A		404040F0	F560F2F6	60F9F440	40F1F548					*RCOMM IS READY : 05-26-94 15.*
000064	F2F9													*29
42	1	02	0000	0000	2	94.146	15.29.1062	CNT01	0000	00	0	30	00	50
105	1	02	0000	0000	3	94.146	15.29.1063	CNT01	0000	00	0	F2	00	50
000000	155C5C5C	40C7D6D6	C440C1C6	E3C5D9D5		D6D6D5C5	5C40C9C9	D5E3C5D9	C3D6D4D4					GOOD AFTERNOON** INTERCOMM*
000032	40C9E240	D9C5C1C4	E8407A40	4040F0F5		60F2F660	F9F44040	FLF548F2	F91526					* IS READY : 05-26-94 15.29.*
42	1	02	0000	0000	2	94.146	15.29.1063	CNT01	0000	00	0	FA	00	50
42	1	02	0000	0000	3	94.146	15.29.1094	CNT01	0000	00	0	F3	00	50
108	0	02	0000	0000	4	94.146	15.29.1123	TEST1	0000	00	0	01	00	50
000000	FF02002D	013C5C5C	5C40C7D6	D6C440C1		C6E3C5D9	D5D6D6D5	5C5C4040	C9D5E3C5					GOOD AFTERNOON** INTE*
000032	D9C3D6D4	D440C9E2	40D9C5C1	C4E8407A		404040F0	F560F2F6	60F9F440	40F1F548					*RCOMM IS READY : 05-26-94 15.*
000064	F2F9													*29
42	1	02	0000	0000	4	94.146	15.29.1123	TEST1	0000	00	0	30	00	50
103	1	02	0000	0000	5	94.146	15.29.1124	TEST1	0000	00	0	F2	00	50
000000	5C5C5C40	C/D6D6C4	40C1C6E3	C5D9D5D6		D6D5E3C5	4040C9D5	E3C5D9C3	D6D4D4D0					GOOD AFTERNOON** INTERCOMM *
000032	C9E240D9	C5C1C4E8	407A4040	40F0F560		F2F660F9	F44040F1	F548F2F9	37					* IS READY : 05-26-94 15.29.*
42	1	02	0000	0000	4	94.146	15.29.1124	TEST1	0000	00	0	FA	00	50
42	1	02	0000	0000	5	94.146	15.29.1236	TEST1	0000	00	0	F3	00	50
67	0	F2	MM/D4D4	000000	6	94.146	15.29.1548	TEST1	0000	00	1	01	00	FF
000000	D4D4E4C3	68E2C8D6	E6B4DE2	E3D2E2E3		C1E368D4	C1D7F1D0	26						*RNUC,SHOW,(LSTKSTAT,MAP1).
42	1	F2	MM/D4D4	000000	6	94.146	15.29.1548	TEST1	0000	00	1	30	00	FF
350	1	02	0000	MM/D4D4	7	94.146	15.29.1562	TEST1	0000	00	1	F2	00	67
000000	F5C31140	401D0C140	4040401D	F0C5D5E3		C5D940E3	D9C1D5E2	C1C3E3C9	D6D540C3					*SC.A .CENTER TRANSACTION C*
000032	D6C4C511	C2F51DFO	C5D5E3C5	DY40C4C1		E3C17411	C5C51DEB	D7C1D9E3	40D5D67A					*ODE.B5.OENTER DATA.EE.YPART NO1*
MSGLEN	THREAD	QPR	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	PMN	LOG	BLK	YMI

Figure 67. Release 10 Simulation Mode Log Print (Page 1 of 6)

MSGLEN	THREAD	UPK	ASC	SSC	MMN	DATE	TIME	TID	FLGS	USR	BMN	LOG	BLK	YMI	PAGE
000064	1D5011C5	D41DF011	C6D510E8	E6C8E240	6	94.146	15.29.1562	TEST1	0000	00	1	FA	00	FF	2
000096	E306C3D2	40E2E3C1	E3E4E27A	1148D21D	6	94.146	15.29.1562	TEST1	0000	00	1	FA	00	FF	2
000128	4C3A1DF0	114CE51D	F006D9C4	C5D940E4	7	94.146	15.29.1759	TEST1	0000	00	1	F3	00	67	
000160	F0D7D9C9	C3C57A1D	40114D07	1DF0114F	8	94.146	15.29.2066	TEST1	0000	C6				FF	
000192	C1E340E6	C1D0C5C8	08E4E2C5	7A11D1E5	8	94.146	15.29.2066	TEST1	0000	00	2	01	00	FF	
000224	1DF01102	F51DF0D6	0540C5C1	05C47A1D	8	94.146	15.29.2066	TEST1	0000	00	2	30	00	FF	
000256	7A1D4011	D3E61DF0	11D4C51D	F0D6D340	8	94.146	15.29.2066	TEST1	0000	00	2	30	00	FF	
000288	1DF0C1E2	40D6C57A	1D401104	F61DF011	8	94.146	15.29.2066	TEST1	0000	00	2	30	00	FF	
000320	04E40908	6811C54F	F1F2F3F4	F511C65E	8	94.146	15.29.2066	TEST1	0000	C6				FF	
000352	1 F2	RQ/D9D8	..70000	..70000	8	94.146	15.29.2066	TEST1	0000	00	2	30	00	FF	
000384	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000416	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000448	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000480	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000512	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000544	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000576	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000608	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000640	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000672	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000704	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000736	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000768	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000800	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000832	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000864	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000896	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000928	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000960	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
000992	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001024	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001056	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001088	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001120	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001152	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001184	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001216	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001248	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001280	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001312	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001344	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001376	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001408	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001440	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001472	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001504	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001536	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001568	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001600	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001632	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001664	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001696	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001728	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001760	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001792	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001824	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001856	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001888	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001920	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001952	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
001984	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002016	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002048	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002080	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002112	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002144	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002176	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002208	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002240	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002272	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002304	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002336	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002368	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002400	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002432	1 02	..70000	MM/U4D4	..70000	9	94.146	15.29.2135	TEST1	0000	00	2	F2	00	67	
002464	1 02	..70000	MM/U4D4	..70000	9	94.146</									

DATE	TIME	THREAD	OPK	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	BMN	LOG	BLK	YMI	PAGE
000000	04E40908	6811C54F	F1F2F3F4	F811C65E	F2F0F026											3
42	1	F2	RQ/D9D8	..//0000		12	94.146	15.29.2878	TEST1	0000	00	4	30	00	FF	
411	1	02	..//0000	RQ/D9D8		13	94.146	15.29.2881	TEST1	0000	00	4	F2	00	67	
000000	F5C31140	4010C140	4040401D	F0C5D5E3												
000032	D6C4C511	C2F51D0F	C5D3E3C5	D40C4C1												
000096	1D50F1F2	F3F4F81D	F011C605	1D8E60C9												
000128	F0E2E3D6	C3D240E2	E3C1E3E4	E27A1148												
000160	D9C4C5D9	40E4D5C9	E3E27A1D	40C5C1C3												
000192	4058F5F0	F548F0F5	F0F61D0F	114FD510												
000224	40E6C1D9	C5C8D6E4	E2C57A11	D1E51D0F												
000256	D9B840C3	5D3483C	D2C7401D	F011D2F5												
000288	F5F0F71D	F011D3D6	1DFOC1E2	40D6C67A												
000320	F006D540	D6D9C4C5	D97A1D40	F5F8F5F0												
000352	1D40F1F0	61DF161	F8F21D0F	1140C113	03											
42	1	F2	RQ/D9D8	..//0000		12	94.146	15.29.2881	TEST1	0000	00	4	FA	00	FF	
42	1	02	..//0000	RQ/D9D8		13	94.146	15.29.3107	TEST1	0000	00	4	F3	00	67	
62	0	F2	RQ/D9D8	..//0000		14	94.146	15.29.3395	TEST1	0000	C6	5	01	00	FF	
000000	04E40908	6811C54F	F1F2F3F4	F111C65E	F6F0F026											
42	1	F2	RQ/D9D8	..//0000		14	94.146	15.29.3395	TEST1	0000	00	5	30	00	FF	
122	1	02	..//0000	RQ/D9D8		15	94.146	15.29.3402	TEST1	0000	00	5	F2	00	67	
000000	F1C31148	5012585F	11587F1D	E8C5D9D9												
000032	D7C1D9E3	40F1F2F3	F4F140D5	D8E340C6												
000064	C540F6F0	F03C5D68	401D0F11	5CF91303												
42	1	F2	RQ/D9D8	..//0000		14	94.146	15.29.3402	TEST1	0000	00	5	FA	00	FF	
42	1	02	..//0000	RQ/D9D8		15	94.146	15.29.3533	TEST1	0000	00	5	F3	00	67	
62	0	F2	RQ/D9D8	..//0000		16	94.146	15.29.3856	TEST1	0000	C6	6	01	00	FF	
000000	04E40908	6811C54F	F1F2F3F4	F511C65E	F2F0F026											
42	1	F2	RQ/D9D8	..//0000		16	94.146	15.29.3856	TEST1	0000	00	6	30	00	FF	
121	1	02	..//0000	RQ/D9D8		17	94.146	15.29.3857	TEST1	0000	00	6	F2	00	67	
000000	F1C31148	5012585F	11587F1D	E8C5D9D9												
000032	C9D5E3C1	D3C9C440	C4C1E3C1	7A40D7C1												
000064	C5D9C9C3	3C5D5840	1D0F11C5	F91303												
42	1	F2	RQ/D9D8	..//0000		16	94.146	15.29.3857	TEST1	0000	00	6	FA	00	FF	
42	1	02	..//0000	RQ/D9D8		17	94.146	15.29.3991	TEST1	0000	00	6	F3	00	67	
62	0	F2	RQ/D9D8	..//0000		18	94.146	15.29.4246	TEST1	0000	C6	7	01	00	FF	
MSGLEN	THREAD	OPK	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	BMN	LOG	BLK	YMI		

Figure 67. Release 10 Simulation Mode Log Print (Page 3 of 6)

DATE	TIME	15.30.24	**** I N T E R C O M M L U G D I S P L A Y ****				PAGE							
MSGLEN	THREAD	QPK	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	BMN	LUG	BLK	VMI
000000	D4E4D908	6811C54F	F1F2F3F4	F111C65E	C2F0F026					*MURQ, .E712345.F.100.				
42	1	F2	RQ/D9D8	./0000	18	94.146	15.29.4296	TEST1	0000	00		7	30	00
120	1	02	./0000	RQ/D9D8	19	94.146	15.29.4296	TEST1	0000	00		7	F2	00
000000	F1C31148	5012585F	11587F1D	E8C509D9	06D94004	C5E2E2C1	C7C57A11	5CF81DC8	*1C..6..5..5".YERRUR MESSAGE!..*8.H*					
000032	C9D5E5C1	D3C9C440	C4C1E3C1	7A40E6C8	E2D5D640	D4E4E2E3	40C2C540	D5E4D4C5	*INVALID DATA! MHSND MUST BE NUME*					
000064	D9C9C33C	506B4010	F0115CF9	1303					*RIC.), .0.*9..*					
42	1	F2	RQ/D9D8	./0000	18	94.146	15.29.4296	TEST1	0000	00		7	FA	00
42	1	02	./0000	RQ/D9D8	19	94.146	15.29.4430	TEST1	0000	00		7	F3	00
62	0	F2	RQ/D9D8	./0000	20	94.146	15.29.4663	TEST1	0000	C6		8	01	00
000000	D4E4D908	6811C54F	F1F2F3F4	E711C65E	F2F0E826				*MURQ, .E71234X.F.20Y.					
42	1	F2	RQ/D9D8	./0000	20	94.146	15.29.4663	TEST1	0000	00		8	30	00
131	1	02	./0000	RQ/D9D8	21	94.146	15.29.4663	TEST1	0000	00		8	F2	00
000000	F1C31148	5012585F	11587F1D	E8C509D9	06D94004	C5E2E2C1	C7C57A11	5CF81DC8	*1C..6..5..5".YERRUR MESSAGE!..*8.H*					
000032	C9D5E5C1	D3C9C440	C4C1E3C1	7A40D7C1	D9E3D5D6	40C1D5C4	40E6C8E2	D5D640D4	*INVALID DATA! PARTNO AND MHSND M*					
000064	E4E2E340	C2C540D5	E4D4C509	C9C34040	4040D1F0	115CF913	03		*UST BE NUMERIC .0.*9..*					
42	1	F2	RQ/D9D8	./0000	20	94.146	15.29.4663	TEST1	0000	00		8	FA	00
42	1	02	./0000	RQ/D9D8	21	94.146	15.29.4809	TEST1	0000	00		8	F3	00
62	0	F2	RQ/D9D8	./0000	22	94.146	15.29.5133	TEST1	0000	C6		9	01	00
000000	D4E4D908	6811C54F	F1F2F3F4	F911C65E	F1F0F026				*MURQ, .E712349.F.100.					
42	1	F2	RQ/D9D8	./0000	22	94.146	15.29.5134	TEST1	0000	00		9	30	00
113	1	02	./0000	RQ/D9D8	23	94.146	15.29.5137	TEST1	0000	00		9	F2	00
000000	F1C31148	5012585F	11587F1D	E8C509D9	06D94004	C5E2E2C1	C7C57A11	5CF81DC8	*1C..6..5..5".YERRUR MESSAGE!..*8.H*					
000032	D7C1D9E3	40D5E4D4	C2C50940	F1F2F3F4	F940D5D6	E340C6D6	E4D5C448	3C5D6840	*PART NUMBER 12349 NOT FOUND..*)					
000064	1DF0115C	F91303							*0.*9..*					
42	1	F2	RQ/D9D8	./0000	22	94.146	15.29.5137	TEST1	0000	00		9	FA	00
42	1	02	./0000	RQ/D9D8	23	94.146	15.29.5268	TEST1	0000	00		9	F3	00
62	0	F2	RQ/D9D8	./0000	24	94.146	15.29.5489	TEST1	0000	C6		10	01	00
000000	D4E4D908	6811C54F	F1F2F3F4	F211C65E	F1F0F026				*MURQ, .E712342.F.100.					
42	1	F2	RQ/D9D8	./0000	24	94.146	15.29.5489	TEST1	0000	00		10	30	00
430	1	02	./0000	RQ/D9D8	25	94.146	15.29.5491	TEST1	0000	00		10	F2	00
000000	F5C31140	4010C140	4040401D	F0C5D9E3	C5D940E3	D9C1D5E2	C1C3E3C9	D6U540C3	*5C..A .OENTER TRANSACTION C*					
000032	D6C4C511	C2F51D0F	C5D5E3C5	D940C4C1	E3C17A11	C5C51D68	D7C1D9E3	40D5067A	*UDE..85.OENTER DATA:EE.YPART NU*					
000064	1D50F1F2	F3F4F210	F011C6D5	1DE8E6C8	E240D5D6	7A1D50F1	F0F010F0	11C9C51D	*.812342.O.FN.YMHS NO:..E100.O.IE.*					
000096	F0E2E3D6	C3D240E2	E3C1E3E4	E27A1148	D51DFOC4	C5E2C3D9	C9D7E3C9	D6U57A1D	*O5TOCK STATUS:..N.ODESCRIPTION!.*					
MSGLEN	THREAD	QPK	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	BMN	LUG	BLK	VMI

Figure 67. Release 10 Simulation Mode Log Print (Page 4 of 6)

