INTERCOMM

PL/1
PROGRAMMERS GUIDE

ISOGON
CORPORATION
330 Seventh Avenue, New York, New York 10001

LICENSE: INTERCOMM TELEPROCESSING MONITOR

PL/1 Programmers Guide

Publishing History

Publication	Date	Remarks
First Edition	April 1974	This manual corresponds to Intercomm Release 6.0.
Second Edition	February 1991	This manual corresponds to Intercomm Releases 9.0 and 10.0.

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
- Direct calls via INTLOAD from loaded programs to user subroutines
- DWSSNAP Facility (online debugging/DSA snaps)
- VSAM data set access under Dynamic File Allocation (DFA)
- Subsystem message flushing
- GETDATE macro.

The following are desupported under Release 10:

- AMIGOS file access method
- DISAM file access method
- PL/1-F compiler.

PREFACE

Intercomm is a state-of-the-art teleprocessing monitor system executing on the IBM System/370 family of computers and operating under the control of IBM Operating Systems (MVS/370, 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 PL/1 Programmers Guide explains the organization of Intercomm from the application programmer's point of view and illustrates the procedures for creating PL/1 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 Users Guide
- Dynamic Data Queuing Facility
- Page Facility
- Operating Reference Manual: "Message Management"
  "File Management"

LIST OF ILLUSTRATIONS

Figure Page

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 of 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.

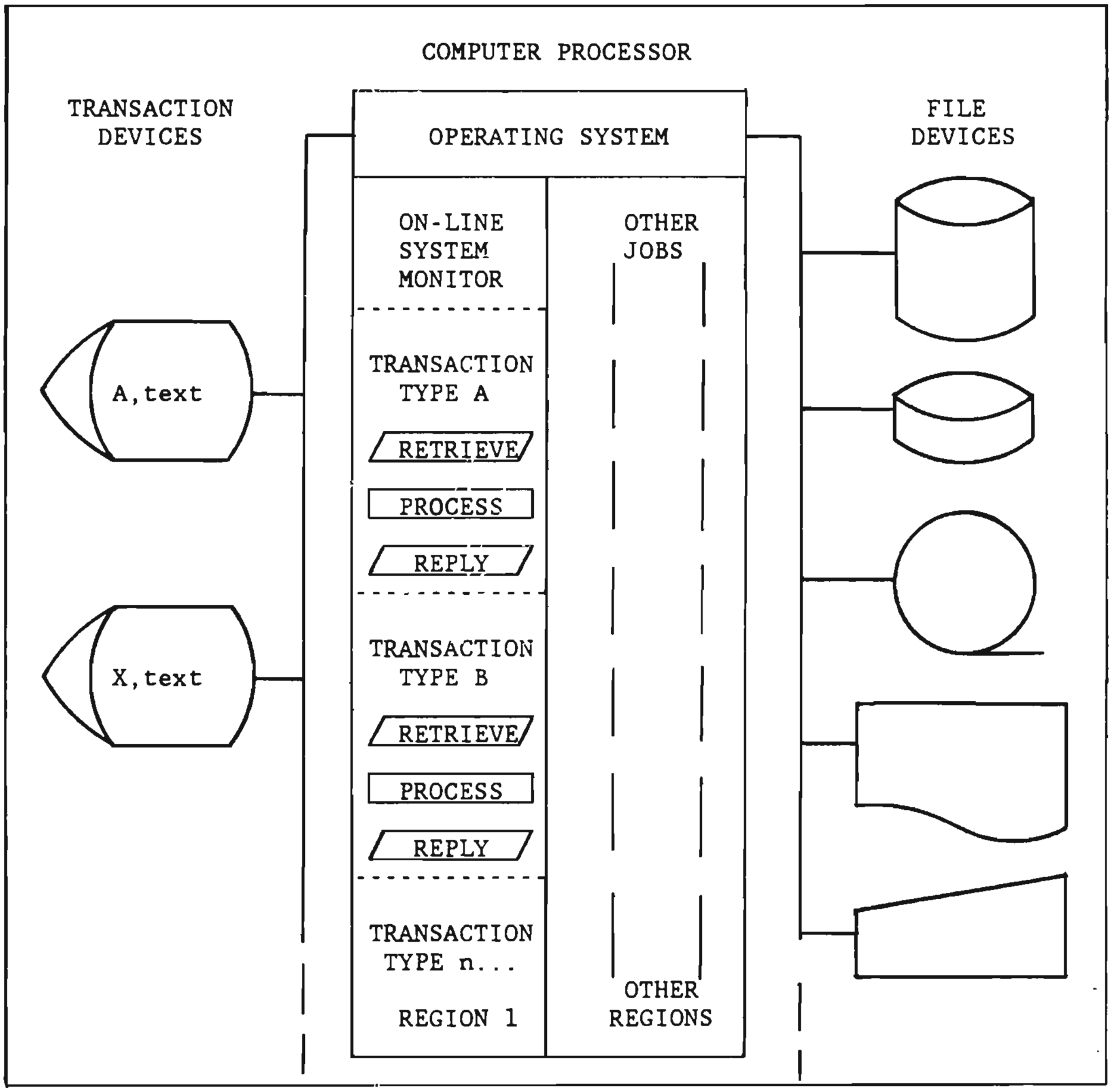


Figure 1. On-line Transaction Processing in a Multiprogramming Environment

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.

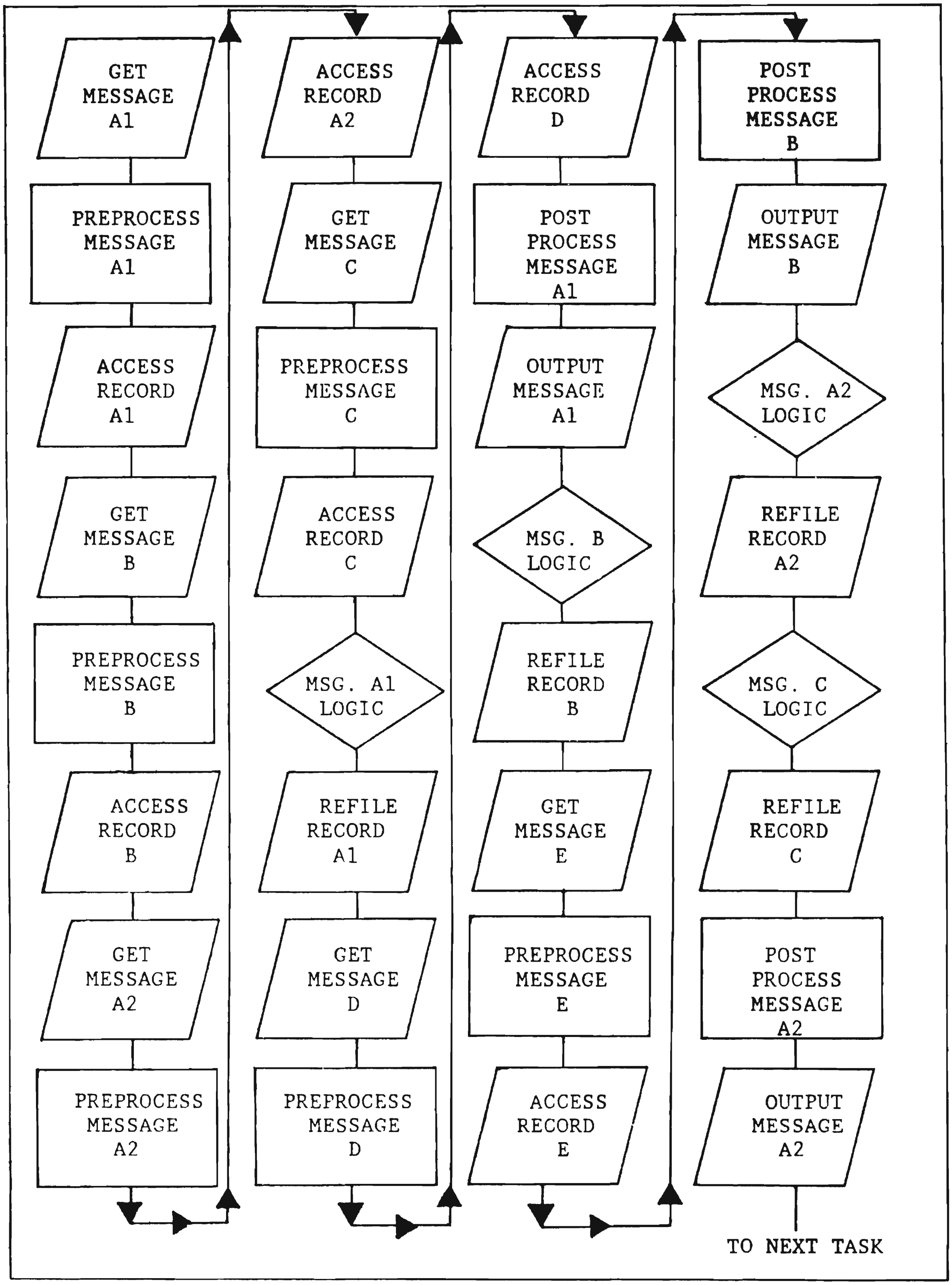


Figure 3. Multithreading in an On-line Environment

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.