DATE	TIME	UPK	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	BMN	LOG	BLK	VMI
94.146	15.30.24	**** I N T E R C O M M L O G D I S P L A Y ****												
MSGLEN	THREAD	UPK	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	BMN	LOG	BLK	VMI
000128	40F361F8	40CYD>40	C603C1E3	40C8C5C1		C440C2D3	C1C3U240	C3C1D4D9	C9C1C7C5	* 3/8 IN FLAT HEAD BLACK CARRIAGE*				
000160	40C2D603	E3C4C0D9	40114C5A	1DF0114C		E51JF0D6	U9C4U5U9	40E4D5C9	E3E27A1D	* BULT-R <.0.>V.00KDER UNITS1.*				
000192	40C4D6E9	404010F0	1140C61D	F007D4C9		C3C57A1D	4028F5F0	F644F1F6	F1F61D40	* DUZ .0.(F.0PRICE: \$506.1616.0*				
000224	114FD31D	F0E2E3D6	C3D240E2	E3C1E3E4		E240C1E3	40E6C1U9	C5C8D6E4	E2C37A11	*.7N.0STOCK STATUS AT WAREHOUSE:*				
000256	D1E51D0F	D3D6C3C1	E3C9D6D5	7A1D40D4		C9C1D4C9	6840C6D3	C1483C02	C7401D0F	*JV.0LOCATTIUN: MIAMI, FLA..KG .0*				
000288	11D2F51D	FJDB0540	C8C1U5C4	7A1D40F6		F1F5F0F6	F1F41D0F	11D3D01D	F0C1E240	*.K5.0DM HANO: .6150614..0.L0.OAS *				
000320	D6C67A1D	40F0F361	F0F581F8	F21D40F1		D4C51D0F	D6U540D6	D4C4C5D9	7A1D40F5	*UJ1.0J/05/82.0.ME.0UN ORDER: .5*				
000352	F0F4F0F4	F0F41D0F	11U4E61D	F0C1E240		D6C67A1D	40F1F061	F1F161F8	F21D40F1	*040404.0.MM.0AS OF1: 10/11/82.0.*				
000384	40C11303									* A..				
42	1	F2	R0/D9D8	./0000	24	94.146	15.24.5491	TEST1	0000	00	10	FA	00	FF
42	1	02	./0000	R0/0908	25	94.146	15.24.5738	TEST1	0000	00	10	F3	00	67
47	0	00	./J/00D1	./0000	26	94.146	15.30.0053	CNT01	0000	00	0	01	00	57
000000	D509C3C4	Z6								*NRCD.				*
42	1	00	./J/00D1	./0000	26	94.146	15.30.0054	CNT01	0000	00	0	30	00	57
108	1	00	./U/00E4	./J/00D1	27	94.146	15.30.0054	TUALL	0000	00	0	01	00	50
000000	FF02002D	013C5C5C	5C40C7D6	D6C440C1		C6E3C5D9	U5D6U6D5	5C5C4040	C9D5E3C5	*.....** GOOD AFTERNOON** INTE*				
000032	D9C3D6D4	D440C9E2	40C3D3D6	E2C5C47A		404040F0	F560F2F6	60F9F440	40F1F548	*RCUMM IS CLOSED: 05-26-94 15.*				
000064	F3F0									*30				*
42	1	00	./J/00D1	./0000	26	94.146	15.30.0054	CNT01	0000	00	0	FA	00	57
42	1	00	./U/00E4	./J/00D1	27	94.146	15.30.0054	TUALL	0000	00	0	30	00	50
42	1	00	./U/00E4	./J/00D1	27	94.146	15.30.0054	TUALL	0000	00	0	FA	00	50
108	0	00	./U/00E4	./J/00D1	28	94.146	15.30.0054	CNT01	0000	00	0	01	00	50
000000	FF02002D	013C5C5C	5C40C7D6	D6C440C1		C6E3C5D9	U5D6U6D5	5C5C4040	C9D5E3C5	*.....** GOOD AFTERNOON** INTE*				
000032	D9C3D6D4	D440C9E2	40C3D3D6	E2C5C47A		404040F0	F560F2F6	60F9F440	40F1F548	*RCUMM IS CLOSED: 05-26-94 15.*				
000064	F3F0									*30				*
42	1	00	./U/00E4	./J/00D1	28	94.146	15.30.0054	CNT01	0000	00	0	30	00	50
105	1	02	./0000	./U/00E4	29	94.146	15.30.0054	CNT01	0000	00	0	F2	00	50
000000	155C5C5C	40C7D6D6	C440C1C6	E3C5D9D5		D6D6D55C	5C4040C9	D5E3C5D9	C3D6D4D4	*.** GOOD AFTERNOON** INTERCOMM*				
000032	40C9E240	C3D3D6E2	C5C47A40	4040F0F5		60F2F660	F9F44040	F1F548F3	F01526	* IS CLOSED: 05-26-94 15.30.**				
42	1	00	./U/00E4	./J/00D1	28	94.146	15.30.0055	CNT01	0000	00	0	FA	00	50
42	1	02	./0000	./U/00E4	29	94.146	15.30.0057	CNT01	0000	00	0	F3	00	50
108	0	00	./U/00E4	./J/00D1	30	94.146	15.30.0106	TEST1	0000	00	0	01	00	50
000000	FF02002D	013C5C5C	5C40C7D6	D6C440C1		C6E3C5D9	U5D6U6D5	5C5C4040	C9D5E3C5	*.....** GOOD AFTERNOON** INTE*				
000032	D9C3D6D4	D440C9E2	40C3D3D6	E2C5C47A		404040F0	F560F2F6	60F9F440	40F1F548	*RCUMM IS CLOSED: 05-26-94 15.*				
000064	F3F0									*30				*
42	1	00	./U/00E4	./J/00D1	30	94.146	15.30.0107	TEST1	0000	00	0	30	00	50
MSGLEN	THREAD	UPK	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	BMN	LOG	BLK	VMI

Figure 67. Release 10 Simulation Mode Log Print (Page 5 of 6)

DATE	TIME	LOG	DIS	PL	AY	****	PAGE							
MSGLEN	THREAD	UPK	RSC	SSC	MMN	DATE	TIME	TID	FLGS	USR	BMN	LOG	BLK	YMI
103	1	02	./0000	.U/00E4	31	94.146	15.30.0107	TEST1	0000	00	0	F2	00	50
000000	5C5C5C40	C70806C4	40C1C6E3	C309D506	D6D55C5C	4040C9D5	E3C5D4C3	D6U40440	****	6000	AFTERNOON**	INTERCOMM *		
000032	C9E240C3	D3D6E2C5	C47A4040	40F0F360	F2F660F9	F44040F1	F548F3F0	37	*IS	CLOSED:	05-26-94	15.30.		*
42	1	00	.U/00E4	.J/00D1	30	94.146	15.30.0107	TEST1	0000	00	0	FA	00	50
47	0	00	.J/00D1	./0000	32	94.146	15.30.0195	CNT01	0000	00	0	01	00	FC
000000	D5D9C3C4	2b							*NRCD.					*
42	1	00	.J/00D1	./0000	32	94.146	15.30.0195	CNT01	0000	00	0	30	00	FC
42	1	02	./0000	.U/00E4	31	94.146	15.30.0291	TEST1	0000	00	0	F3	00	50
78	0	00	./0000	./0000	0	94.146	15.30.1161	0000	00	0	AA	00	00
000000	C9D5E3C5	D4C306D4	D440C303	D8E2C5C4	D6E6D540	D4C5E2E2	C1C7C540	C9D5E3E3	*INTERCOMM	CLOSEDDOWN	MESSAGE	INTT*		*
000032	F0F0F1E7								*001X					*

Figure 67. Release 10 Simulation Mode Log Print (Page 6 of 6)



Appendix A

COBOL JCL PROCEDURES

A.1 OS/V S and ANS COBOL COMPILE AND LINK JCL

The following JCL procedures are supplied on the Intercomm release library, SYMREL. Check with your System Manager before using them to ensure they reside on your installation's system procedure library (SYS1.PROCLIB) and to verify parameters to code. When appropriate, SYSLIB references Intercomm libraries.

COBUPC:	COBOL compile
Example:	// EXEC COBUPC,Q=TEST,NAME=COBPROG

COBUPCL:	COBOL compile and linkedit for a resident program, or a dynamically loaded program which will be dynamically linkedited at Intercomm startup (the linkedit step PARM overrides AMODE=31 and RMODE=ANY cause the program to be loaded above the 16M line).
Example:	// EXEC COBUPCL,Q=TEST,NAME=COBPROG,LMOD=COBPROG // PARM.LKED='LIST,XREF,LET,NCAL,REUS,AMODE=31,RMODE=ANY'

LIBECOB:	IEBUPDTE step, followed by COBOL compilation producing object module only: add //LIB.SYSIN to specify IEBUPDTE control and change cards.
Example:	// EXEC LIBECOB,Q=TEST,NAME=COBPROG,OMOD=COBPROG

LIBECOBL:	IEBUPDTE step, followed by same JCL as COBUPCL.
Note;	LKED override parms for 31 Amode, as in COBUPCL, may be used.

Figure A-1. Intercomm-supplied COBOL JCL Procedures

Refer to the Intercomm Operating Reference Manual for further details on JCL parameter requirements, and other COBOL procedures for COBOL-F and a dynamically loaded non-reentrant (does not call COBREENT) program which is not dynamically linkedited (linked with INTLOAD).

NOTES: if not using Intercomm-supplied JCL, ensure NCAL option used on linkedit step parms:
if program written as psuedo-reentrant (uses DWS and only calls COBREENT), add REUS to linkedit parms;
NODYNAM is a required compiler option - add it to compile step PARM= parameters if the system default is DYNAM.

A.2 VS COBOL II COMPILE and LINK JCL

```

//COB2PC PROC NAME=MISSING,Q=XYZ,P=INT,L=LIB,U=USR
//COB EXEC PGM=IGYCRCTL,
// PARM=(LIST,MAP,NOFDUMP,NOTEST,RENT,XREF,'DATA(24)',RES,NODYNAM,
// 'TRUNC(BIN)',NOOBJECT,NUM,NOVBREF,LIB,'LINECOUNT(55)')
//STEPLIB DD DSN=SYS1.COB2COMP,DISP=SHR COBOL II COMPILER LIBRARY
//SYSLIB DD DSN=&P..SYM&Q,DISP=SHR USER PROGRAMS
// DD DSN=&P..SYM&U,DISP=SHR USER MAPS/COPY MEMBERS
// DD DSN=&P..SYM&L,DISP=SHR SYMLIB
// DD DSN=&P..SYMREL,DISP=SHR SYMREL
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT5 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT6 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT7 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD SYSOUT=A
//SYSLIN DD DUMMY
//SYSIN DD DSN=&P..SYM&Q.(&NAME),DISP=SHR

```

Figure A-2. VS COBOL II JCL to Compile a Program

This PROC should be placed on the Intercomm PROCLIB. To use this PROC code:

```

//COMPILE EXEC COB2PC,NAME=program,Q=program-library-suffix,
// P=Intercomm-libraries-prefix

```

Note that the P parameter may be hard-coded in the PROC (replace the value INT with the desired library-prefix-name).

```

//GOB2PCL PROC NAME=MISSING,Q=XYZ,P=INT,L=LIB,U=USR,LMOD=GO,AM=24,RM=24
//COB EXEC PGM=IGYCRCTL,
// PARM=(LIST,MAP,NOFDUMP,NOTEST,RENT,XREF,'DATA(24)',RES,NODYNAM,
// 'TRUNC(BIN)',OBJECT,NUM,NOVBREF,LIB,'LINECOUNT(55)')
//STEPLIB DD DSN=SYS1.COB2COMP,DISP=SHR COBOL II COMPILER LIBRARY
//SYSLIB DD DSN=&P..SYM&Q,DISP=SHR USER PROGRAMS
// DD DSN=&P..SYM&U,DISP=SHR USER MAPS/COPY MEMBERS
// DD DSN=&P..SYM&L,DISP=SHR SYMLIB
// DD DSN=&P..SYMREL,DISP=SHR SYMREL
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT5 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT6 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT7 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD SYSOUT=A
//SYSIN DD DSN=&P..SYM&Q.(&NAME),DISP=SHR
//SYSLIN DD DSNNAME=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(TRK,(3,3)),DCB=(BLKSIZE=80,LRECL=80,RECFM=FB)
//LKED EXEC PGM=IEWL,
// PARM='LIST,XREF,MAP,LET,NOCALL,RENT,AMODE=&AM,RMODE=&RM',
// COND=(5,LT,COB)
//SYSLIB DD DSN=SYS1.COB2LIB,DISP=SHR
// DD DSN=&P..MOD&Q,DISP=SHR
// DD DSN=&P..MODLIB,DISP=SHR
// DD DSN=&P..MODREL,DISP=SHR
//SYSLMOD DD DSN=&P..MOD&Q.(&LMOD),DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSLIN DD DSNNAME=&&LOADSET,DISP=(OLD,DELETE)
// DD DDNAME=SYSIN
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN DD DUMMY

```

Figure A-3. VS COBOL II JCL to Compile and Link a Program

This PROC should be placed on the Intercomm PROCLIB. To use this PROC code:

```

//COMPLINK EXEC COB2PCL,NAME=source-program,LMOD=load-name,
// Q=program-library-suffix,P=Intercomm-libraries-prefix

```

Note: the parameters AM and RM are used to specify to the Linkage Editor the execution time AMODE and RMODE if the program is dynamically loaded (not in Intercomm linkedit). Add AM=31,RM=ANY to the EXEC statement to cause a program to be loaded above the 16M line.

The linkedit parameters RENT, NOCALL and LET are required.



Appendix B

SOURCE STATEMENT LIBRARY COPY MEMBERS

The following members in the Intercomm SYMREL source library contain source statement code which can be inserted in a COBOL subsystem procedure simply by coding COPY member-name at the desired source line. SYMREL must be named in the DD statement concatenation for the SYSLIB data set for compilations (automatic if Intercomm-supplied COBOL procedures used).

NOTE: The block size of SYMREL is 6160 as released.

ICOMHEXC

The ICOMHEXC member defines a series of commonly used one-byte codes. Many are defined by multi-punches, hence the VALUE printed may not reflect the actual hexadecimal code; to be COPYed into Working-Storage Section.

```
01  HEX-CODES.
    05  HEX-00  PIC X VALUE ' '.
    05  CODE-00 REDEFINES HEX-00  PIC X.
    05  HEX-15  PIC X VALUE 'N'.
    05  CODE-21 REDEFINES HEX-15  PIC X.
    05  HEX-37  PIC X VALUE '7'.
    05  CODE-55 REDEFINES HEX-37  PIC X.
    05  HEX-50  PIC X VALUE '&'.
    05  CODE-80 REDEFINES HEX-50  PIC X.
    05  HEX-51  PIC X VALUE 'J'.
    05  CODE-81 REDEFINES HEX-51  PIC X.
    05  HEX-52  PIC X VALUE 'K'.
    05  CODE-82 REDEFINES HEX-52  PIC X.
    05  HEX-53  PIC X VALUE 'L'.
    05  CODE-83 REDEFINES HEX-53  PIC X.
    05  HEX-54  PIC X VALUE 'M'.
    05  CODE-84 REDEFINES HEX-54  PIC X.
    05  HEX-55  PIC X VALUE 'N'.
    05  CODE-85 REDEFINES HEX-55  PIC X.
    05  HEX-56  PIC X VALUE 'O'.
    05  CODE-86 REDEFINES HEX-56  PIC X.
    05  HEX-57  PIC X VALUE 'P'.
    05  CODE-87 REDEFINES HEX-57  PIC X.
    05  HEX-72  PIC X VALUE '2'.
    05  CODE-114 REDEFINES HEX-72  PIC X.
    05  HEX-FF  PIC X VALUE '''.
    05  CODE-225 REDEFINES HEX-FF  PIC X.
```

Figure B-1. ICOMHEXC COPY Member

ICOMINMG

The ICOMINMG member provides an input message header definition for use in the Linkage Section (first 01 level). Figures B-2, B-3, and B-4 describe various system level versions of ICOMINMG.