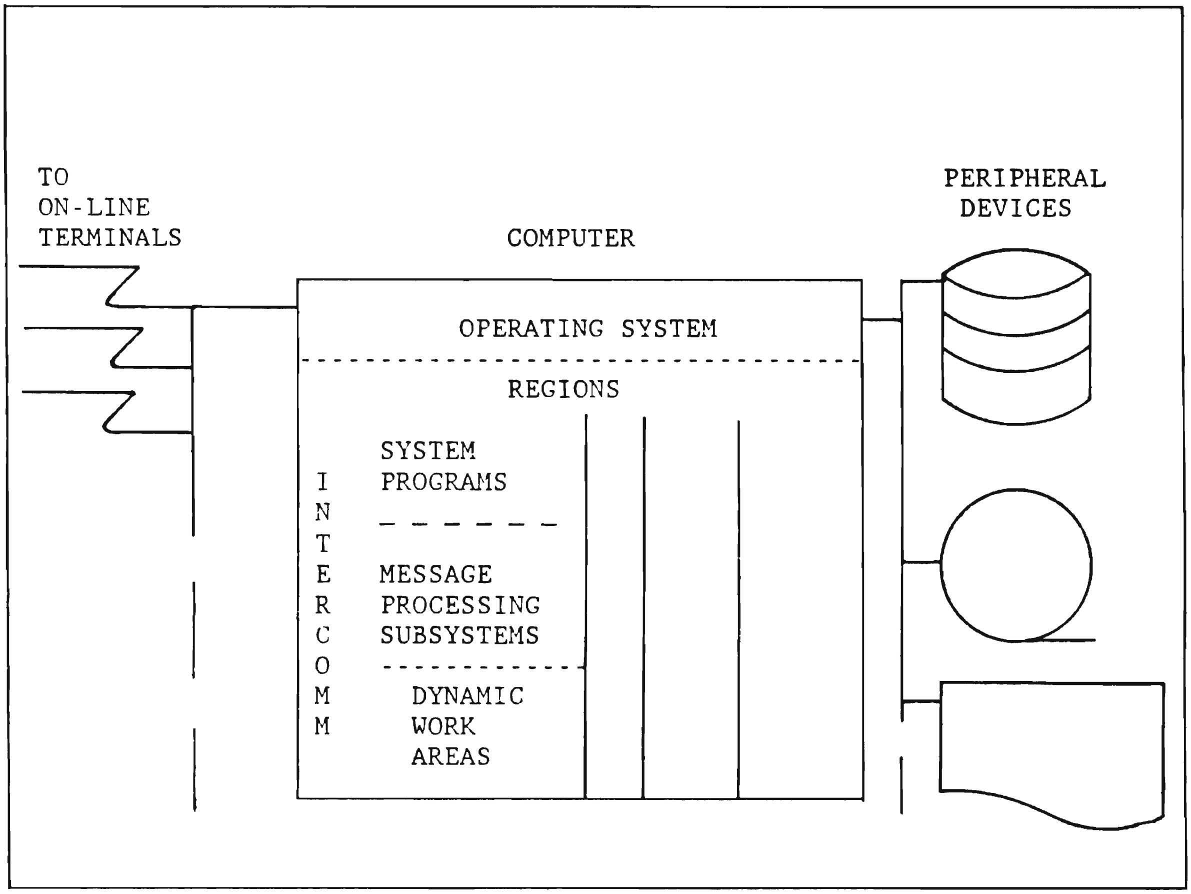


Figure 4. The Intercomm Environment

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.

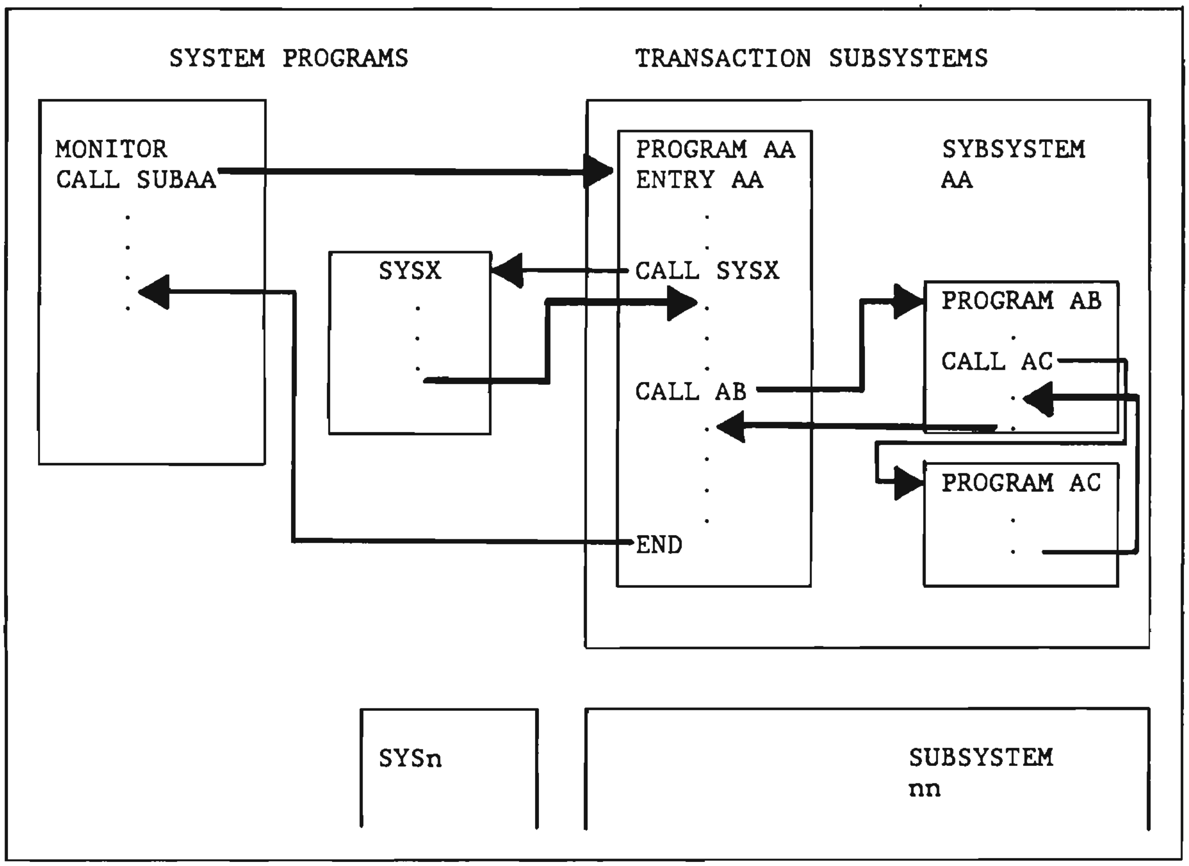


Figure 5. Intercomm Control Sequence

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.

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.

2.3 SYSTEM PROGRAMS

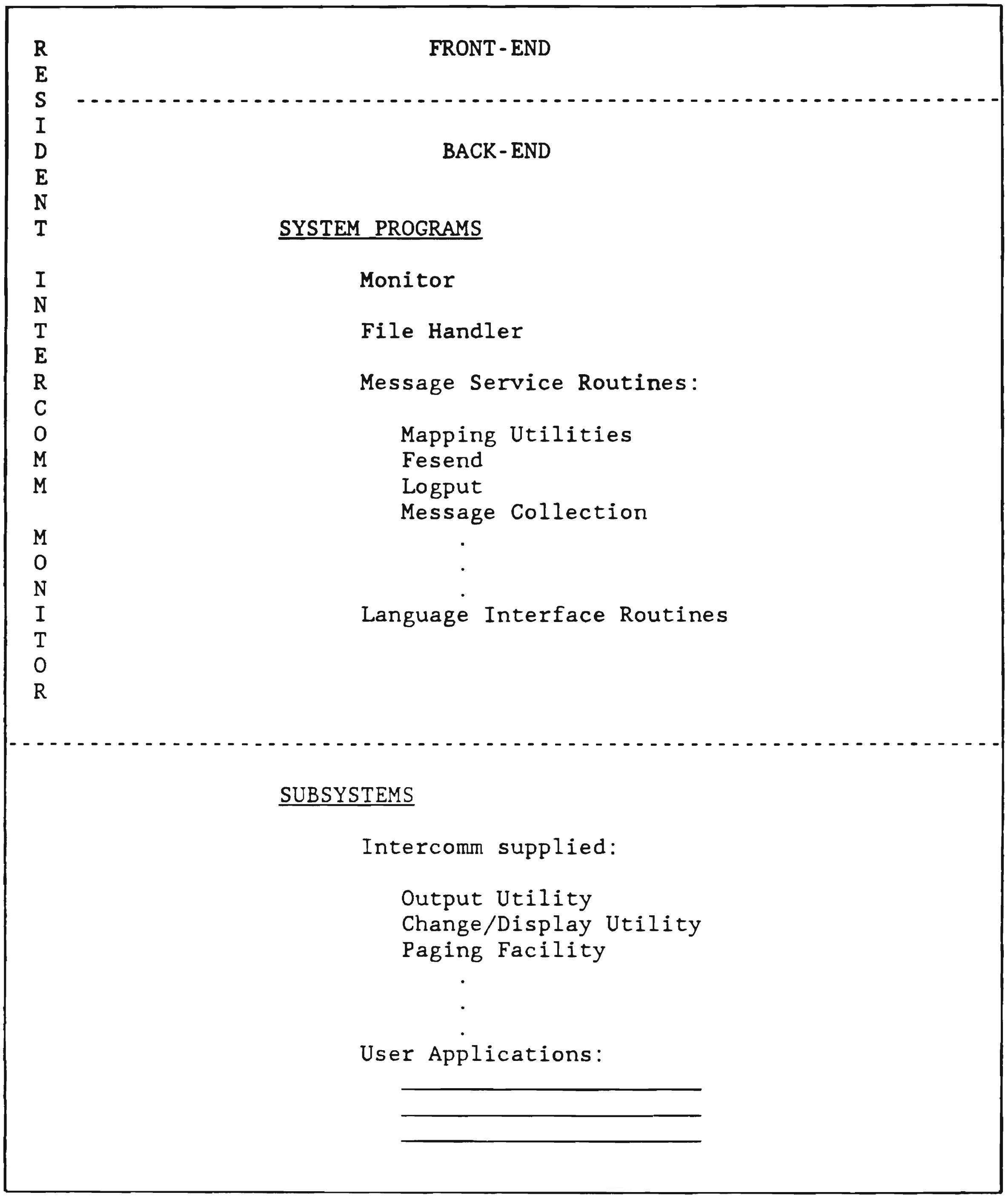
Intercomm system programs are written in Assembler language and include the Monitor, File Handler, high-level language interface routines to maintain reentrancy, and message processing service routines.

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.



RESIDENT INTERCOMM MONITOR

FRONT-END

BACK-END

SYSTEM PROGRAMS

Monitor

File Handler

Message Service Routines:

Mapping Utilities
Fesend
Logput
Message Collection
.
.
.

Language Interface Routines

SUBSYSTEMS

Intercomm supplied:

Output Utility
Change/Display Utility
Paging Facility
.
.
.

User Applications:

Figure 6. Intercomm System Components

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 even a batch program. (See Dynamic Data Queuing.)

- CRT Page Facility

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

- Data Base Management System Support (DBMS)

  This facility consists of separate service routines for each supported DBMS (IDMS, System 2000, Model 204, ADABAS, TOTAL, DL/I, or a user DBMS), which allows access to the DBMS from Intercomm. (See the 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, 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.)

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 file of output display screens created by the PAGE system program. Subsystems make use of this feature by calling the page storage 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 as AUTOMATIC and processed in a dynamic storage area (DSA) obtained for the exclusive use of one processing thread. Since PL/1 manages its own storage depending upon variable characteristics and attributes, an interface is provided which obtains a unique area of storage (from Intercomm administered core pools) for each iteration (thread) of each PL/1 subsystem. The storage area is passed to the PL/1 program for use as an ISA (Initial Storage Area). Further details of this interface, 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 BTVERB 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 BTVERB 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 PL/1 subsystems are:

- LANG=RPL1

  For reentrant PL/1 subsystems (REENTRANT coded for OPTIONS on mainline PROC statement); LANG=PL1 if nonreentrant.

- SBSP=xxxxxxxx or LOADNAM=xxxxxxxx (for dynamic load)

  Specifies the subsystem entry, that is, the main PROC name of the PL/1 subsystem (SBSP), or the load module name (LOADNAM).

- SPAC=nnnnnn

  Specifies for PL/1 subsystems, the amount of ISA the interface routine PREPLI should acquire, clear to binary zeros, and pass to the PL/1 initialization routine (PLICALLB) for each message being passed to this subsystem (program DSA size, plus DSA sizes of called user PL/1 subroutines, plus 2000 bytes). Use PL/1 compiler STORAGE option to determine DSA sizes.

- PL1LNK={BASED   }
  {NONBASED}

  Indicates whether the parameters passed by Intercomm to the PL/1 subsystem are to be in the non-standard 'BASED' form, whereby the program expects to receive them in the form of dummy arithmetic scalars, or whether the program expects the parameters in the form of Locator/Descriptors for standard character strings (default). Further details of program linkage techniques are described in Chapter 3.

- 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 SCT) 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 (HHMMSSTH)	N
MSGHTID	5	Terminal identification (originating terminal on input messages, destination terminal on output) or Broadcast Group name	Y
MSGHCON	2	Reserved area	N
MSGHCON+1 (MSGHRETN)	(1)	Subsystem return code (for log code X'FA' entries only)	N
MSGHFLGS	2	Message indicator flags	N
MSGHBMN	3	Front End message number-Rel 10 (binary)	N
MSGHSSCH	1	High-order sending subsystem code	M
MSGHUSR	1	Reserved***	L
MSGHADDR	2	Used for special processing by the Front End (MSGHBMN - Rel 9)	N
MSGHLOG	1	Log code (see Figure 11)	L
MSGHBLK	1	Reserved area	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 messages to be processed by the Edit Utility, contains a C'F' to indicate that the input message was from a 3270 CRT and contains SBA sequences.

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 <u>BTAM Terminal Support Guide</u>).

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 PL/1 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 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.

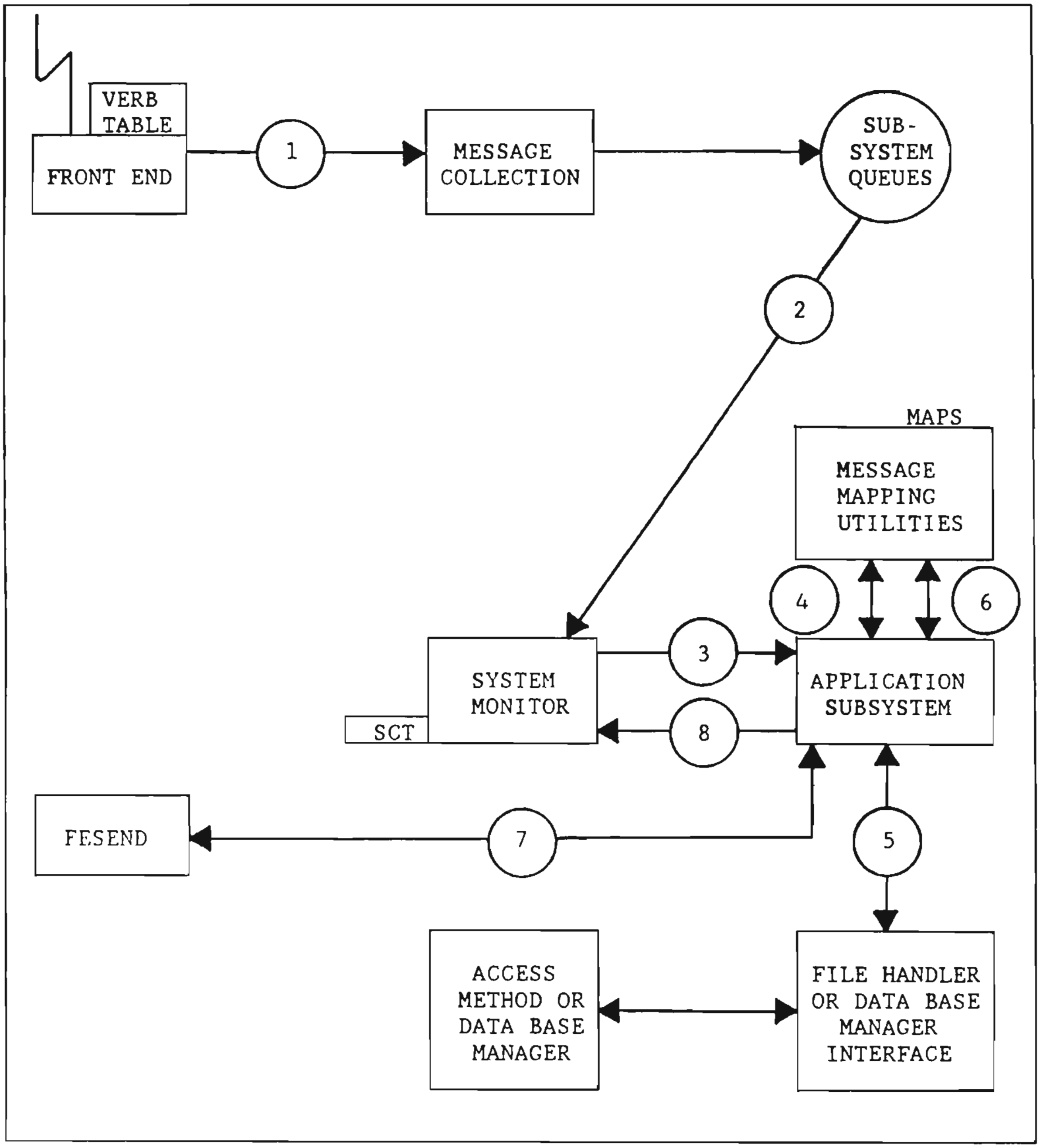


Figure 8. Intercomm Message Flow Using Message Mapping

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.