01	INPUT-MESSAGE.		
04	MESSG-HDR.		
06	MSGH-LENGTH	PIC S9999	COMP.
06	MSGH-QPR	PIC X.	
06	MSGH-RSCH	PIC X.	
06	MSGH-RSC	PIC X.	
06	MSGH-SSC	PIC X.	
06	MSGH-MMN	PIC XXX.	
06	MSGH-DATE.		
	08 MSGH-YR	PIC 99.	
	08 MSGH-PERIOD	PIC X.	
	08 MSGH-JULIAN-DAY	PIC 999.	
06	MSGH-TIME.		
	08 MSGH-HH	PIC 99.	
	08 MSGH-MM	PIC 99.	
	08 MSGH-SS	PIC 99.	
	08 MSGH-TH	PIC 99.	
06	MSGH-TID.		
	08 MSGH-TI1	PIC X.	
	08 MSGH-TI2-3	PIC XX.	
	08 MSGH-TI4-5	PIC 99.	
06	MSGH-FLGS	PIC X(2).	
06	MSGH-PID	PIC X(5).	
06	MSGH-PIDX REDEFINES MSGH-PID.		
	08 FILLER	PIC X(2).	
	08 MSGH-BMN	PIC X(3).	
06	MSGH-SSCH	PIC X.	
06	MSGH-ADDR	PIC X(3).	
06	MSGH-ADRX REDEFINES MSGH-ADDR.		
	08 MSGH-USR	PIC X.	
	08 FILLER	PIC X(2).	
06	MSGH-LOG	PIC X.	
06	MSGH-BLK	PIC X.	
06	MSGH-VMI	PIC X.	

Figure B-2. New Release 10 ICOMINMG COPY Member
(post SM Level 2240)

ICOMINMG con't.

01	INPUT-MESSAGE.		
04	MESSG-HDR.		
06	MSGH-LENGTH	PIC S9999	COMP.
06	MSGH-QPR	PIC X.	
06	MSGH-RSCH	PIC X.	
06	MSGH-RSC	PIC X.	
06	MSGH-SSC	PIC X.	
06	MSGH-MMN	PIC XXX.	
06	MSGH-DATE.		
	08 MSGH-YR	PIC 99.	
	08 MSGH-PERIOD	PIC X.	
	08 MSGH-JULIAN-DAY	PIC 999.	
06	MSGH-TIME.		
	08 MSGH-HH	PIC 99.	
	08 MSGH-MM	PIC 99.	
	08 MSGH-SS	PIC 99.	
	08 MSGH-TH	PIC 99.	
06	MSGH-TID.		
	08 MSGH-TI1	PIC X.	
	08 MSGH-TI2-3	PIC XX.	
	08 MSGH-TI4-5	PIC 99.	
06	MSGH-CON	PIC S9999	COMP.
06	MSGH-FLGS	PIC X(2).	
06	MSGH-BMN	PIC X(3).	
06	MSGH-SSCH	PIC X.	
06	MSGH-USR	PIC X.	
06	MSGH-ADDR	PIC XX.	
06	MSGH-LOG	PIC X.	
06	MSGH-BLK	PIC X.	
06	MSGH-VMI	PIC X.	

Figure B-3. Basic Release 10 ICOMINMG COPY Member

ICOMINMG con't.

01	INPUT-MESSAGE.		
04	MESSG-HDR.		
06	MSGH-LENGTH	PIC S9999	COMP.
06	MSGH-QPR	PIC X.	
06	MSGH-RSCH	PIC X.	
06	MSGH-RSC	PIC X.	
06	MSGH-SSC	PIC X.	
06	MSGH-MMN	PIC XXX.	
06	MSGH-DATE.		
	08 MSGH-YR	PIC 99.	
	08 MSGH-PERIOD	PIC X.	
	08 MSGH-JULIAN-DAY	PIC 999.	
06	MSGH-TIME.		
	08 MSGH-HH	PIC 99.	
	08 MSGH-MM	PIC 99.	
	08 MSGH-SS	PIC 99.	
	08 MSGH-TH	PIC 99.	
06	MSGH-TID.		
	08 MSGH-TI1	PIC X.	
	08 MSGH-TI2-3	PIC XX.	
	08 MSGH-TI4-5	PIC 99.	
06	MSGH-CON	PIC S9999	COMP.
06	MSGH-PID	PIC X(5).	
06	MSGH-SSCH	PIC X.	
06	MSGH-USR	PIC X.	
06	MSGH-BMN	PIC XX.	
06	MSGH-LOG	PIC X.	
06	MSGH-BLK	PIC X.	
06	MSGH-VMI	PIC X.	

Figure B-4. Release 9 ICOMINMG COPY Member

ICOMDWS

The ICOMDWS member provides a Dynamic-Work-Space definition with an output message header for use in the Linkage Section (fifth 01 level) of reentrant subsystems. (It can also be used in the Working-Storage Section of nonreentrant or VS COBOL II subsystems). Figures B-5, B-6, and B-7 describe various system level versions of ICOMDWS.

```

01 DYNAMIC-WORKING-STORAGE.
  02 OUTPUT-MESSAGE.
    04 OMESG-HDR.
      06 OMSGH-LENGTH          PIC S9999  COMP.
      06 OMSGH-QPR             PIC X.
      06 OMSGH-RSCH            PIC X.
      06 OMSGH-RSC             PIC X.
      06 OMSGH-SSC             PIC X.
      06 OMSGH-MMN             PIC XXX.
      06 OMSGH-DATE.
        08 OMSGH-YR             PIC 99.
        08 OMSGH-PERIOD         PIC X.
        08 OMSGH-JULIAN-DAY     PIC 999.
      06 OMSGH-TIME.
        08 OMSGH-HH             PIC 99.
        08 OMSGH-MM             PIC 99.
        08 OMSGH-SS             PIC 99.
        08 OMSGH-TH             PIC 99.
      06 OMSGH-TID.
        08 OMSGH-TI1            PIC X.
        08 OMSGH-TI2-3          PIC XX.
        08 OMSGH-TI4-5          PIC 99.
      06 OMSGH-FLGS             PIC X(2).
      06 OMSGH-PID              PIC X(5).
      06 OMSGH-PIDX REDEFINES OMSGH-PID.
        08 FILLER                PIC X(2).
        08 OMSGH-BMN             PIC X(3).
      06 OMSGH-SSCH             PIC X.
      06 OMSGH-ADDR             PIC X(3).
      06 OMSGH-ADRX REDEFINES OMSGH-ADDR.
        08 OMSGH-USR             PIC X.
        08 FILLER                PIC X(2).
      06 OMSGH-LOG              PIC X.
      06 OMSGH-BLK              PIC X.
      06 OMSGH-VMI              PIC X.

```

Figure B-5. New Release 10 ICOMDWS COPY Member
(post SM Level 2240)

ICOMDWS con't.

01	DYNAMIC-WORKING-STORAGE.		
02	OUTPUT-MESSAGE.		
04	OMESSG-HDR.		
06	MSGH-LENGTH	PIC S9999	COMP.
06	MSGH-QPR	PIC X.	
06	MSGH-RSCH	PIC X.	
06	MSGH-RSC	PIC X.	
06	MSGH-SSC	PIC X.	
06	MSGH-MMN	PIC XXX.	
06	MSGH-DATE.		
	08 MSGH-YR	PIC 99.	
	08 MSGH-PERIOD	PIC X.	
	08 MSGH-JULIAN-DAY	PIC 999.	
06	MSGH-TIME.		
	08 MSGH-HH	PIC 99.	
	08 MSGH-MM	PIC 99.	
	08 MSGH-SS	PIC 99.	
	08 MSGH-TH	PIC 99.	
06	MSGH-TID.		
	08 MSGH-TI1	PIC X.	
	08 MSGH-TI2-3	PIC XX.	
	08 MSGH-TI4-5	PIC 99.	
06	MSGH-CON	PIC S9999	COMP.
06	MSGH-FLGS	PIC X(2).	
06	MSGH-BMN	PIC X(3).	
06	MSGH-SSCH	PIC X.	
06	MSGH-USR	PIC X.	
06	MSGH-ADDR	PIC XX.	
06	MSGH-LOG	PIC X.	
06	MSGH-BLK	PIC X.	
06	MSGH-VMI	PIC X.	

Figure B-6. Basic Release 10 ICOMDWS COPY Member

ICOMDWS con't.

01	DYNAMIC-WORKING-STORAGE.		
02	OUTPUT-MESSAGE.		
04	OMESSG-HDR.		
06	OMSGH-LENGTH	PIC S9999	COMP.
06	OMSGH-QPR	PIC X.	
06	OMSGH-RSCH	PIC X.	
06	OMSGH-RSC	PIC X.	
06	OMSGH-SSC	PIC X.	
06	OMSGH-MMN	PIC XXX.	
06	OMSGH-DATE.		
	08 OMSGH-YR	PIC 99.	
	08 OMSGH-PERIOD	PIC X.	
	08 OMSGH-JULIAN-DAY	PIC 999.	
06	OMSGH-TIME.		
	08 OMSGH-HH	PIC 99.	
	08 OMSGH-MM	PIC 99.	
	08 OMSGH-SS	PIC 99.	
	08 OMSGH-TH	PIC 99.	
06	OMSGH-TID.		
	08 OMSGH-TI1	PIC X.	
	08 OMSGH-TI2-3	PIC XX.	
	08 OMSGH-TI4-5	PIC 99.	
06	OMSGH-CON	PIC S9999	COMP.
06	OMSGH-PID	PIC X(5).	
06	OMSGH-SSCH	PIC X.	
06	OMSGH-USR	PIC X.	
06	OMSGH-BMN	PIC XX.	
06	OMSGH-LOG	PIC X.	
06	OMSGH-BLK	PIC X.	
06	OMSGH-VMI	PIC X.	

Figure B-7. Release 9 ICOMDWS COPY Member

ICOMSBS

The ICOMSBS member defines the computational halfword constants used by reentrant COBOL subsystems and subroutines in calls through COBREENT to the named Intercomm system service routines and to user subroutines, if added.

```

01 REENTSBS-CODES.
* THESE CODES REPRESENT OFFSETS FOR ROUTINE ADDRESSES IN THE
* TABLE NAMED REENTSBS. ONLY THE MOST COMMONLY USED VALUES
* ARE INCLUDED HERE; THE USERS MANUAL HAS A COMPLETE LIST.
* IF OFFSET ODD, THEN TRUE OFFSET=-(OFFSET+1)
  05 INITLU6          PIC 9(4) COMP VALUE 103.
  05 INTSORTC        PIC 9(4) COMP VALUE 99.
  05 DWS-SNAP        PIC 9(4) COMP VALUE 95.
  05 MAPFREE         PIC 9(4) COMP VALUE 91.
  05 FECMRLSE        PIC 9(4) COMP VALUE 87.
  05 FESEND          PIC 9(4) COMP VALUE 83.
  05 FESENDC         PIC 9(4) COMP VALUE 79.
  05 DYN-ALLOCATE    PIC 9(4) COMP VALUE 75.
  05 DYN-ACCESS      PIC 9(4) COMP VALUE 71.
  05 MAPURGE         PIC 9(4) COMP VALUE 67.
  05 MAPCLR          PIC 9(4) COMP VALUE 63.
  05 MAPEND          PIC 9(4) COMP VALUE 59.
  05 MAPOUT          PIC 9(4) COMP VALUE 55.
  05 MAPIN           PIC 9(4) COMP VALUE 51.
  05 INTUNSTO        PIC 9(4) COMP VALUE 47.
  05 INTSTORE        PIC 9(4) COMP VALUE 43.
  05 INTFETCH        PIC 9(4) COMP VALUE 39.
  05 FECMFDBK        PIC 9(4) COMP VALUE 35.
  05 FECMDDQ         PIC 9(4) COMP VALUE 31.
  05 DQ-WRITE        PIC 9(4) COMP VALUE 27.
  05 DQ-READX        PIC 9(4) COMP VALUE 23.
  05 DQ-WRITE        PIC 9(4) COMP VALUE 19.
  05 DQ-READ         PIC 9(4) COMP VALUE 15.
  05 DQ-CLOSE        PIC 9(4) COMP VALUE 11.
  05 DQ-OPEN         PIC 9(4) COMP VALUE 07.
  05 DQ-BUILD        PIC 9(4) COMP VALUE 03.
  05 FH-SELECT       PIC 9(4) COMP VALUE 4.
  05 FH-RELEASE      PIC 9(4) COMP VALUE 8.
  05 FH-READ         PIC 9(4) COMP VALUE 12.
  05 FH-WRITE        PIC 9(4) COMP VALUE 16.
  05 FH-GET          PIC 9(4) COMP VALUE 20.
  05 FH-PUT          PIC 9(4) COMP VALUE 24.
  05 FH-RELEX        PIC 9(4) COMP VALUE 28.
  05 FH-FEOV         PIC 9(4) COMP VALUE 32.
  05 TABUILD         PIC 9(4) COMP VALUE 36.
  05 TABOPEN         PIC 9(4) COMP VALUE 40.
  05 TABPUT          PIC 9(4) COMP VALUE 44.
  05 TABGET          PIC 9(4) COMP VALUE 48.
  05 TABSORT         PIC 9(4) COMP VALUE 52.
  05 TABEND          PIC 9(4) COMP VALUE 56.

```

Figure B-8. ICOMSBS COPY Member (Page 1 of 2)

05	COBPUT	PIC 9(4)	COMP VALUE 68.
05	MSGCOL	PIC 9(4)	COMP VALUE 72.
05	COBSTORF	PIC 9(4)	COMP VALUE 76.
05	CONVERSE	PIC 9(4)	COMP VALUE 80.
05	DBINT	PIC 9(4)	COMP VALUE 84.
05	LOGPUT	PIC 9(4)	COMP VALUE 88.
05	PAGE-FILE	PIC 9(4)	COMP VALUE 92.
05	FH-GETV	PIC 9(4)	COMP VALUE 96.
05	FH-PUTV	PIC 9(4)	COMP VALUE 100.
*	CODES 104 AND UP INDICATE USER ADDITIONS TO THE TABLE		

Figure B-8. ICOMSBS COPY Member (Page 2 of 2)

NOTE: INITLU6 and TABUILD through TABEND are not supported below SM level 2241. INTSORTC and DWS-SNAP are not supported for Release 9.

Appendix C

INTERCOMM TABLE SUMMARY

Basic tables are included in the Intercomm release library (SYMREL) and must be modified (added to) for each installation. An asterisk (*) indicates optional tables which may be generated individually at each installation according to application program requirements.

TABLE or CSECT Name	Description	Created by	SYMREL and MODREL Member Name
BROADCST	*Output Broadcast Table	BCGROUP macro	PMIBROAD
BTAMSCTS	Front End Queue Table (BTAM/TCAM/GFE only)	SYCTTBL macro	BTAMSCTS
BTVRBTB	Front End Verb Table	BTVERB macro	BTVRBTB
(User-name)	Front End Network Configuration Table	LINEGRP, BLINE BTERM macros, etc. VCT, LUNIT, LCOMP macros, etc.	FENETWRK (BTSAMP) (VTSAMP)
CHNGTB	*Change Table for Change/Display Utilities	DC's	None
File Description Records (DESnnnnn)	*File Descriptions Data Set (DES000); generated by file load utility PMIEXLD (for Change/Display Utility)	FDHDR, FDETL macros	None
IXFDSCTn	File Handler Data Set Control Table	IXFDSCTA macro	IXFDSCT1 (50 DDs) IXFDSCT2 (100 DDs) IXFDSCT3 (200 DDs)
KEYTABLE	*Display Utility Key Transformation Routing Table	DC's	None
PADDTBLE	*Edit Utility Pad Table	PADD macro	PADDTBLE

Figure C-1. Table Names and Associated Macro Instructions (Page 1 of 2)

TABLE or CSECT Name	Description	Created by	SYMREL and MODREL Member Name
PAGETBL	*Page Facility Table (obsolete - Rel 10)	PAGETBL macro	PAGETBLE
PMIALTRP	*Output Utility Alternate Format Table	PMIALTRN macro	None
PMIDEVTB	Back End Device Table	DEVICE macro	PMIDEVTB
PMIFILET	*Change/Display File Table	GENFTBLE macro	PMIFILET
PMIRCNTB	*Output Utility Format Table (OFT)	CSECT	PMIRCNTB
		REPORT, LINE ITEM macros	RPTnnnnn
		PMISTOP macro	PMIRCEND
PMIRPTAB	*Output Utility Company/Report/Terminal Table	DC's	None
PMISTATB	Back End Station Table	STATION macro	PMISTATB
PTRNTBLE	*Display Utility Symbol Edit Pattern Table	PATRN macro	None
REENTSBS	Subroutine Entries List	SUBMODS macro	REENTSBS
REPTAPE	*Output Utility Batch Report Table	DC's	None
SPA/SPAEXT	System Parameter Table (SPA)	SPALIST macro	INTSPA
SCT	Subsystem Control Table (SCT)	SYCTTBL macro	INTSCT
TUNERTBL	Fine Tuner Subsystem Commands (obsolete - Rel 10)	DC's	none
VERBTBL	*Edit Control Table (ECT)	VERBGEN, VERB, PARM, PMIELIN macros	PMIVERBS

Figure C-1. Table Names and Associated Macro Instructions (Page 2 of 2)

Component Name	Tables Used
Change/Display Utility	CHNGTB File Description Records KEYTABLE PMIFILET PTRNTBLE
Edit Utility	PADDTBLE PMIFILET PMIVERBS PMIDEVTB PMISTATB
File Handler	IXFDSCTn FAR statements
Front/End TP Interface	BTVRBTB Front End Network Table BTAMSCTS (BTAM only)
Message Mapping Utilities	MMUVTBL LOGCHARS PMIDEVTB PMISTATB User-coded Maps
Monitor	REENTSBS INTSPA INTSCT BROADCAST TUNERTBL (obsolete - Rel 10)
Output Utility	PMIALTRP PMIDEVTB PMIFILET PMIRCNTB PMIRPTAB PMISTATB REPTAPE RPTnnnnn (user-coded OFTs)
Page Facility	PAGETBLE (obsolete - Rel 10)

Figure C-2. Components and Associated Table Names



Appendix D

CALLING USER SUBROUTINES FROM REENTRANT SUBSYSTEMS

COBOL programs created under Intercomm Release 8.0 (and lower) needed certain coding conventions followed for calling user-coded COBOL subroutines which themselves contain calls to COBREENT.

COBREENT saves registers, parameter lists, and other program specifications prior to calling the requested subroutine and restores these values for return to the calling program. An area of dynamic storage is therefore required by COBREENT.

There is a 256-byte save area utilized by COBREENT which is prefixed to the Dynamic-Work-Space associated with message processing for a reentrant COBOL subsystem. This area is transparent to the user. However, a subsystem which calls a COBOL subroutine in turn calling COBREENT must provide a 256-byte save area prefix as part of the passed Dynamic-Work-Space to be used by the subroutine. The address +256 of this area is passed as the fifth parameter to the called subroutine.

For example, Subsystem A may require 72 bytes of Dynamic-Work-Space for message processing. An additional 120 bytes of Dynamic-Work-Space will be utilized by Subroutine X which is called by Subsystem A via COBREENT. Subroutine X also calls COBREENT. The DWS definition in Subsystem A must include 256 bytes (aligned) to be used as a save area by COBREENT when called by Subroutine X. When Subsystem A is activated, the GET parameter of the SYCTTBL must specify 448 bytes of storage to be obtained (72 bytes for Subsystem A plus 256 bytes for COBREENT save area for Subroutine X plus 120 bytes for Subroutine X Dynamic-Work-Space). See Figures D-1 and D-2.

Figures D-3 and D-4 illustrate this example. The corresponding COBOL coding for Subsystem A is shown in Figure D-3, for Subroutine X in Figure D-4.

Subroutines called by the reentrant COBOL subsystem may not be passed any parameter defined to be longer than 4095 bytes if the subroutine in turn calls COBREENT using that same parameter. This is a COBOL compiler restriction on BLL cell addressability for passed parameters.

The called subroutine may not in turn call COBREENT to call another user COBOL subroutine; only Intercomm service routines or Assembler subroutines may be called from the subroutine. The subroutine-code of the SUBMODS entry in USRSUBS (COPY'd into REENTSBS) defining the subroutine must be added to ICOMSBS, or may be defined in the 77-level WORKING-STORAGE of the subsystem as illustrated in Figure D-3.

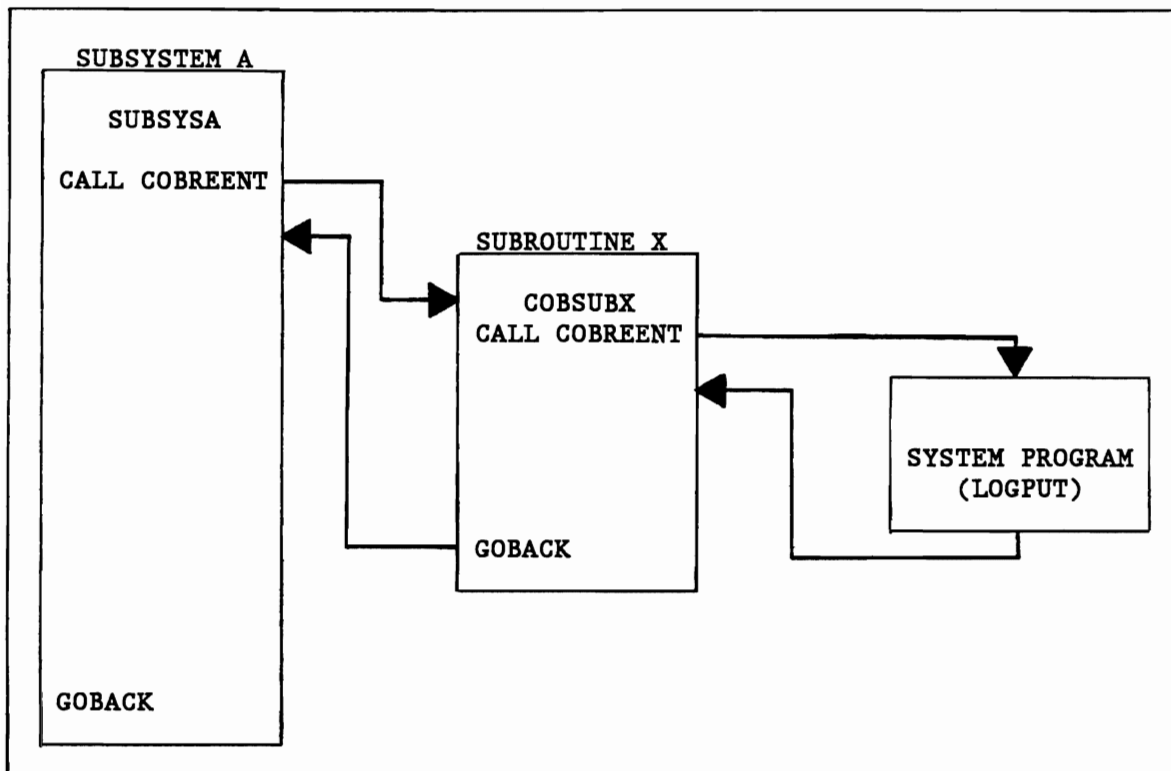


Figure D-1. COBOL Subsystem/Subroutine Nested Calls

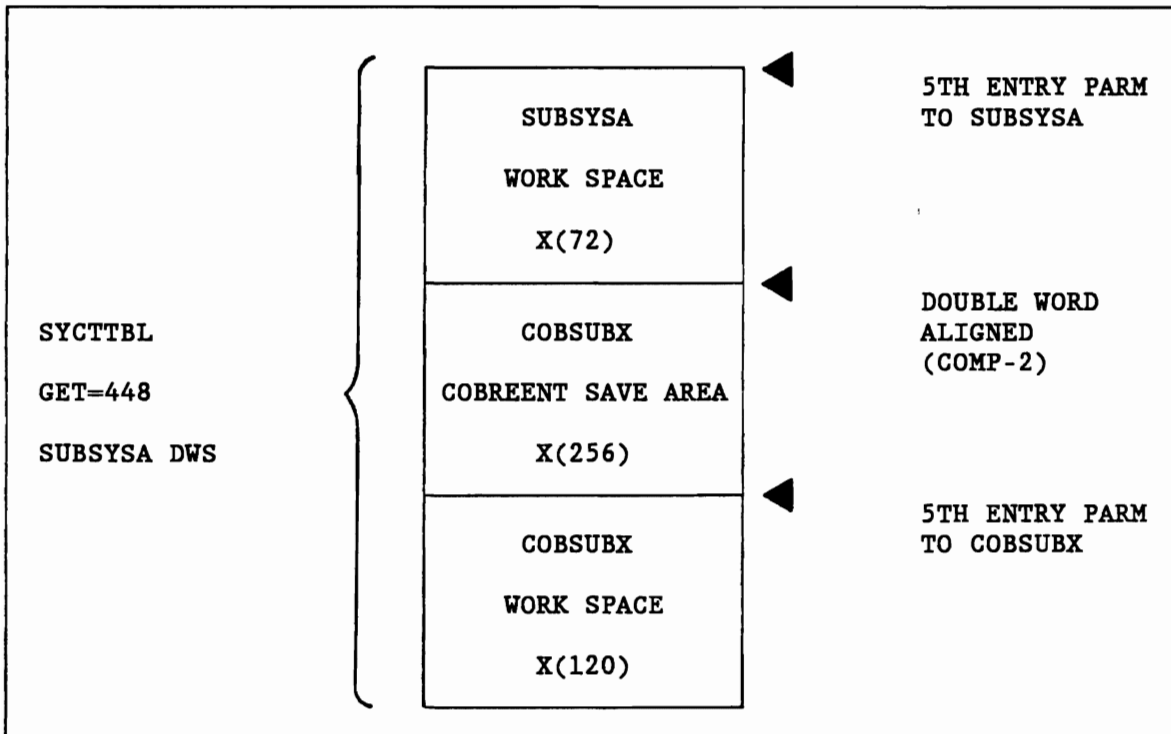


Figure D-2. COBREENT Save Areas
244

WORKING-STORAGE SECTION.		SUBSYSYA
77 COBSUBX-POINTER	PICTURE S9(4) COMP VALUE +104.	SUBSYSYA
* THIS IS THE LOCATION IN REENTSBS OF THE COBSUBX VCON.		SUBSYSYA
LINKAGE-SECTION.		SUBSYSYA
01 INPUT-MESSAGE	PIC X(500).	SUBSYSYA
01 SPA	PIC X(500).	SUBSYSYA
01 SCT	PIC X(100).	SUBSYSYA
01 R-CODE	PIC S9(7) COMP.	SUBSYSYA
01 DWS.		SUBSYSYA
02 SUBSYSYA-WORK-AREAS.		SUBSYSYA
05 MISCELLANEOUS-AREAS	PIC X(56).	SUBSYSYA
05 CODE-FIELD	PIC 99.	SUBSYSYA
05 PARM-ALIGNMENT	PIC S9(7) COMP.	SUBSYSYA
05 LOGIC-PARM-FOR-COBSUBX	PIC 99.	SUBSYSYA
02 FORCE-DOUBLE-WORD	COMP-2 SYNC.	SUBSYSYA
02 COBREENT-SAVE-AREA-FOR-COBSUBX	PIC X(256).	SUBSYSYA
02 COBSUBX-DWS	PIC X(120).	SUBSYSYA
PROCEDURE DIVISION		SUBSYSYA
USING INPUT-MESSAGE, SPA, SCT, R-CODE, DWS.		SUBSYSYA
.		SUBSYSYA
.		SUBSYSYA
.		SUBSYSYA
CALL 'COBREENT'		SUBSYSYA
USING COBSUBX-POINTER, INPUT-MESSAGE, SPA, SCT,		SUBSYSYA
R-CODE, COBSUBX-DWS, LOGIC-PARM-FOR-COBSUBX.		SUBSYSYA
.		SUBSYSYA
.		SUBSYSYA
.		SUBSYSYA
MOVE CODE-FIELD TO R-CODE.		SUBSYSYA
GOBACK.		SUBSYSYA

Figure D-3. Reentrant Subsystem Sample Coding.

WORKING-STORAGE SECTION.	COBSUBX
77 CONSTANT-ITEMS PIC X9(8) VALUE 'CONSTANT'.	COBSUBX
77 LOGPUT-POINTER-88 PIC S9(4) COMP VALUE +88.	COBSUBX
LINKAGE-SECTION	COBSUBX
01 P1 PIC X.	COBSUBX
01 P2 PIC X.	COBSUBX
01 P3 PIC X.	COBSUBX
01 P4 PIC X.	COBSUBX
01 DYNAMIC-WORK-SPACE	COBSUBX
02 MISCELLANEOUS-WORK-AREAS PIC X(120).	COBSUBX
.	COBSUBX
.	COBSUBX
01 MY-FIRST-SUBROUTINE-PARAMETER PIC 99.	COBSUBX
PROCEDURE DIVISION	COBSUBX
USING P1, P2, P3, P4, DYNAMIC-WORK-SPACE,	COBSUBX
MY-FIRST-SUBROUTINE-PARAMETER.	COBSUBX
.	COBSUBX
.	COBSUBX
GO TO ERROR1, ERROR2, ERROR3, ETC.	COBSUBX
DEPENDING ON MY-FIRST-SUBROUTINE-PARAMETER.	COBSUBX
.	COBSUBX
.	COBSUBX
ERROR1.	COBSUBX
ERROR2.	COBSUBX
ERROR3.	COBSUBX
ETC.	COBSUBX
.	COBSUBX
.	COBSUBX
CALL 'COBREENT' USING LOGPUT-POINTER-88, P1.	COBSUBX
GOBACK.	COBSUBX

Figure D-4. Reentrant Subroutine Sample Coding.