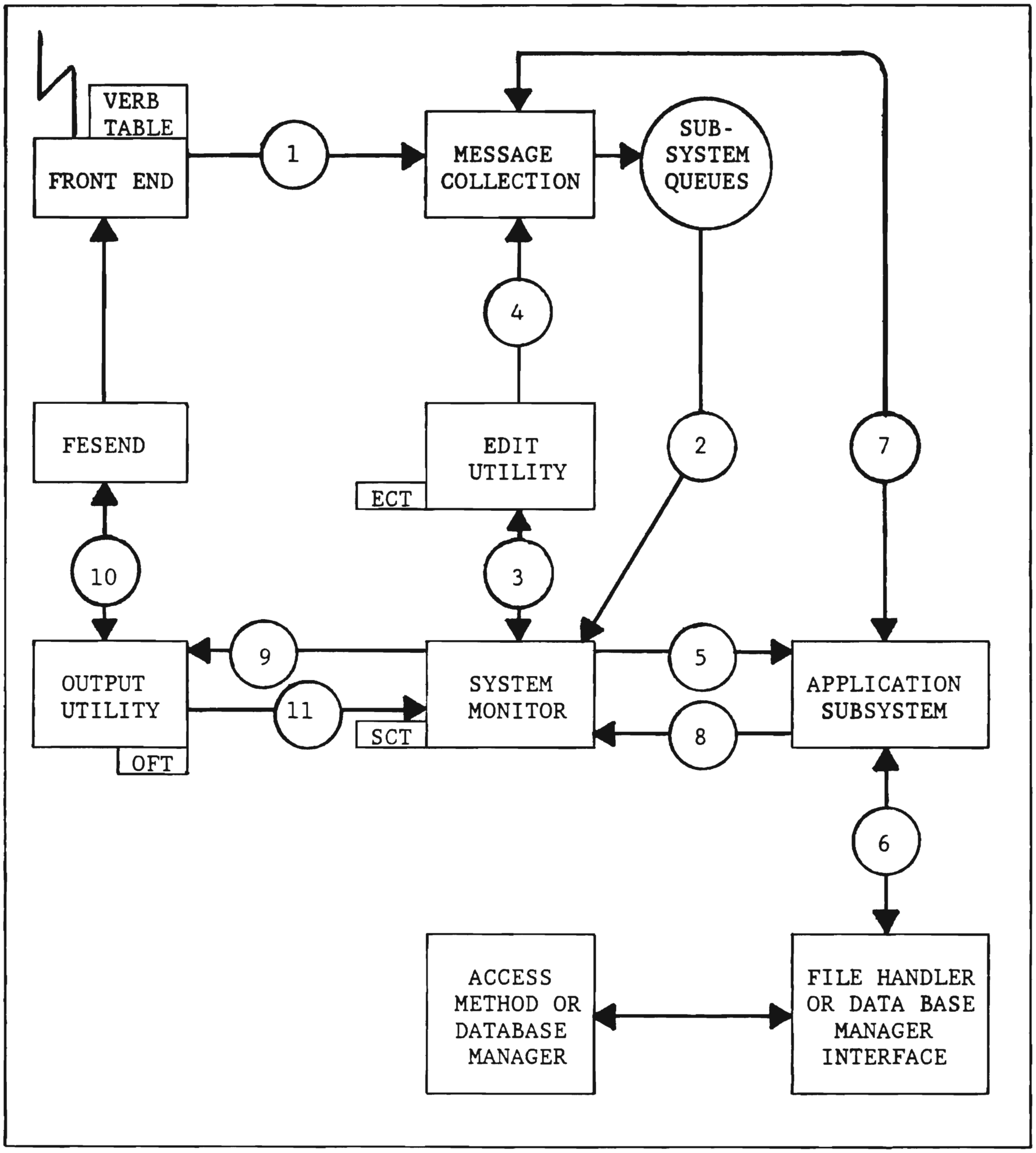


Figure 9. Intercomm Message Flow Using Edit and Output

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.

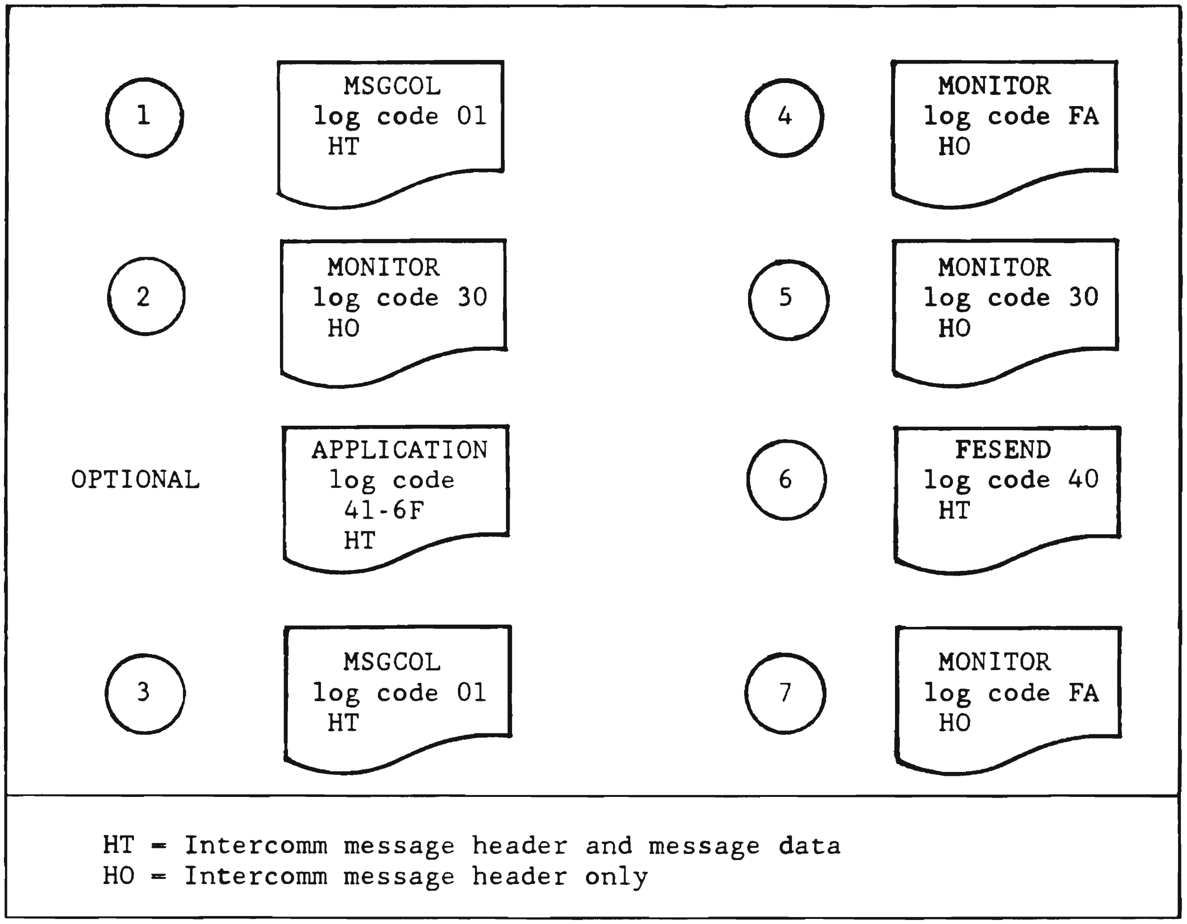
A timing analysis utility (Log Analysis), which is supplied with Intercomm, may be used off-line to produce a report of message queuing and processing time. Statistics for messages by terminal, verb, subsystem, and/or system totals are provided. See the Operating Reference Manual.