Appendix E

NONREENTRANT SUBSYSTEMS

E.1 INTRODUCTION

Nonreentrant OS/VIS or ANS COBOL subsystems do not use a Dynamic Working Storage area in the Linkage Section. Instead, as illustrated in Figure E-1, both constant and variable data areas are defined in the Working-Storage Section. Therefore, nonreentrant subsystems are always single-threaded; each input message is processed to completion and the subsystem returns to the Monitor before a new message can be processed by the subsystem. On the SYCTTBL macro for the subsystem, MNCL=1 and LANG=COB must be coded. For FORTRAN, code LANG=FORT. The GET (and FREE) parameters are not coded. Obviously, if messages might be input to the subsystem from more than one terminal concurrently, response time for each message queued after the first will be increasingly slower. Thus, it is recommended that COBOL subsystems always be coded as reentrant. Loading above the 16M line and VS COBOL II are not supported for non-reentrant programs.

To call a user subroutine (must be coded as nonreentrant if COBOL) or an Intercomm service routine, COBREENT is not used. Instead, the subsystem calls the routine directly as illustrated by the COBPUT call in the sample subsystem in Figure E-2. For calls to File Handler routines, use only the routine name such as GET or WRITE; omit the FH-prefix. For the Dynamic File Allocation, Page and DDQ special features, see the respective manuals for routine names. User subroutines must be resident in the Intercomm load module or in the same overlay segment as the subsystem; they are not defined in REENTSBS. The Intercomm environment for a nonreentrant subsystem is illustrated in Figure E-3.

E.2 CONSIDERATIONS FOR USING CONVERSE

Control of a conversational program environment is accomplished by Intercomm in different ways depending on the subsystem's residency:

- Resident

A copy of the subsystem load module is rolled out to a disk data set during "CONVERSE time", that is, the time between the call to CONVERSE and scheduling the next message from the same terminal. Subsequently, the load module is rolled in to process that next message.

- Overlay Loaded

Same as above, except the loaded overlay region may contain other subsystems to process other messages during (and after) "CONVERSE time."

- Dynamically Loaded

Same as above, except the subsystem remains in core until all "conversations in progress" have terminated.

The CONVERSE "rollout" file is a BDAM data set preformatted via the Intercomm utility CREATEGF (see Operating Reference Manual). The record format is fixed unblocked, with a block size of 1024. For on-line execution, a DD statement must be added to the Intercomm execution JCL, as follows:

```
//CONVSFIL DD DSN=INT.CONVSFIL,DISP=OLD,DCB=(DSORG=DA,OPTCD=R)
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EXAMPLE1.
REMARKS. THIS IS A NON-REENTRANT INTERCOMM COBOL SUBSYSTEM PROGRAM.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 CONSTANT-ITEMS                PICTURE X(8) VALUE 'CONSTANT'.
.
.   THESE ARE NEVER-CHANGING LITERAL VALUES...
.
77 INDEPENDENT-ITEMS            PICTURE S9(6)V99 VALUE ZERO COMP-3.
.
.   THESE ARE ACCUMULATORS, SAVE AREAS, FLAGS, ETC...
.
77 RETURN-VALUE                 PIC 99 VALUE 0.
.
01 OUTPUT-MESSAGE-AREA         PICTURE X(...).
01 FILE-RECORD-AREA            PICTURE X(...).
.
.   ADDITIONAL AREAS...
.
LINKAGE SECTION.
01 INPUT-MESSAGE-AREA COPY ICOMINMG.
   02 INPUT-TEXT                PICTURE X(...).
01 SPA                          PICTURE X.
01 SCT                          PICTURE X.
01 INTERCOMM-RETURN-CODE       PICTURE S9(8) COMP.

PROCEDURE DIVISION USING INPUT-MESSAGE-AREA,SPA,SCT,
INTERCOMM-RETURN-CODE.
.
.   PROGRAM PROCESSING LOGIC...
.
.   ...GO TO INTERCOMM.
.
INTERCOMM.
MOVE RETURN-VALUE TO INTERCOMM-RETURN-CODE.
GOBACK.
```

Figure E-1. Nonreentrant COBOL Subsystem Structure

```

PP 5740-CB1 RELEASE 2.4                                IBM CS/VS COBOL

1

00001  COCC10 ID DIVISION.                                00000010
00002  C00020 PROGRAM-ID. ECHMSG.                        00000020
00003  000030 REMARKS. THIS NON-REENTRANT SUBSYSTEM ECHOS AN INPLT MESSAGE 00000030
00004  C00040          CONTAINING UP TO 500 CHARACTERS OF TEXT BACK TO THE 00000040
00005  C00050          ORIGINATING TERMINAL, USING THE OUTPUT UTILITY.    00000050
00006  C00060          IT COPIES THE INPUT TO THE OUTPUT MESSAGE AREA.    00000060
00007  000070          MODIFIES THE MESSAGE HEADER, APPENDS THE AUTHOR'S NAME, 00000070
00008  C00080          AND MESSAGE ENDING CHARACTER, BEFORE CALLING COBPWT TO 00000080
00009  000090          QUEUE THE MESSAGE FOR THE INPUT TERMINAL.            00000090
00010  CCC100 ENVIRONMENT DIVISION.                      00000100
00011  C00110 DATA DIVISION.                            00000110
00012  000120 WORKING-STORAGE SECTION.                   00000120
00013  000130 01 HEX-CODES COPY ICCMHEXC.                00000130
00014  C          01 HEX-CODES.                            00000100
00015  C          05 HEX-00 PIC X VALUE ' '.              00000200
00016  C          05 CCDE-00 REDEFINES HEX-CO PIC X.     00000300
00017  C          05 HEX-15 PIC X VALUE ' '.              00000400
00018  C          05 CCDE-21 REDEFINES HEX-15 PIC X.     00000500
00019  C          05 HEX-37 PIC X VALUE ' '.              00000600
00020  C          05 CCDE-55 REDEFINES HEX-37 PIC X.     00000700
00021  C          05 HEX-50 PIC X VALUE 'E'.              00000800
00022  C          05 CCDE-80 REDEFINES HEX-50 PIC X.     00000900
00023  C          05 HEX-51 PIC X VALUE ' '.              00001000
00024  C          05 CCDE-81 REDEFINES HEX-51 PIC X.     00001100
00025  C          05 HEX-52 PIC X VALUE ' '.              00001200
00026  C          05 CCDE-82 REDEFINES HEX-52 PIC X.     00001300
00027  C          05 HEX-53 PIC X VALUE ' '.              00001400
00028  C          05 CCDE-83 REDEFINES HEX-53 PIC X.     00001500
00029  C          05 HEX-54 PIC X VALUE ' '.              00001600
00030  C          05 CCDE-84 REDEFINES HEX-54 PIC X.     00001700
00031  C          05 HEX-55 PIC X VALUE ' '.              00001800
00032  C          05 CCDE-85 REDEFINES HEX-55 PIC X.     00001900
00033  C          05 HEX-56 PIC X VALUE ' '.              00002000
00034  C          05 CCDE-86 REDEFINES HEX-56 PIC X.     00002100
00035  C          05 HEX-57 PIC X VALUE ' '.              00002200
00036  C          05 CCDE-87 REDEFINES HEX-57 PIC X.     00002300
00037  C          05 HEX-72 PIC X VALUE ' '.              00002350
00038  C          05 CCDE-114 REDEFINES HEX-72 PIC X.    00002352
00039  C          05 HEX-FF PIC X VALUE ' '.              00002400
00040  C          05 CCDE-255 REDEFINES HEX-FF PIC X.     00002500

00042  CCC150 01 AUTHORS-NAME.                            00000150
00043  000160 02 OUT-NAME PIC X(10) VALUE ' T.ELGUERA'. 00000160
00044  00017C 02 OLT-MSG REDEFINES OLT-NAME.             00000170
00045  C00180 04 NAME-CHAR PIC X OCCURS 10 TIMES.       00000180
00046  C00250 01 WCRK-SPACE COPY ICCMCHWS.               00000181
00047  C          01 WCRK-SPACE.                          00000100
00048  C          02 OUTPUT-MESSAGE.                      00000200
00049  C          04 CPSSG-MCR.                            00000300
00050  C          06 OPSGH-LENGTH PIC $9999 COMP.        00000400
00051  C          06 OPSGH-CPR PIC X.                    00000500
00052  C          06 OPSGH-RSC+ PIC X.                   00000600
00053  C          06 OPSGH-RSC PIC X.                    00000700

```

Figure E-2. Echo Message Example; Nonreentrant COBOL (Page 1 of 3)

2

00054 C		06	OMSGH-SSC	PIC X.	00000800
8C055 C		06	OPSGH-MMN	PIC XXX.	00000900
0C056 C		06	OPSGH-DATE.		00001000
0C057 C		08	OMSGH-YR	PIC 99.	00001100
0C058 C		08	OMSGH-PERIOD	PIC X.	00001200
00059 C		08	OMSGH-JULIAN-DAY	PIC 999.	00001300
0CC60 C		06	OPSGH-TIME.		00001400
0CC61 C		08	OMSGH-MM	PIC 99.	00001500
00062 C		08	OMSGH-MM	PIC 99.	00001600
0C063 C		08	OMSGH-SS	PIC 99.	00001700
0C064 C		08	OMSGH-TM	PIC 99.	00001800
00065 C		06	OPSGH-TID.		00001900
0CC66 C		08	OMSGH-T11	PIC X.	00002000
00067 C		08	OMSGH-T12-3	PIC XX.	00002100
0C068 C		08	OMSGH-T14-5	PIC 99.	00002200
0CC69 C		06	OPSGH-COM	PIC S9999 COMP.	00002300
00070 C		06	OPSGH-FLGS	PIC X(2).	00002400
0CC71 C		06	OMSGH-BPM	PIC X(3).	00002450
0C072 C		06	OPSGH-SSCM	PIC X.	00002500
0C073 C		06	OPSGH-USR	PIC X.	00002600
00074 C		06	OPSGH-ADDR	PIC XX.	00002700
0C075 C		06	OPSGH-LOG	PIC X.	00002750
00076 C		06	OMSGH-BLK	PIC X.	00002800
0CC77 C		06	OPSGH-VPI	PIC X.	00002900
00079	C0C260	04	OUTPUT-MESSAGE-TEXT	PIC X	OCCURS 510 TIMES.00000182
0CC80	CC0270	02	CCBPUT-RETURN-CODE	PIC 99.	00000183
00081	C00280	88	QUEUED	VALUE ZERO.	00000184
0C082	000290	02	I	PIC S9(4) COMP.	00000185
00083	000300	02	J	PIC S9(3) COMP.	00000186
00084	C00190		LINKAGE SECTION.		00000190
0C085	C00200	01	INPUT-MESSAGE COPY ICOMIANG.		00000200
0CC86 C		01	INPUT-MESSAGE.		00000160
0CC87 C		04	MESSG-MDR.		00000200
0CC88 C		06	MSGH-LENGTH	PIC S9999 COMP.	00000300
00089 C		06	MSGH-CPR	PIC X.	00000400
0CC90 C		06	MSGH-RSCH	PIC X.	00000500
00091 C		06	MSGH-RSC	PIC X.	00000600
00092 C		06	MSGH-SSC	PIC X.	00000700
0CC93 C		06	MSGH-MMN	PIC XXX.	00000800
0CC94 C		06	MSGH-DATE.		00000900
00095 C		08	MSGH-YR	PIC 99.	00001000
00096 C		08	MSGH-PERIOD	PIC X.	00001100
0C097 C		08	MSGH-JULIAN-DAY	PIC 999.	00001200
0C098 C		06	MSGH-TIME.		00001300
0CC99 C		08	MSGH-MM	PIC 99.	00001400
00100 C		08	MSGH-MM	PIC 99.	00001500
00101 C		08	MSGH-SS	PIC 99.	00001600
0C102 C		08	MSGH-TM	PIC 99.	00001700
0C103 C		06	MSGH-TID.		00001800
00104 C		08	MSGH-T11	PIC X.	00001900
0C105 C		08	MSGH-T12-3	PIC XX.	00002000
00106 C		08	MSGH-T14-5	PIC 99.	00002100
0C107 C		06	MSGH-COM	PIC S9999 COMP.	00002200
0C108 C		06	MSGH-FLGS	PIC X(2).	00002300
0C109 C		06	MSGH-BPM	PIC X(3).	00002350
00110 C		06	MSGH-SSCM	PIC X.	00002400

Figure E-2. Echo Message Example; Nonreentrant COBOL (Page 2 of 3)

```

3

00111 C          06 MSGH-USR          PIC X.          00002500
00112 C          06 MSGH-ADDR        PIC XX.         00002600
00113 C          06 MSGH-LOG         PIC X.          00002700
00114 C          06 MSGH-BLK         PIC X.          00002750
00115 C          06 MSGH-VMI         PIC X.          00002800

00117 C00210      04 INPLT-MESSAGE-TEXT PIC X          OCCURS 500 TIMES. 00000210
00118 C          01 SYSTEM-PARAMETER-TABLE PIC X.         00000220
00119 C          01 SUBSYSTEM-CONTROL-TABLE PIC X.         00000230
00120 C          01 INTERCOMM-RET-CCDE PIC S9(7)          COMPUTATIONAL.00000240
00121 C          000310 PROCEDURE DIVISION USING          00000310
00122 C          000320          INPLT-MESSAGE          00000320
00123 C          000330          SYSTEM-PARAMETER-TABLE 00000330
00124 C          000340          SUBSYSTEM-CONTROL-TABLE 00000340
00125 C          000350          INTERCOMM-RET-CODE.     00000350
00126 C          000370          MOVE PESSG-HDR TC CMESG-HDR. 00000370
00127 C          000380          MOVE DMSGH-RSCH TC DMSGH-SSCH. 00000380
00128 C          00039C          MOVE DMSGH-RSC TC DMSGH-SSC. 00000390
00129 C          000400          MOVE LCH-VALUES TC DMSGH-RSCH. 00000400
00130 C          000410          MOVE LCH-VALUES TC DMSGH-RSC. 00000410
00131 C          000420          MOVE HEX-57 TC DMSGH-VMI.    00000420
00132 C          000430          PERFORM MOVE-A-CHARACTER VARYING I FROM +1 BY +1 00000430
00133 C          000440          UNTIL I IS EQUAL TO MSGH-LENGTH - 42. 00000440
00134 C          000450          PERFORM NAME-PCVE VARYING J FROM +1 BY +1 UNTIL J > +10. 00000450
00135 C          000460          MOVE HEX-37 TO OUTPUT-MESSAGE-TEXT (I). 00000460
00136 C          000470          COMPUTE DMSGH-LENGTH = I + 42. 00000470
00137 C          000480          CALL 'COBPUT' USING  DLPUT-MESSAGE 00000480
00138 C          000510          CCBPUT-RETURN-CODE. 00000510
00139 C          000520          IF NOT QUEUED 00000520
00140 C          000530          PCVE COBPUT-RETURN-CODE TO INTERCOMM-RET-CODE 00000530
00141 C          000540          ELSE 00000540
00142 C          000550          MOVE ZEROS TO INTERCOMM-RET-CODE. 00000550
00143 C          000560          GOBACK. 00000560
00144 C          000570          SUBROUTINE SECTION. 00000570
00145 C          000580          MOVE-A-CHARACTER. 00000580
00146 C          000590          MOVE INPUT-MESSAGE-TEXT (I) TO OUTPUT-MESSAGE-TEXT (I). 00000590
00147 C          000600          NAME-PCVE. 00000600
00148 C          000610          MOVE NAME-CHAR (J) TO DLPUT-MESSAGE-TEXT (I). 00000610
00149 C          000620          COMPUTE I = I + 1. 00000620

```

Figure E-2. Echo Message Example; Nonreentrant COBOL (Page 3 of 3)

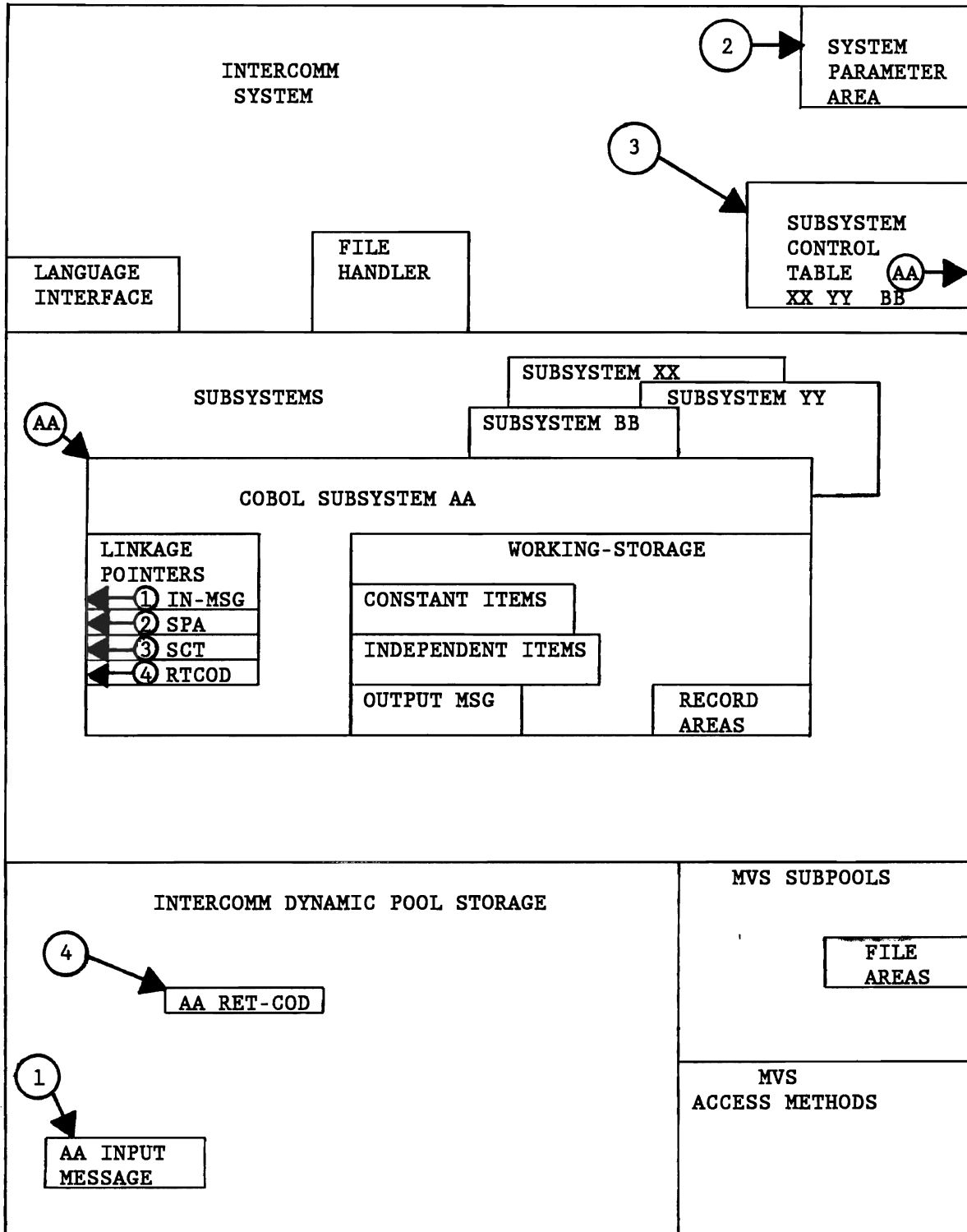


Figure E-3. Nonreentrant Application Program Environment

INDEX

	<u>Page</u>		<u>Page</u>
Abend 519	45	Broadcast Group	20,75,102,103,107
Abend 10nn	46.3	Broadcast Table (PMIBROAD)	239,241
ACCESS function (DFA)	110	BROADCAST. <u>See</u> Broadcast Table.	
ADABAS data base	15,52	BSAM access method	51-52,57,60-61
ALLOCATE function (DFA)	110	BTAM	11
ALTER (COBOL verb)	45	--simulator	111,129-131
Alternate Format Table (PMIALTRP)	240,241	--switched devices	21
Alternate Index (VSAM)	71	BTAMSCTS. <u>See</u> Front End Queue Table.	
Alternate Path processing (VSAM)	71	BTAMSIM module	131
Alternate terminal processing	75	BTERM macro	239
AMODE linkedit parameter	45,46.1,225,227	BTSAMP table	239
AMP parameter (VSAM JCL)	67,74	BTVERB macro	
Application programs. <u>See</u> Subsystems.		--and BTVRBTB	17,18
Assembler Language	16,34,105,110.1	--and editing	35
AUTOLOK feature	94	--and priority	21
Back End--defined	11	--and sample subsystem	130,132,168
Back End Device Table. <u>See</u> Device Table.		--and subsystem codes	18
Back End Station Table. <u>See</u> Station Table.		--table summary	239
Backout-on-the-Fly	6,34	BTVRBTB. <u>See</u> Front End Verb Table.	
Batch execution	1,3,30	Cancelled messages	34,46.4
Batch Report Table (REPTAPE)	240,241	Change/Display Utility	
BCGROUP macro	239	--described	16
BDAM access method		--and formatting required,	
--COBOL verbs	51	fixed text messages	76-77,167
--DCB parameters	64	--and message switching	41
--DD statement parameters	52	--and Output Utility	76-77
--and exclusive control	57	--and sample subsystem	167
--option codes	65	--tables used by	241
--and File Handler Parameters	54	Change table (CHNGTB)	167,180,239,241
--processing	64	Checkpointing	6,28
--return codes	66	CHNGTB. <u>See</u> Change Table.	
--sample inquiry subsystem	111	CICS (IBM)	11
--service routines	51,64	Closing files	51,52,57
BDW. <u>See</u> Block Descriptor Word.		COBLOGCH copy member	46.2,111
Binary table search	30,110.1	COBPUT module	
BISAM access method		--calls to	39,97,99,100,101,247
--COBOL verbs	51	--error recovery	98
--DCB parameters	62	--function	39
--DD statement parameters	52	--and message flow using EDIT	
--and exclusive control	57	and OUTPUT	24
--and ISAM/VSAM compatibility	74	--and message queuing	97,99-101
--processing	62	--and message switching	35,41,99-101
--return codes	63	--and Output Utility	76
--service routines	51,62	--return codes	98
BLDL parameter (SUBMODS macro)	46	--and sample inquiry	
BLDL parameter (SYCTTBL macro)	46	subsystem	111,169-178
BLHOT module	29	--and subsystem logic using	
BLINE macro	239	EDIT and OUTPUT	37
Block Descriptor Word	61	COBREENT module	
		--and Amode switching	38,46.3
		--and COBSTORF	100
		--and CONVERSE	90

	<u>Page</u>		<u>Page</u>
--described	95	--calling format	90
--and DWSSNAP	46.4	--and conversation implementation options	81
--and exclusive control	59	--described	88
--and FECMDDQ	107	--and nonreentrant subsystems	247-248
--and FECMFDBK	108	--return codes	92
--and FECMRLSE	109	--roll-out file	247-248
--and FESEND	103	--subsystem logic using	89-91
--and File Handler	54	--and VS COBOL II	46.3,88
--function	38-39,95	--and XA/ESA	88
--and GET (QISAM)	62	CONVSFIL data set	248
--and GET (QSAM)	60	COPRE utility (MMU)	46.2
--and GETV (VSAM)	67-68	CREATEGF utility	64,248
--and INTSORTC	110	CREATSIM utility	129-130
--and LOGPUT	104	--JCL for	134-135
--and MSGCOL	99	CRT Page Facility. <u>See</u> Page Facility.	
--and passed parameters	39,46,46.3,95	Data Base interface	7,15,19,52,105,110
--and PUT (QISAM)	62	Data Set Control Table	17
--and PUT (QSAM)	60	Data Set Name Sharing (VSAM)	67,71
--and PUTV (VSAM)	67-68	Data strings	15,84,86
--and READ (BDAM)	64	Dataspeed 40 terminal	21,49
--and READ (BISAM)	62	DBINT function	110
--and READ (BSAM)	60	DBMS support	15,24,34,52,110
--and reentrant coding	45,105,243	DDQ. <u>See</u> Dynamic Data Queuing.	
--and RELEASE	56-57	Debugging program problems	129
--routine pointers (REENTSBS)	96-97	<u>See</u> also DWSSNAP Facility.	
--and sample subsystem	111	DES000 data set	167,239
--and save areas	38,46.3,95,105,243,244	Device control characters	49
--and SELECT	56	DEVICE macro	240
--and subroutine calls	105	Device Table (PMIDEVTB)	17,102,240,241
--and VS COBOL II	38,46.2,46.3,95	DFA. <u>See</u> Dynamic File Allocation.	
--and WRITE (BDAM)	64	Dispatcher	12,105,110.1
--and WRITE (BISAM)	62	DISPLAY (COBOL verb)	
--and WRITE (BSAM)	60	--and VS COBOL II	217
--and XA/ESA	46	DIVIDE (COBOL verb)	45
COBSTORF module	100-101	DL/I data base	15,52
COBUPC procedure	225	DMAP (of COBOL program)	45
COBUPCL procedure	225	DSCT. <u>See</u> Data Set Control Table.	
COB2PC procedure	226	DSN (FAR) option.	
COB2PCL procedure	227	<u>See</u> Data Set Name Sharing.	
Commands (Intercomm)	16,46.3,93-94,102	Dumps	129
Company/Report/Terminal Table (PMIRPTAB)	240,241	DWS. <u>See</u> Dynamic-Work-Space.	
COMPUTE (COBOL verb)	45	DWSCHK parameter (SPALIST macro)	46
Constants, defining	41	DWSCHK parameter (SYCTTBL macro)	46
--for VS COBOL II	46.2	DWSSNAP Facility	
--and XA/ESA	46	--coding format	46.5
Conversational subsystem. <u>See</u> Subsystems, conversational.		--described	46.4
Conversational time-out (Front End) and FESEND	103	--examples	46.5
CONVERSE		--parameters	46.4
--automatic time-out	90,92	--and user snaps	110.1
		--and VS COBOL II	46.3

	<u>Page</u>		<u>Page</u>
Dynamic Data Queuing		--and subroutine interface	105-106
--and conversational subsystem		--and subsystem structure	31-32
	81,86-87	--and VS COBOL II	46-46.3
--described	15	DYNLLOAD module	46
--and Front End Data Queuing	107	ECHOMSG sample subsystem	42-44.2
--and Message Mapping Utilities	47	ECT. <u>See</u> Edit Control Table.	
--and segmented input messages	22,30	EDIT. <u>See</u> Edit Utility.	
--subroutine entry names	110	Edit Control Table (PMIVERBS)	
--types of queues	86	--associated components	241
--use of	30,86-87	--described	49
Dynamic File Allocation	15,110	--macros	240
Dynamic file backout	6,34	--and message flow using EDIT	
Dynamic linkedit	30,46.3	and OUTPUT	24
Dynamic loaded programs	18,38,131,168	--and required fields	50
--above the 16M line		--and subsystem testing	167,180
	38-39,45,46.1,46.3,95,105	--and USRVERBS	167,180
--and CONVERSE	88,92	EDIT parameter (BTVERB macro)	35,37
--and dynamic linkedit	30	Edit subroutines	49
--linkedit of	225,227	Edit Utility	
--and VS COBOL II	46.1,46.3	--concepts	14,49
--and XA/ESA	45-46	--and CONVERSE	91
Dynamic Working Storage. <u>See</u>		--function	14
Dynamic-Work-Space.		--message flow using	24-25
Dynamic-Work-Space.		--and message switching	41,97
--and Assembler Language		--and PREPROG	38
Subroutines	105	--processing results	49-50
--and COBOL		--and sample subsystem	167
subroutines	46.1,46.3,105-106,243	--subsystem logic using	37-38
--and COBPUT	97	--and subsystem testing	111,167,180
--and COBSTORF	100	--tables used by	17,241
--and CONVERSE	89	--using	49-50
--and DWS checking	46,46.3,100	--and verb message	
--and DWSSNAP facility	46.3,46.4-46.5	identifier (MSGHVMI)	35,38,41
--example	44	Entry sequenced data sets (VSAM)	
--and File Handler	53		54,56,67,69,70
--and FREE parameter (SYCTTBL		--loading of	67,70
macro)	19,100,101	Error messages	
--and GET parameter (SUBMODS		--and calls to service routines	55
macro)	106	--and conversational subsystems	79
--and GET parameter (SYCTTBL		--and debugging	129
macro)	18,44.3,100,101	--and Dynamic Working Storage	
--and intersubsystem queuing	97,100-101	protection option	46
--and Message Collection	100-101	--and Edit Utility	50
--and Message Mapping Utilities	47	ESA (IBM). <u>See</u> XA/ESA.	
--and message switching	99-101	ESDS. <u>See</u> Entry sequenced data sets.	
--and nonreentrant subsystems	247	ESS. <u>See</u> Extended Security System.	
--and output message format	41	Exclusive Control	
--and parameters to COBREENT	39,46,95	--and File Handler Control Word	53
--prefix to	32	--for non-VSAM files	57-59
--protection option	18,46,46.3,130	--processing (file/record)	57-58
--and reentrant coding		--releasing (file/record)	59
conventions	16,39,45	--and resource control	110.1
--snaps of	46.3,46.4-46.5,110.1		

	<u>Page</u>		<u>Page</u>
--and Store/Fetch	85	FESENDC entry point	40,94,102-103,106
--for VSAM files	70	FETCH function	84-85
EXEC JCL statement parameter	131,168	FHCW. <u>See</u> File Handler Control Word.	
EXTDSCT. <u>See</u> External DSCT.		File access. <u>See</u> File Handler.	
Extended Security System	15,29,110.1	FILE command	56,57
EXTERM macro	110.1	File Attribute	
External DSCT	53-54,56-57,59,68	Record	17,51,56,61,62,67,71,241
FAR. <u>See</u> File Attribute Record.		File Description Records	76,167,239,241
FD (File Description)	45,52	--illustrated	182
FDETL macro	239	File Descriptions (FDs) and	
FDHDR macro	239	COBOL	45,52
FDR. <u>See</u> File Description Records.		File Handler	
FECM.	102	--access methods supported	52
<u>See</u> Front End Control Messages.		--and batch execution	30
FECMDDQ. <u>See</u> Front End Data		--and closing a file	52,57
Queuing.		--control areas	53
FECMFDBK. <u>See</u> Front End Feedback		--and CONVERSE	91
Messages.		--described	12,51-52
FECMRLSE. <u>See</u> Front End Queue		--and direct access files	64-66
Release.		--error checking	55
Feedback codes (VSAM)	70,73	--exclusive control	12
FENETWRK table	136,239	--non-VSAM files	57-59
FESEND		--VSAM files	59,70
--described	102-103	--general concepts	51-52
--and conversational time-out	103	--and indexed sequential files	62-63
--and CONVERSE time-out message	90	--and ISAM/VSAM compatibility	74
--and FESENDC entry point	94,102-103	--and message flow using EDIT	
--and Front End Control		and OUTPUT	24-25
Messages	106	--and message flow using MMU	22-23
--and Front End Feedback		--and opening a file	12,51,52,57
Messages	108	--parameters	53-54
--and Front End Queue		--return codes	53,55-56,59-60,63,66,73
Release	103,108	--and sample inquiry subsystem	111
--function	14,35	--SELECT and RELEASE functions	56
--and INTERLOG log codes	27-29	--and sequential access	
--and LOCK/UNLK commands	94	files	60-61
--and message flow using EDIT		--service routines	51,54-55,247
and OUTPUT	24-25	--subsystem processing	52-53
--and message flow using MMU	22-23	--tables used by	239,241
--and message header altering	21,40,41,42	--and TCTV time-out	19
--and Message Mapping		--and undefined record format	61
Utilities	22-23,47	--and variable length records	61
--and MSGHRSC and MSGHRSCH	35,40	--and VSAM	51,54,56-59,67-73
--option codes	103	--and XA/ESA	46,54,56,57
--and Output Utility	24-25,38,75	File Handler Control Word	
--return codes	103	--and automatic error checking	55
--and subsystem coding	35,40	--and closing a file	57
--and subsystem logic using		--described	53-54
EDIT and OUTPUT	38	--and direct access files	64-65
--and subsystem logic using MMU	36	--and exclusive control	57
--and terminal messages	21,40,41	--and indexed sequential files	62
--and VTAM	103,108	--and ISAM/VSAM compatibility	74
		--return codes	55