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 <u>before</u> 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).



HT = Intercomm message header and message data
HO = Intercomm message header only

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
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'C0'	C0	HT	Region started (Multiregion only) (Text=Region-id(s))	MRINTER	No
C'A'	C1	HT	Message successfully queued for Satellite Region	MRQMNGR 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	FESEND	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-F6 F7-F8	HO HT	Transmitted DDQ msg status: see SNA Term. Support Gd.	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 linkedited to called Intercomm resident 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.

Chapter 3

CODING AN INTERCOMM SUBSYSTEM IN PL/1

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. Although the PL/1 subsystem is a subroutine to Intercomm, it should be defined as a MAIN procedure in the PL/1 environment. The parameters must be defined to the PL/1 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 arithmetic variable (FIXED BIN(31)) into which the subsystem must place an appropriate Intercomm return code before returning control to Intercomm.

The first three of these parameters may be defined as character strings or as pointer variables (address of Locator/Descriptors for simple character string areas in parameter list), or as dummy arithmetic variables which actually are the addresses of the character strings, depending on the coding of the SYCTTBL macro PL1LNK parameter (see Section 3.7 for further details).

Figures 12 and 13 illustrate a reentrant PL/1 subsystem with parameters defined as pointer variables (most common and easiest usage). A precise definition 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. If so, the parameters would be declared as structures defining the individual fields within the table areas as required by the subsystem. Structures defined for the IN_MSG and OUT_MSG areas would be required to assist with message manipulation: for this purpose, a member called PLMSGHD is provided, which declares the fields of the Intercomm message header as level 5 entries within a structure (see Figure 17).

```
 EXAMPLE1:  PROC (INMSG_PTR,SPA_PTR,SCT_PTR,ICOM_RC)
                 OPTIONS(MAIN,REENTRANT);
   /*           DEFINE THE PASSED PARAMETERS:                      */
 DCL   (INMSG_PTR,SPA_PTR,SCT_PTR) POINTER;
 DCL   IN_MSG       CHAR(4096) BASED INMSG_PTR;    /* INPUT PARM 1 */
 DCL   SPA          CHAR(500)  BASED SPA_PTR;      /* INPUT PARM 2 */
 DCL   SCT          CHAR(100)  BASED SCT_PTR;      /* INPUT PARM 3 */
 DCL   ICOM_RC      FIXED BIN(31);                 /* INPUT PARM 4 */
   /*           DEFINE STATIC STORAGE AREAS:                       */
   /*     THESE AREAS SHOULD HAVE THE INITIAL ATTRIBUTE            */
   /*     AND NOT BE MODIFIED.                                     */
   /*                                                              */
 DCL  VMI_57      BIT(8) ALIGNED  INIT('01010111')    STATIC;
 DCL  RSC_OUTPUT  BIT(8) ALIGNED  INIT('11100100')    STATIC;
 DCL  RSCH_OUTPUT BIT(8) ALIGNED  INIT('11100100')    STATIC;
 DCL  FILE_NAME   CHAR(8)         INIT('MYFILE  ')    STATIC;
                                 .
                                 .
                                 .
   /*           DEFINE VARIABLE STORAGE AREAS:                     */
   /*     THESE AREAS WILL BE DEFINED IN AUTOMATIC STORAGE         */
   /*     AND WILL BE ASSIGNED FROM THE PROVIDED ISA.              */
   /*     THERE WILL BE ONE SET OF AREAS FOR EACH MESSAGE          */
   /*     THREAD INVOKED.                                          */
   /*                                                              */
 DCL  OUT_MSG             CHAR(2048);                /* OUTPUT MSG */
 DCL  I,J                 FIXED BIN(15);             /* COUNTERS   */
 DCL  FILE_RECORD_AREA    CHAR(200);                 /* READ AREA  */
 DCL  ICOM_RETURN_VALUE   FIXED BIN(31);             /* RETURN CODE*/
                                 .
                                 .
                                 .
   /*           NOW DEFINE PROCESSING PROGRAM LOGIC.               */
   /*                                                              */
1 MAINLINE: DO;
     ICOM_RC = 0;               /* INIT THE INTERCOMM RETURN CODE */
     .
     .                                      Program Processing Logic
     .
     ICOM_RC = ICOM_RETURN_VALUE;         /* SET ICOM RETURN CODE */
     RETURN;
 END EXAMPLE1;
```

Figure 12.   Reentrant PL/1 Subsystem Structure

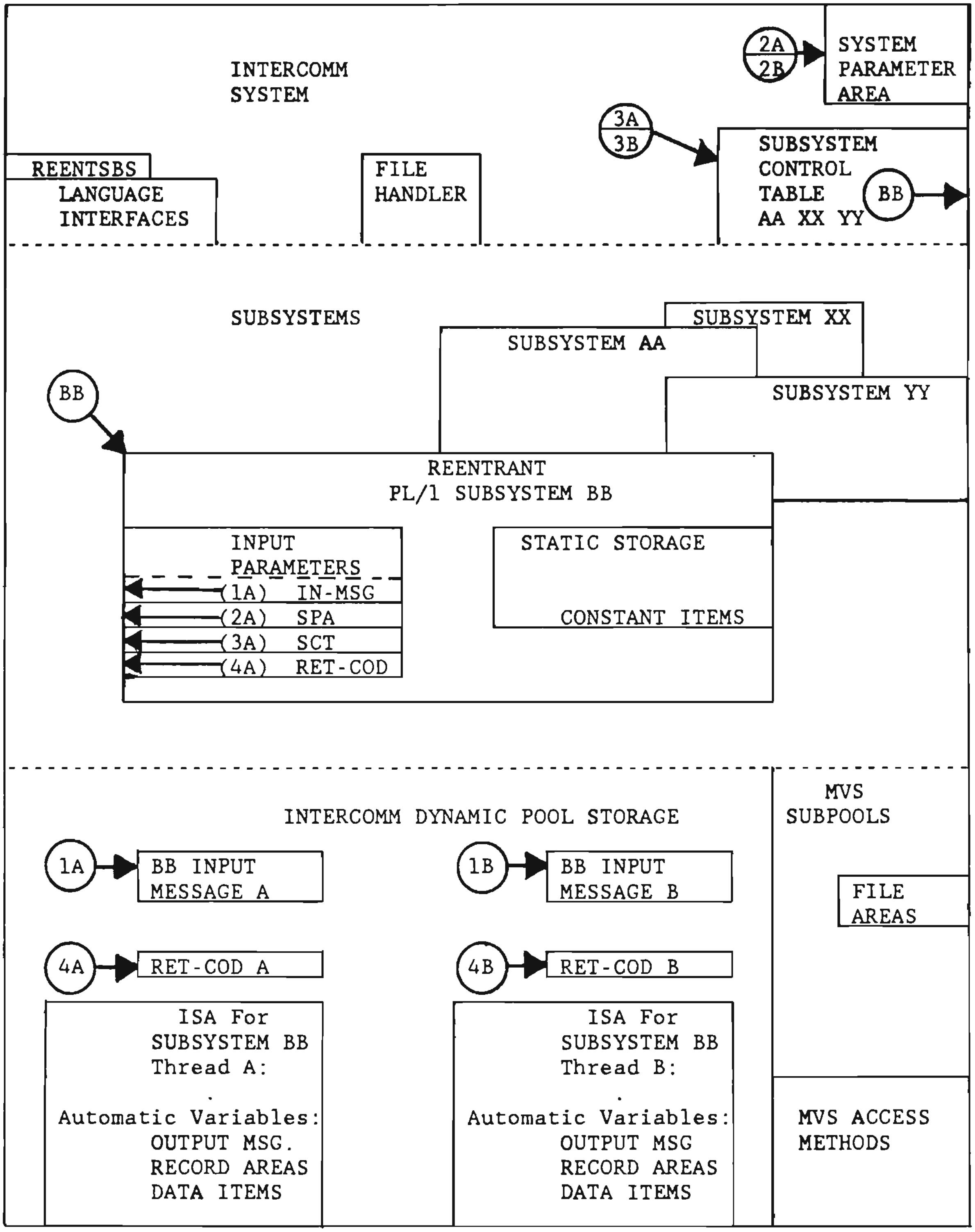


Figure 13. Reentrant Application Program Environment.