	<u>Page</u>		<u>Page</u>
--and undefined records	61	--and ISAM/VSAM compatibilty	70,74
--and VSAM files	67-73	--and sequential access files	60-61
File Handler Data Set Control Table	17,239	--and subsystem processing	52
File I/O verbs (COBOL)	46.2,51	GETDATE macro	110.1
File Recovery feature	6,51,104	GETV service routine	
File Table (PMIFILET)	240,241	--calling format	67-68,72
Fine Tuner Commands	240	--and exclusive control	70
Flushed messages	21,29	--and FHCW reason codes	70
Format Description Record.		--function	51
<u>See</u> File Description Records.		--and subsystem processing	52
FORTRAN	31,247	--and VSAM files	67-68
FREE parameter		--and VSAM processing options	69-71
(SYCTTBL macro)	19,46,100-101,247	GOBACK (COBOL verb)	36,37,40,45,46.1,52
Front End		HEXCON macro	110.1
--and conversational processing	81,103	HPRTY parameter (BTVERB macro)	21
--defined	11	IAM access method	52,54
--and TP queuing interface	12	IBM 2740 Terminal	21
Front End Control Messages	106-109	IBM 3270 Terminals	11,21,49,108,131
--and FESEND	102	--and simulation testing	130
--return codes	106	ICOMDWS copy	
Front End Data Queuing	30,107,108	member	40,43,111,115,234-236
Front End Feedback Messages	107-108	ICOMHEXC copy member	40,42,169,230
Front End Network Configuration Table	17,239,241	ICOMINMG copy	
Front End Queue Release	103,108-109	member	40,42,111,115,231-233
Front End Queue Table (BTAMSCCTS)	239,241	ICOMLINK macro	136,185,215
Front End Verb Table (BTVRBTB)		ICOMSBS copy member	
--and AUTOLOK	94	--described	39
--creation of	17,239,241	--illustrated	42,114
--and CONVERSE	90-91	--listing of	237-238
--and editing	37,49	--and sample inquiry subsystem	111
--function	17-18	--and service routines	54,110,110.1
--and message processing (VMI)	35	--and user subroutines	105
--and USBTVRB	18,130,132,168	--and XA/ESA	46
FTUN command	46.3	IDMS data base	15,52
GENFTBLE macro	240	IGZEBST (IBM) module	46.3
GET parameter (SUBMODS macro)		IGZOnnI error message (IBM)	46.3
46,46.1,106		IJKDELAY service routine	30,110.1
GET parameter (SYCTTBL macro)		IJKPRINT service routine	30
--described	18	Indicative dumps	129
--and Dynamic Working Storage protection option	46	INITLU6 function	11,96,110,237,238
--and Message Collection	100-101	INTDEQ macro	110.1
--and subroutines	46.1,243	INTENQ macro	110.1
--and subsystem structure	31,46.1,247	INTERCOMM environment	9-15
GET service routine		INTERCOMM tables	17,239-241
--calling format	60,62	--REENTSBS subroutines table	39,95-97,240
--and exclusive control	57-59	INTERLOG file 26-29,104,161-166,191-196	
--function	51	--JCL for	138
--and indexed sequential files	62	INTFETCH function	110
		INTPOST macro	110.1

	<u>Page</u>		<u>Page</u>
INTSCT table	18,130,167,240,241	LIMCT parameter (JCL)	52,64
--and USRSCTS	18,130,132,167,180	LINE macro	240
INTSORT Facility	109-110	Line control	6-7
--return codes	110	LINEGRP macro	239
INTSORTC entry point	109,237,238	Linkage Section	
INTSPA table	82-83,240,241	--and DWS snaps	46.4-46.5
INTSTORE function	110	--example	
INTTCB macro	91	32,42,115,127,171,205,245,246,248	
INTTIME macro	110.1	--and File Handler control	
INTUNSTO function	110	areas	53
INTWAIT macro	110.1	--and ICOMINMG and ICOMDWS	40
ISAM (see also BISAM and QISAM)		--order of Intercomm parameters	31
--access by File Handler	52	--and output message format	41
--File Handler processing	62	--and reentrant coding	
--File Handler return codes	63	conventions	45
--File Handler service routine		--and System Parameter List	83,99
parameters	54	Linkedit	
--and VSAM	67,70,73,74	130,136-137,168,185-186,225,227	
ITCBPMSS field in ITCB	91	--dynamic	30,46.3
ITEM macro	240	--and loading above the 16M line	
IXFCHKPT module	28	45,46.1,225,227	
IXFDSCTA macro	239	LKEDT procedure	137,186,216
IXFDSC1/2/3/n table	239,241	LNAME parameter (SUBMODS macro)	46,105
IXFLOG (File Recovery logging)	28	LOAD command	30,46.3
JCL		Loading above the 16M line	
--COBOL procedures	225-227	--requirements for. <u>See</u> XA/ESA.	
--CREATSIM utility	134-135	LOADNAM parameter (SYCTTBL macro)	18,46
--DD statement for File		LOCK command	93-94
Handler	12,51-52	Log Analysis utility	21,26
--for direct access files	64-66	Log codes	26-29,46.4,104
--for indexed sequential files	62	Logging. <u>See</u> INTERLOG file, LOGPUT.	
--for ISAM/VSAM compatibility	74	LOGCHARS table	241
--sample Simulation Mode	136-138	LOGPROC module	28
--sample Test Mode	185-187	LOGPRINT utility	21,26
--sample VS COBOL II test	215-217	--sample JCL for	138
--for VSAM files	67	--sample printout	161-166,191-196,218-223
JOBCAT DD statement (VSAM)	131	LOGPUT module	
Journaling. <u>See</u> Logging.		--calling format	104
KEYTABLE (Change/Display)	239,241	--function	14,26
Key sequenced data sets (VSAM)	54,67,71	--and INTERLOG entries	26,28-29
KEYCREAT utility	64	--and user entries to	
KSDS. <u>See</u> Key sequenced data sets.		INTERLOG	27,104
LANG parameter (SYCTTBL macro)	18,46,46.3,247	LOGTROUT table	104
LAYOUT macro	110.1	LSR (pools) option (VSAM)	67,71
LCOMP macro	239	LU6.2 support	11,15,28,110
LET parameter (linkedit)	46.3,225,227	LUNIT macro	236
LIBECOB procedure	225	Macros (Intercomm)	
LIBECOBL procedure	225	--for features	110.1
LIBELINK procedure	132,180	--for tables	239-240
		MAPCLR function	110
		MAPEND function	34,36,48,108,110

	<u>Page</u>		<u>Page</u>
MAPFREE function	110	--service routines	110
MAPIN function	36,48,106,110	--and subsystem logic	35-36,48
MAPOUT function	36,48,110	--and subsystem testing	129,130,131
MAPURGE function	110	--and symbolic maps	46.2,47
MERGE (COBOL verb)	46.2	--tables used by	17,241
Message Collection		--and VS COBOL II	46.2
--calling format	99	Message switching	35,41,99,100
--and COBPUT	39,97-101	MMU. <u>See</u> Message Mapping Utilities.	
--function	14	MMUC command	130,134,141
--and INTERLOG	28-29	MMUVTBL table	241
--log codes	27	MNCL parameter (SYCTTBL	
--and message switching	99-101	macro)	19,46.2,46.3,131,168,247
--and MSGHMMN	20	MODCNTRL macro	105
--and MSGHUSR	21	Monitor control functions	7
--and partial message freeing		MONOVLY module	46.3
99,100-101	99,100-101	MRINTER module	28
--return codes	34,99-100	MROTPUT module	34
Message ending character	40	MRQMNGR module	28-29
Message flow		MSGAC module	29
--using Edit and Output	24-25	MSGCOL. <u>See</u> Message Collection.	
--using MMU	22-23	MSGHBLK field	20
Message header		MSGHBMN field	20,26,183
--changing	35	MSGHCON field	20,34
--and CONVERSE	90	MSGHDAT field	20,26,104
--COPY members for	231-236	MSGHFLGS field	20
--described	19-21	MSGHLEN field	
--and Message Collection	97,99	--described	20
--and message flow using EDIT		--and Front End Control	
and OUTPUT	24-25	Messages	106
--and message flow using MMU	22-23	--and message switching	99-101
--and message processing logic	35	--and output messages	40
--and message routing	18	--and user log entries	104
--and message routing	18	MSGHLOG field	20,26,104
--and output messages	35,77	MSGHMMN field	20,26
--specifications for the Output		MSGHQPR field	20
Utility	77	--and COBPUT call	97
--and subsystem structure	31,39-40	--described	22
--and Test Mode input	183	--and segmented input	76,97,102
--and user log entries	104	MSGHRETN field	20,34
Message Mapping Utilities		MSGHRSC field	
--and Assembler Language		--and assigning verb to terminal	94
subroutines	105	--and COBPUT	97
--described	14,47	--described	20
--and Front End Control		--and Message Collection	99
Messages	106-109	--and message routing	18,35
--function	14	--and output messages	39,77
--and log codes	27	--and subsystem logic using	
--maps for sample subsystem	133	EDIT and OUTPUT	37
--message flow using	22-23	--and Test Mode	183
--and messages to Front End	102	MSGHRSC field	
--and MSGHQPR	22	--and assigning verb to	
--and MSGHVMI	40	terminal	94
--processing	22-23,30,47-48	--and COBPUT	97
--and sample inquiry			
subsystem	111,130		

	<u>Page</u>		<u>Page</u>
--described	20	NAME parameter (SUBMODS macro)	105,110.1
--and Message Collection	99	NCAL. <u>See</u> NOCALL parameter.	
--and message routing	18,35	Network Table	17,239,241
--and output messages	39,77	NOCALL parameter (linkedit)	46.3,225,227
--and subsystem logic using		NODYNAM compiler option	46.1,225
EDIT and OUTPUT	37	Nonreentrant subsystems	16,31,46.3,247-252
--and Test Mode	183		
MSGHSSC field	20,39,106	OFT. <u>See</u> Output Format Table.	
MSGHSSCH field	20,39,106	On-line environment (described)	1-3
MSGHTID field		On-line testing	129
--described	20	OPEN (FAR) option	51
--and FESEND/FESENDC	102-103	Opening files	12,51-52
--and Front End Data Queuing	107	OPTIMIZE (VS COBOL II) option	46.1
--and Front End Queue Release	108	OTQUEUE user exit (VTAM)	21
--and message switching	41	Output Format Number	97,168
--and Output Utility	75	Output Format Table	
--and Store/Fetch facility	84	--associated components	241
--and subsystem coding	40	--creation of	240
--and Test Mode	183	--described	76
MSGHTIM field	20,26,104	--and message flow using	
MSGHUSR field	20,21	EDIT and OUTPUT	24
MSGHVMI field		--and subsystem testing	168,181-182
--and Change/Display Utility	76-77,167	Output Utility	
--and COBPUT	97	--and COBPUT	97
--described	20,22	--and CONVERSE	90
--and Edit Utility	35,38,41,97	--described	16,75
--and FESEND	21,40,102	--and Front End feedback	
--and Front End Control		messages	108
Messages	106	--function	14
--and Front End data queuing	107	--and INTERLOG log codes	26-27
--and input messages	35	--message flow using	24-25
--and message processing	35	--message header	
--and message switching	35,41	specifications for	77
--and output messages	40	--and message switching	30,41
--and Output Utility	35,75-77	--and MSGHQPR	22
--and PREPROG	38	--and MSGHVMI	35,77
--and subsystem logic using		--processing	24,75-76
EDIT and OUTPUT	37	--and sample inquiry	
--and Test Mode	183	subsystem	111,168
--values	35	--subsystem logic using	35,37-38
Multiregion System		--and subsystem testing	168
--and batch interface	30	--tables used by	17,241
--and COBPUT calls	98	Overlay linking	46.3
--and testing	129		
Multithread processing		Pad character table (PADDTBLE)	239,241
--and COBREENT	38	PADD macro	239
--described	4-5	PADDTBLE. <u>See</u> Pad character table.	
--and exclusive control	57	Page Facility	15,16,47,110,240,241,247
--and message flow using MMU	22	PAGETBL macro	240
--and reentrant subsystems	16,38	PAGETBLE table	240,241
--and subsystem testing	131,168	PARM macro	180,240
MVS/XA (IBM). <u>See</u> XA/ESA.		PATRN macro	240

	<u>Page</u>		<u>Page</u>
Pattern Table (PTRNTBLE)	240,241	--and ISAM/VSAM compatibility	74
PERFORM (COBOL verb)	45,46.1	QOPEN function	87,110
PMIALTRN macro	240	QPR. <u>See</u> MSGHQPR field.	
PMIALTRP. <u>See</u> Alternate Format Table.		QREAD function	87,110
PMIBROAD. <u>See</u> Broadcast Table.		QREADX function	87,110
PMICANC. <u>See</u> USRCANC.		QWRITE function	87,110
PMIDEVTB. <u>See</u> Device Table.		QWRITEX function	87,110
PMIELIN macro	240	QSAM access method	51-52,57,60-61
PMIEXLD utility	167,239	Queues	
PMIFILET. <u>See</u> File Table.		--and COBPUT	97
PMIRCEND. <u>See</u> Output Format Table.		--and Dynamic Data Queuing	15,86-87
PMIRCNTB. <u>See</u> Output Format Table.		--and FESEND	41,102
PMIRPTAB. <u>See</u> Company/Report/ Terminal Table.		--and Front End	11
PMISNAP macro	110.1	--illustrated	23,25
PMISTATB. <u>See</u> Station Table.		--and INTERLOG	26
PMISTOP DD statement	131	--and Message Collection	22,24,99,101
PMISTOP macro	240	--and message switching	41
PMITEST module	167,168	--and TP interface	12
PMIVERBS. <u>See</u> Edit Control Table.		RBA. <u>See</u> Relative byte address.	
PMIWTO macro	110.1	RBN. <u>See</u> Relative block number.	
PMIWTOR macro	110.1	RDW. <u>See</u> Record Descriptor Word.	
PREPROG module	35,38,41,46,49,91	READ service routine	
--and XA/ESA	38,46	--calling format	60,62,64
PREPROGE user exit	46.2	--and direct access files	64-66
PREPROGI user exit	46.2	--and exclusive control	57-59
PTRNTBLE. <u>See</u> Pattern Table.		--function	51
PUT service routine		--and indexed sequential files	62-63
--calling format	60,62	--and ISAM/VSAM compatibility	70,74
--and exclusive control	59	--and sequential access files	60-61
--function	51	--and subsystem processing	52
--and indexed sequential files	62-63	READONLY (FAR) option	51,71
--and sequential access files	60-61	READY TRACE (COBOL)	46.2
--and subsystem processing	52	Reason codes, VSAM	70,73
--and ISAM/VSAM compatibility	70,74	Record Descriptor Word (RDW)	61,68
PUTV service routine		Reentrant subsystems	
--calling format	68,72	--and COBREENT	38-39
--and exclusive control	70	--coding conventions	45
--and FHCW reason codes	70	--and Dynamic Working Storage	18,45
--function	51	--environment	33
--and subsystem processing	52	--example	42-44.2
--and VSAM files	67-68	--and File Handler service routines	54
--and VSAM processing options	69-70	--and FREE parameter (SYCTTBL macro)	19
QBUILD function	87,110	--and LANG parameter (SYCTTBL macro)	18
QCLOSE function	87,110	--and loading above the 16M line	45-46
QISAM access method		--and Message Collection	100-101
--and exclusive control	57-59	--and MNCL parameter (SYCTTBL macro)	19,131,168
--and File Handler parameters	54	--structure	31-32
--and File Handler service routines	51-52,62		

	<u>Page</u>		<u>Page</u>
--and user subroutines		--VSAM	69-70,73
	105-106,243-246	--from Front End Control	
--vs. nonreentrant subsystems	16	Message facility	106
--and VS COBOL II	46.1-46.3	--from INTSORTC	110
REENTSBS table		--from LOGPUT	104
--creation of	240	--from Message Collection	99
--described	95-97	--from queuing routine	40
--function	39	--and RETURN-CODE data name	45
--listing of	96-97	--from subroutines	106,111
--and service routines	46.2,95,110,241	--from subsystems	22,31,34
--and user subroutines	39,46.3,105,243	REUS parameter (linkedit)	45,225
--and USRSUBS	97,130,132,243	REUSE parameter (SYCTTBL macro)	46,46.3
Relative block number (RBN)	54,64-66	RLSE command	108
--example	111	RMODE linkedit parameter	45,46.1,225,227
Relative byte address (RBA)	54,67-69	RPTnnnnn member. <u>See</u> Output Format	
Relative record data set (VSAM)	54,67	Table.	
Relative record number (RRN)		RRDS. <u>See</u> Relative record data set.	
	54,67-70,72	RRN. <u>See</u> Relative record number.	
Relative track and record number	54	Run unit (VS COBOL II)	38,46.1
RELEASE service routine		SAM. <u>See</u> System Accounting and	
--calling format	56	Measurement.	
--and exclusive control	59	SAM access method. <u>See</u> BSAM and QSAM.	
--function	51	Sample subroutine	
--return codes	56	(SQCOBOLB)	111,126-128
--and subsystem processing	52-53	Sample subsystems	42,111,167,197,247
--use of	56-57	Save area	105,243,244
--and VSAM	67,74	--chaining of	46.3
RELEX service routine	59,70	SBA sequences (IBM 3270)	21,27,49
RENT parameter (linkedit)		SBSP parameter (SYCTTBL macro)	18
	45,46.3,225,227	SCT. <u>See</u> Subsystem Control Table.	
REPORT macro	240	SECTEST macro (ESS)	110.1
REPTAPE. <u>See</u> Batch Report Table.		SECUSER routine (ESS)	15
Resident programs	11,46.3,92,105	Segmented messages	
RESET TRACE (COBOL)	46.2	--and COBPUT	97
RESOURC parameter (SYCTTBL macro)	19	--and DDQ	30
RESOURCE macro	19	--and FESEND	102
Resource Manager	12	--and MSGHQPR	22,76,97
RESTART parameter (SYCTTBL macro)	46.4	--and Output Utility	76
Restart/recovery	21,26,46.4	SELECT service routine	
Restarted messages	46.4,88	--calling format	56
Retriever (module)	29	--function	51
RETURN-CODE (COBOL data-name)	45	--and ISAM/VSAM compatibility	74
Return codes		--return codes	56
--from COBPUT	98	--and subsystem processing	52-53
--from CONVERSE	92	--use of	56
--from FECM calls	106	--and VSAM	67,70
--from FESENDC	103	SERVICE RELOAD (COBOL)	46.2,91
--from File Handler		Service routines (INTERCOMM)	
--BDAM	66	--calls to	38-39,45,46.2,95,105
--ISAM	63	SETGLOBE global table	46
--outline	55	Share options (VSAM)	70-71
--RELEX	59	Signed-on operator-id checking	15
--SELECT/RELEASE	56		
--sequential access	60		

	<u>Page</u>
SIMCRTA utility	168
Simulation Mode	
--and BTAM simulator	111,129-131,197
--execution JCL	138,217
--input messages	134-135
--linkedit JCL	136-137,215-216
--logging	131,161-166
--output from	139-160
SIM3270 module	129,131,197
--printout from	131,139-160
Single-thread processing	4,131,168,247
Snap 51	103
Snap 87	46.4
Snap 118	46.3
Snap 123	46.3
Snap 126	46,46.3
Snap program areas	46.3,46.4,110.1
Snaps, Test Mode	168,188-190
SNAPDD data set	46.4
Sort Facility	109-110
SORT (COBOL verb)	46.2
SPA. <u>See</u> System Parameter Area.	
SPAEXT. <u>See</u> System Parameter Area.	
SPALIST macro	17,46,82,240
SSCONV macro	110.1
SSRANGE checking	46.1,46.3
SSSTART macro	110.1
SSTEST macro	110.1
SSSTOP macro	110.1
SSUP command	46.3
STATION macro	240
Station Table (PMISTATB)	17,102,240,241
Statistics	6,26
STEPCAT DD statement (VSAM)	131
STOP command	46
STOP RUN (VS COBOL II)	46.1
Storage pools	46
STORE function	84-85
Store/Fetch facility	
--and conversational subsystems	81,84-85
--and CONVERSE	88
--described	15
--and MMU maps	130
--service routines	110
--use of	30,84-85
STRT command	46,139-140,197
SUBMODS macro	
46,46.1,46.3,105-106,110.1,130,240,243	
Subroutine entries table.	
<u>See</u> REENTSBS table.	

	<u>Page</u>
Subroutines	
--and COBREENT	38-39,46.3,95,105,243
--and CONVERSE	91
--defining via REENTSBS table	46,46.3,95,105
--and DWSSNAP Facility	46.4-46.5
--and dynamic linkedit	30
--interface to	
38-39,45-46,46.3,105-106,243-246,247	
--linkedit of	45,225,227
--sample	126-128
--and VS COBOL II	38,46.1-46.3,105
Subsystems	
--coding	31-46.5
--coding conventions	45
--conversational	
--and CONVERSE	88-92
--design considerations	93-94
--and Dynamic Data Queuing	86-87
--general concepts	79-80
--implementation	81
--and Store/Fetch	84-85
--and USERSPA	82-83
--defined	9-10
--and dynamic linkedit	30
--and file access functions	51
--and File Handler	52-55
--illustration of	32,42-44.2
--Intercomm-supplied	16
--linkedit of	45,225,227
--and loading above the 16M line	38-39,45-46,95
--and message processing	35-38
--and message switching	18,41,99
--nonreentrant	16,247-252
--and output messages	41,102,107-109
--parameters passed to	31-33,40-41
--reentrant	16,18,31-33,45-46.3
--return codes from	34
--sample inquiry	111,167,198
--structure	31-32,34
--testing	111,129-223
--time controlled processing	30
--and VS COBOL II	46.1-46.3,197
<u>See</u> also Reentrant subsystems.	
Subsystem Control Table (SCT)	
--creation of	18,240
--and dynamic storage	45
--function	17-18
--and message collection	100-101
--and message flow using EDIT and OUTPUT	24

	<u>Page</u>		<u>Page</u>
--and message flow using MMU	22	System log. <u>See</u> INTERLOG file.	
--and RESOURCE macro	19	System Monitor	12
--and subsystem entry point	40,46.2	<u>See</u> also Subsystem Controller.	
--and subsystem structure	31-32,248	System Parameter Area/List (SPA)	
--and subsystem testing	130,167,197	--and associated components	82-83
--and user-coded COBOL		--and conversational subsystems	81-83
subroutines	106	--creation of	82,240
Subsystem Controller		--function	17
--and COBPUT return codes	97-98	--and Message Collection	99
--and Dynamic-Work-Space	100,101	--and SPAEXT	240
--function	12	--and subsystem structure	31-33
--and File Handler return codes	55	--and system separator character	49
--and GOBACK	40	--and user-coded COBOL	
--and inter-subsystem queuing	97	subroutines	106
--and INTERLOG	28-29	--and USERSPA	82-83
--and message processing		System Parameter Table. <u>See</u>	
concepts	35	System Parameter Area.	
--and PREPROG	38	Systems Operation Control	6-7
--and return codes	34,41,97,99-100		
--and subsystem structure	31-34	TABEND function	96,110,237,238
Switched devices		TABGET function	96,110,237
--and Edit utility	49	Table Facility	81,88,96,109,110,237
--and MSGHUSR	21	--described	15
SYCTTBL macro		Table sort (INTSORT)	109-110
--and Dynamic Working Storage		Tables (Intercomm)	17,239-241
protection option	46	TABOPEN function	96,110,237
--and freeing dynamic-		TABPUT function	96,110,237
work-space	100,101	TABSORT function	96,110,237
--and Front End Queue Table	239	TABUILD function	96,110,237,238
--function	17,18	Task Global Table (COBOL)	
--and INTSCT	17,18,130,132,167,180,214	--and save area chaining	46.3
--and loading above the 16M line	46	--saving a copy of	95
--and Message Collection	100-101	TCAM	11,130
--and nonreentrant subsystems	18,247	TCTV parameter (SYCTTBL	
--parameters	18-19	macro)	19,110.1
--and RESOURCE macro	19	Teletype Terminal	21
--and restarted messages	46.4	Terminal control	6-7
--and sample inquiry subsystem	111	Test input messages	
--and Subsystem Control Table	17,18,240	111,130,134-135,168,183-184	
--and user-coded COBOL		Test Mode	26,111,129,167
subroutines	106,243	Test Mode message card formats	183
--and VS COBOL II	46.1,46.2,197	Testing Subsystems	111,129,167,197
SYCT400 module	29	TGT. <u>See</u> Task Global Table.	
<u>See</u> also Subsystem Controller.		THDCOM area (VS COBOL II)	95
SYMREL data set	82,225,229	Thread	
SYMUSR data set	82	--defined	4,16
SYSOUT data set (VS COBOL II)	217	--dumps of (debugging)	129
SYSPRINT data set	30	--and File Handler control areas	53
System Accounting and		--and MNCL	19,131,168,247
Measurement (SAM)	110.1	--and RELEASE	56
System components and programs		Thread number	21
11-15,30,110		Time-controlled processing	30
System control commands	30,93-94,102	Time-out	19,88,91

	<u>Page</u>
Timed delay in processing	30
TOTAL data base	15,52
TP queuing interface to Front End	12
Transaction codes/identifiers	17-18
Transactions	
--described	1-3
--sample inquiry	111
TTR. <u>See</u> Relative Track and Record Number.	
TUNERTBL table	240,241
TYPE parameter (SUBMODS macro)	46,46.3
Undefined records	61
UNLK command	93-94
UNSTORE function	85
USAGE parameter (SUBMODS macro)	46,46.3
User exits	30,34,46.2
User logging.	26-27
<u>See</u> also LOGPUT.	
USERSPA	81-84,88
USRBTLOG module (user exit)	29
USRBTVRB	18,130,132,168,214
USRCANC module (user exit)	34,55,100
USRSTS	18,130,132,167,180,214
USRSUBS	97,110.1,130,132,214
--and user subroutines	105,243
USRTRACK macro	110.1
USRVERBS	167,180
Variable-length SAM records	61
Variable-length VSAM records	68
VCT macro	239
VERB macro	180,240
Verb Table. <u>See</u> Front End Verb Table.	
VERBGEN macro	240
VERBTBL. <u>See</u> Edit Control Table.	
VMI (Verb Message Identifier). <u>See</u> MSGHVMI field.	
VS COBOL II	
--and abends from	46.3
--Cobpacks for	216
--and COBREENT	38,46.2,46.3,95
--compiler for	46.2,226,227
--compiler options	46.1,226-227
--and CONVERSE	46.3,88
--COPY statement	46.2,47
--described	46.1-46.3
--dynamic calls from	39,46.2
--and dynamic linkedit	46.3
--and Dynamic Working Storage (DWS)	46.1,46.2,46.3
--and DWSSNAP facility	46.3

	<u>Page</u>
--and File Handler calls	54,56,57
--and linkedit	45,46.1,46.3,227
--and loading above the 16M line	45-46,46.1,46.3,197,227
--and Message Mapping Utilities	47,197
--and message switching	100
--and multithreading	16
--and nonreentrant programs	46.2,247
--and OPTIMIZE compiler optionn	46.1
--parameters to subroutines	46.2,95,105
--and PREPROG	38
--and REENTSBS codes	46.2,54,95
--run unit of	38,46.1
--save area chaining	46.3
--sample program (SQCOBOLV)	198-213
--and single-threading	46.2
--and snaps	46.3
--SSRANGE checking	46.1,46.3
--and SUBMODS macro	46.1,46.3
--and subroutines (user)	46.1,46.2,46.3
--subsystems	16,46.1,46.3
--and SYCTTBL macro	46.1,46.2,46.3,197
--testing of	111,197-223
--and THDCOM area	95
--and VALUE definitions	46.2
--and variable fields	41,46.1,197
VSAM access method	
--DD statement parameters	67
--and Dynamic File Allocation	15
--FAR options for	67,71
--File Handler processing	
--alternate index, keyed files	71
--alternate path processing	71
--call summary	72-73
--data set name sharing	67,71
--ESDS file loading	56,67,70
--example of use	111
--exclusive control	59,70
--file types supported	67
--GETV and PUTV	67-68
--LSR buffer pools	67,71
--processing options	69-70
--reason codes from VSAM	55,70,73
--File Handler Service routines	51-52,54,67-68
--ISAM/VSAM compatibility	67,70,73,74
--POINT processing	69
--sample inquiry subsystem	111
--sequential access	69
--SHAREOPTIONS	70,71
VTAM	11,103,108,130
--OTQUEUE user exit	21
VTSAMP table	239

	<u>Page</u>
Working-storage	40-41,54,243,249
WRITE service routine	
--calling format	60,62,64
--and direct access files	64-65
--and exclusive control	57-59
--function	51
--and indexed sequential files	62-63
--and ISAM/VSAM compatibility	70,74
--and sequential access files	60-61
--and subsystem processing	52
WRITEOVER (FAR) option	56
--and VSAM ESDS files	67,70
XA/ESA (loading above the 16M line)	
--and COBOL programs	38-39,45-46
--and COBREENT calls	39,46,95
--and CONVERSE	88
--extended storage loading	
requirements	45-46
--and FECMDDQ	107
--and File Handler calls	54-57
--and Message Mapping Utilities	111
--and nonreentrant programs	247
--and program linkedit JCL	45,225,227
--and subroutine interfaces	46,95,105
--and subsystem coding	41,45-46,111
--and VS COBOL II	46.1,46.3,227