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), 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	(Not used, reserved)	---
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
900**	Successful completion	None
912	Force Backout-on-the-Fly*	File updates or additions backed out

*See File Recovery Users Guide or Data Base Management System Users Guide

**Used only when a called Assembler Language subroutine (MSGCOL/FESEND) has requeued or freed the input 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

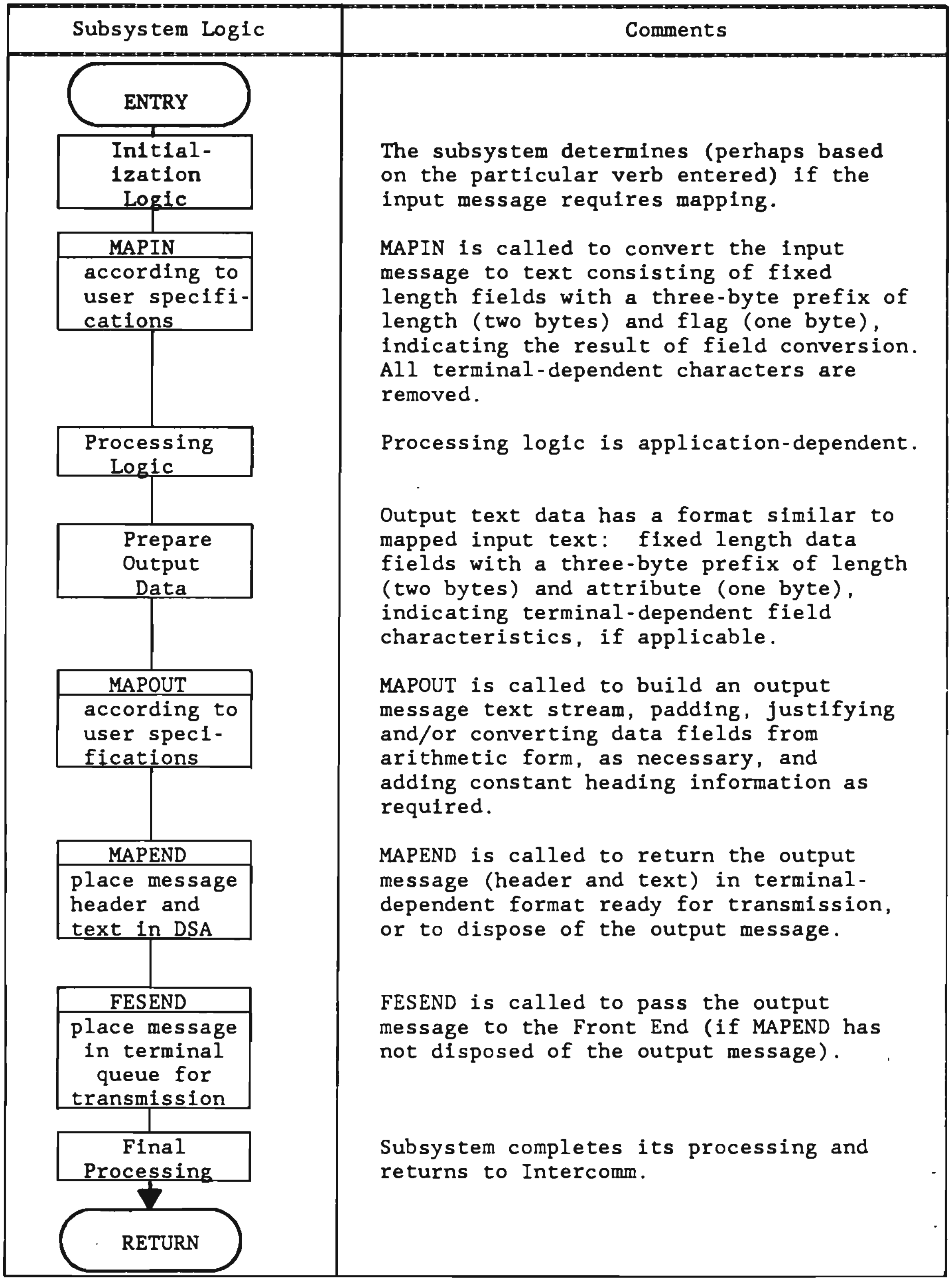
3.2 MESSAGE PROCESSING CONCEPTS

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 PREPLI 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.



The subsystem determines (perhaps based on the particular verb entered) if the input message requires mapping.

MAPIN is called to convert the input message to text consisting of fixed length fields with a three-byte prefix of length (two bytes) and flag (one byte), indicating the result of field conversion. All terminal-dependent characters are removed.

Processing logic is application-dependent.

Output text data has a format similar to mapped input text: fixed length data fields with a three-byte prefix of length (two bytes) and attribute (one byte), indicating terminal-dependent field characteristics, if applicable.

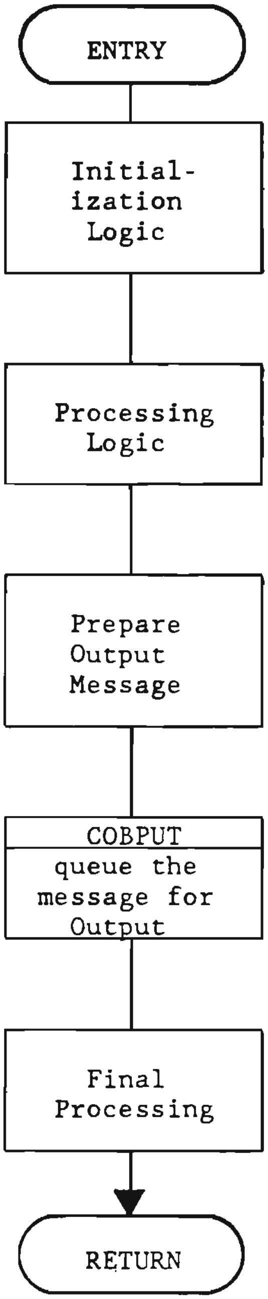
MAPOUT is called to build an output message text stream, padding, justifying and/or converting data fields from arithmetic form, as necessary, and adding constant heading information as required.

MAPEND is called to return the output message (header and text) in terminal-dependent format ready for transmission, or to dispose of the output message.

FESEND is called to pass the output message to the Front End (if MAPEND has not disposed of the output message).

Subsystem completes its processing and returns to Intercomm.

Figure 15. Subsystem Logic Using Message Mapping Utilities



Subsystem Logic	Comments
ENTRY	
Initialization Logic	If the Front End Verb Table indicates EDIT=YES, the subsystem receives an edited input message automatically. The message text consists of fixed length data fields <u>or</u> variable length data fields prefixed with a 1-byte identification and a 1-byte length code (binary values).
Processing Logic	Processing logic is application-dependent.
Prepare Output Message	The subsystem prepares an output message by creating a message header and the appropriate text. Output message text fields are either fixed length data fields or variable length fields with a prefix as described for Edit, above. Message header fields RSCH, RSC, and VMI identify the specific message text format.
COBPUT queue the message for Output	COBPUT is called to queue the output message for processing by the Output Utility subsystem.
Final Processing	Subsystem completes its processing and returns to Intercomm.
RETURN	

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

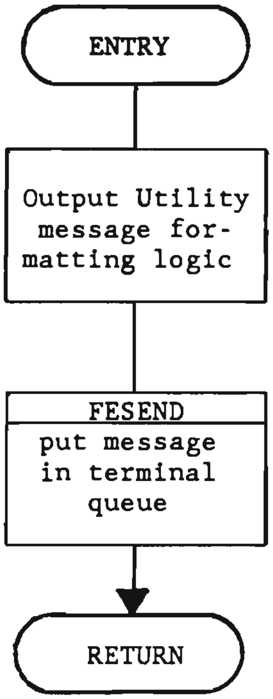
Subsystem Logic	Comments
 ENTRY Output Utility message formatting logic FESEND put message in terminal queue RETURN	The Output Utility performs message formatting according to user specifications, adding constant heading information as required. FESEND is called to pass the output message to the Front End. Output completes its processing and returns to Intercomm.

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

3.3 SUBSYSTEM CODING

The language interface routines are:

- PREPLI--which interfaces the Subsystem Controller to the PL/1 subsystem by initializing the reentrant 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 PREPLI returns to the Subsystem Controller (subsystem not activated). If the subsystem is loaded above the 16M line under XA or ESA, it will receive control in 31-Amode.

  PREPLI optionally supports the PL/1 execution parameters STAE, SPIE and REPORT. FLOW, COUNT, HEAP and TEST (PL/1 V2) are not supported. By default, SPIE and STAE are not used so that Intercomm recovery code receives control and allows the on-line system to continue execution (NOSPIE option), or to gracefully clean up (NOSTAE option) after an abend. The REPORT option is useful only in a test environment to determine the total ISA needed for subsystem execution. To use the REPORT option, a DD statement for the PLIDUMP (SYSOUT) data set (not SYSPRINT) must be added to the Intercomm execution JCL (after the PMISTOP DD statement); see the Operating Reference Manual for option implementation.

- PLIV--a 'top hat' linked with loaded PL/1 subsystems to provide entry points (PLICALLB and subsystem program code via PLIMAIN) to PREPLI. See Appendix A for loaded subsystem linkedit.

- INTLOAD--linked with dynamically loaded PL/1 programs to provide Intercomm service routine and user subroutine linkage, especially if program loaded above 16M line.

- PMIPL1--optional interface, which maintains linkages and save areas (and performs Amode switching under XA or ESA), from PL/1 programs to Intercomm service routines and/or to user subroutines (resident or dynamically loaded). PMIPL1 preserves the multithreaded reentrant PL/1 environment while providing a standard CALL interface to the routines. Note however, that all parameters passed to PMIPL1 (except as noted in Appendix D) must be character data (cannot have arithmetic attributes).

- COBPUT--which is called via PMIPL1, or directly, to copy a message from the automatic storage of a PL/1 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. (Required if PMIPL1 is used).

PL/1 subsystems may directly call Intercomm service routines and user subroutines using standard CALL statements, by declaring the routines as ENTRY OPTIONS (ASM INTER). A member PLIENTRY, listed in Appendix B, provides such declarations for the most commonly used Intercomm service routines. PLIENTRY may be copied into the PL/1 program via a %INCLUDE statement. User-coded PL/1 subroutines may also use this same interface scheme.

If the routines are not called directly, then one routine only is called: PMIPL1, which is declared as ENTRY EXTERNAL. 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 must be in Automatic Storage if the subsystem (subroutine) can be loaded above the 16M line (must be a 24-bit address). Coding format:

```
CALL PMIPL1(routine-code,parm1[,parm2,...]);
```

Subsequent chapters of this manual, and of related message processing facility manuals, contain detailed descriptions of applicable routine-code names and the parameters required for each routine. The Intercomm source text member PENTRY, listed in Appendix B, provides the definition of the halfword routine-code constants (FIXED BIN(15)) used for calling most of the Intercomm service routines via PMIPL1. To ensure that the correct code value is used, PENTRY should be copied into the static storage section of each PL/1 program using a %INCLUDE statement. Routine-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).

For calls to other Intercomm service routines and for user subroutines, add the names and index values to PENTRY and add corresponding entries to the REENTSBS table (see Section 9.1) if the PMIPL1 interface is used, or add the names to PLIENTRY, if the routines are called directly. User subroutine interface is further described in 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. Note that the input parameters are declared as simple character strings.

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 binary zeros - null bit string).

   - 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 give a value to the Intercomm return code field and return to the Subsystem Controller.

The procedure entry point name must correspond to the subsystem entry point (load module name) described in the Subsystem Control Table.

The input-message entry parameter has been further defined 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 as detailed for the output message area (see comments in the sample program).

To assist the programmer in defining the message header, there is a source text member, PLMSGHD, listed in Appendix B. This member may be %INCLUDE'd within a structure defining the input and/or output message areas and is defined to declare level 5 entries within the structure. If the input message area is declared as a character string as in the sample program, a structure may not be used to detail areas of the message; only DEFINED statements may be used as illustrated in the sample program (to prevent program execution errors). A structure may be declared for the input message if the parameter is defined as a pointer (see Figure 12) and the structure is BASED on the pointer.

The entry parameters for the System Parameter Area (SPA) and Subsystem Control Table entry (SCT) 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.

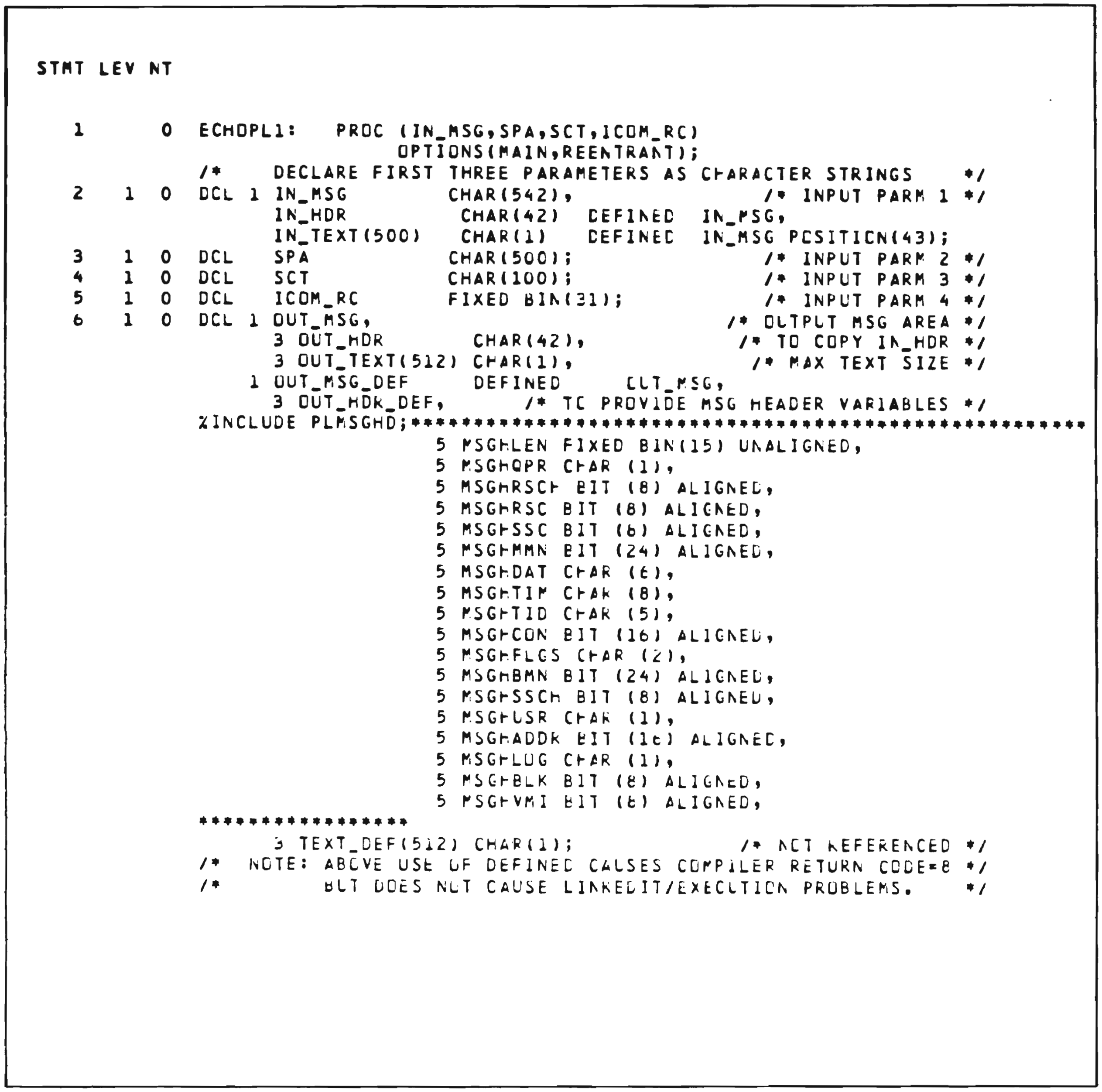
Constants are defined as STATIC items with the INITIAL attribute. All variables, and constants that may be passed as parameters, should be defined in (moved to) automatic storage, so that they are given unique areas for each message thread that is being processed. The allocation of automatic storage is done out of the ISA provided by the PREPLI interface routine, based on the SPAC parameter defined on the subsystem's SYCTTBL definition (see Chapter 2).

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.

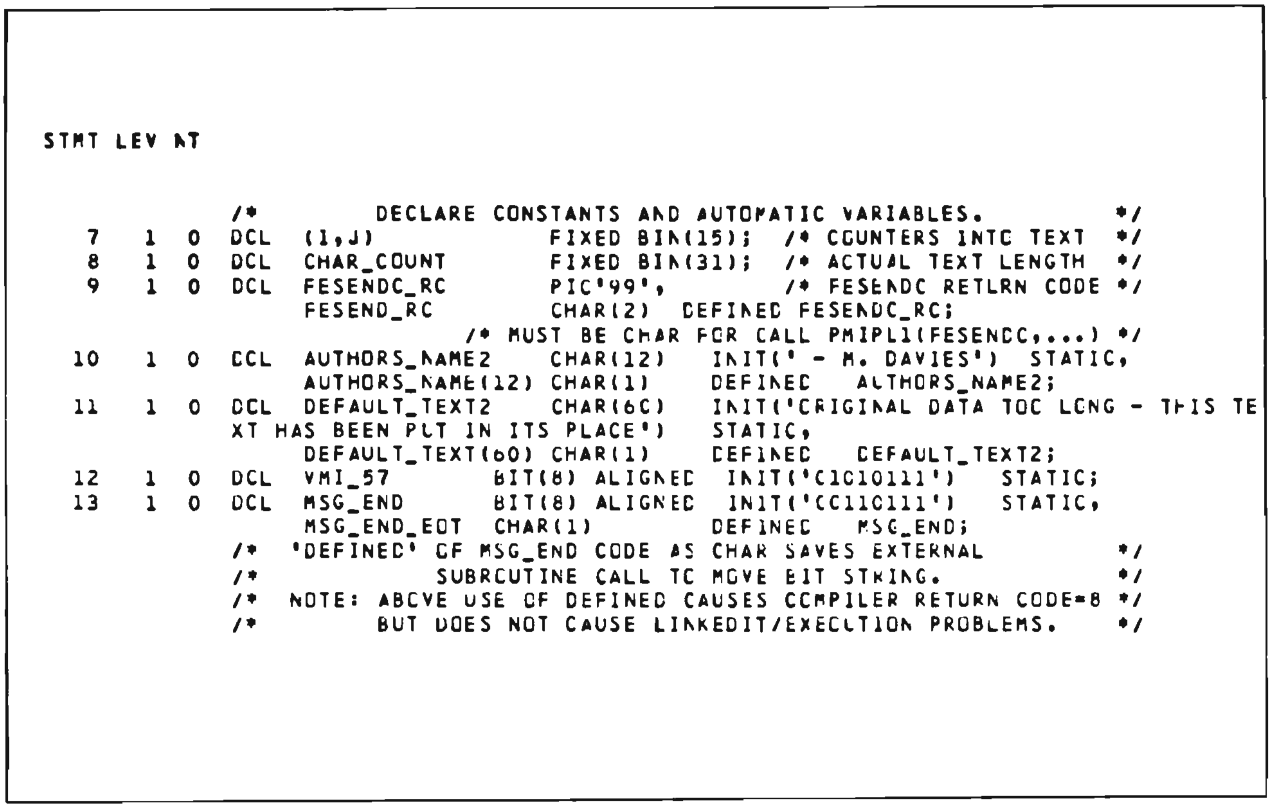
Message switching between subsystems is accomplished by moving the input message to an output message area and then changing the receiving subsystem code in the header and calling COBPUT as usual. The Verb/Message Identifier (MSGHVMI) may be initialized for interpretation by the receiving subsystem. A VMI equal to X'00' indicates that the Edit Utility is to be called by PREPLI prior to activating the receiving PL/1 subsystem.

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



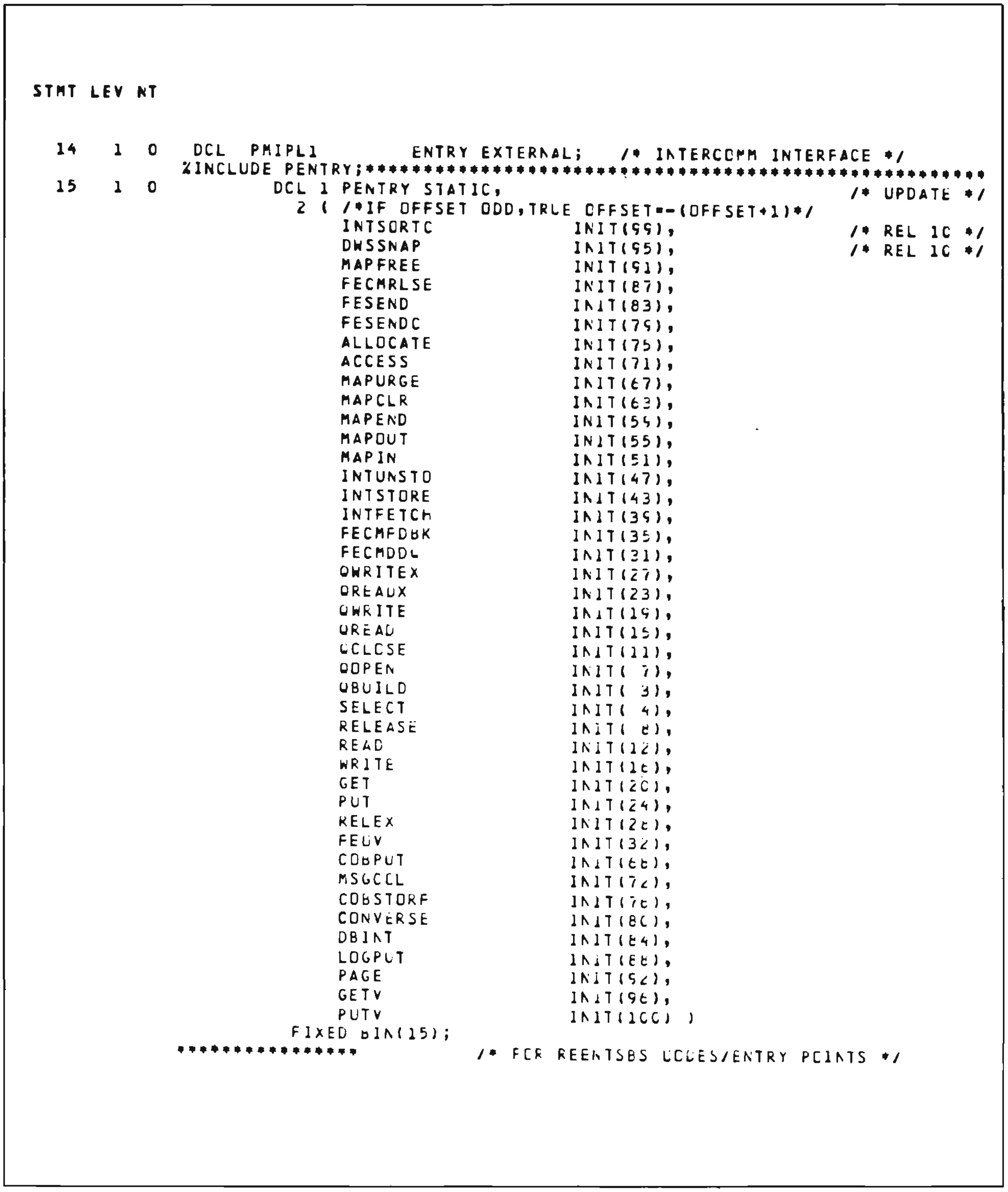
```
STMT LEV NT

  1       0  ECHOPL1:   PROC (IN_MSG,SPA,SCT,ICOM_RC)
                            OPTIONS(MAIN,REENTRANT);
             /*    DECLARE FIRST THREE PARAMETERS AS CHARACTER STRINGS    */
  2   1   0  DCL 1 IN_MSG        CHAR(542),               /* INPUT PARM 1 */
                   IN_HDR         CHAR(42)   DEFINED  IN_MSG,
                   IN_TEXT(500)   CHAR(1)    DEFINED  IN_MSG POSITION(43);
  3   1   0  DCL   SPA           CHAR(500);               /* INPUT PARM 2 */
  4   1   0  DCL   SCT           CHAR(100);               /* INPUT PARM 3 */
  5   1   0  DCL   ICOM_RC       FIXED BIN(31);           /* INPUT PARM 4 */
  6   1   0  DCL 1 OUT_MSG,                             /* OUTPUT MSG AREA */
                   3 OUT_HDR       CHAR(42),             /* TO COPY IN_HDR */
                   3 OUT_TEXT(512) CHAR(1),               /* MAX TEXT SIZE */
                 1 OUT_MSG_DEF     DEFINED      OUT_MSG,
                   3 OUT_HDR_DEF,       /* TO PROVIDE MSG HEADER VARIABLES */
             %INCLUDE PLMSGHD;************************************************************
                                  5 MSGHLEN FIXED BIN(15) UNALIGNED,
                                  5 MSGHQPR CHAR (1),
                                  5 MSGHRSCH BIT (8) ALIGNED,
                                  5 MSGHRSC BIT (8) ALIGNED,
                                  5 MSGHSSC BIT (8) ALIGNED,
                                  5 MSGHMMN BIT (24) ALIGNED,
                                  5 MSGHDAT CHAR (6),
                                  5 MSGHTIM CHAR (8),
                                  5 MSGHTID CHAR (5),
                                  5 MSGHCON BIT (16) ALIGNED,
                                  5 MSGHFLGS CHAR (2),
                                  5 MSGHBMN BIT (24) ALIGNED,
                                  5 MSGHSSCH BIT (8) ALIGNED,
                                  5 MSGHUSR CHAR (1),
                                  5 MSGHADDR BIT (16) ALIGNED,
                                  5 MSGHLOG CHAR (1),
                                  5 MSGHBLK BIT (8) ALIGNED,
                                  5 MSGHVMI BIT (8) ALIGNED,
             *****************
                   3 TEXT_DEF(512) CHAR(1);            /* NOT REFERENCED */
             /*  NOTE: ABOVE USE OF DEFINED CAUSES COMPILER RETURN CODE=8 */
             /*        BUT DOES NOT CAUSE LINKEDIT/EXECUTION PROBLEMS.    */
```

Figure 17. Echo Message Example; Reentrant PL/1
(Page 1 of 4)



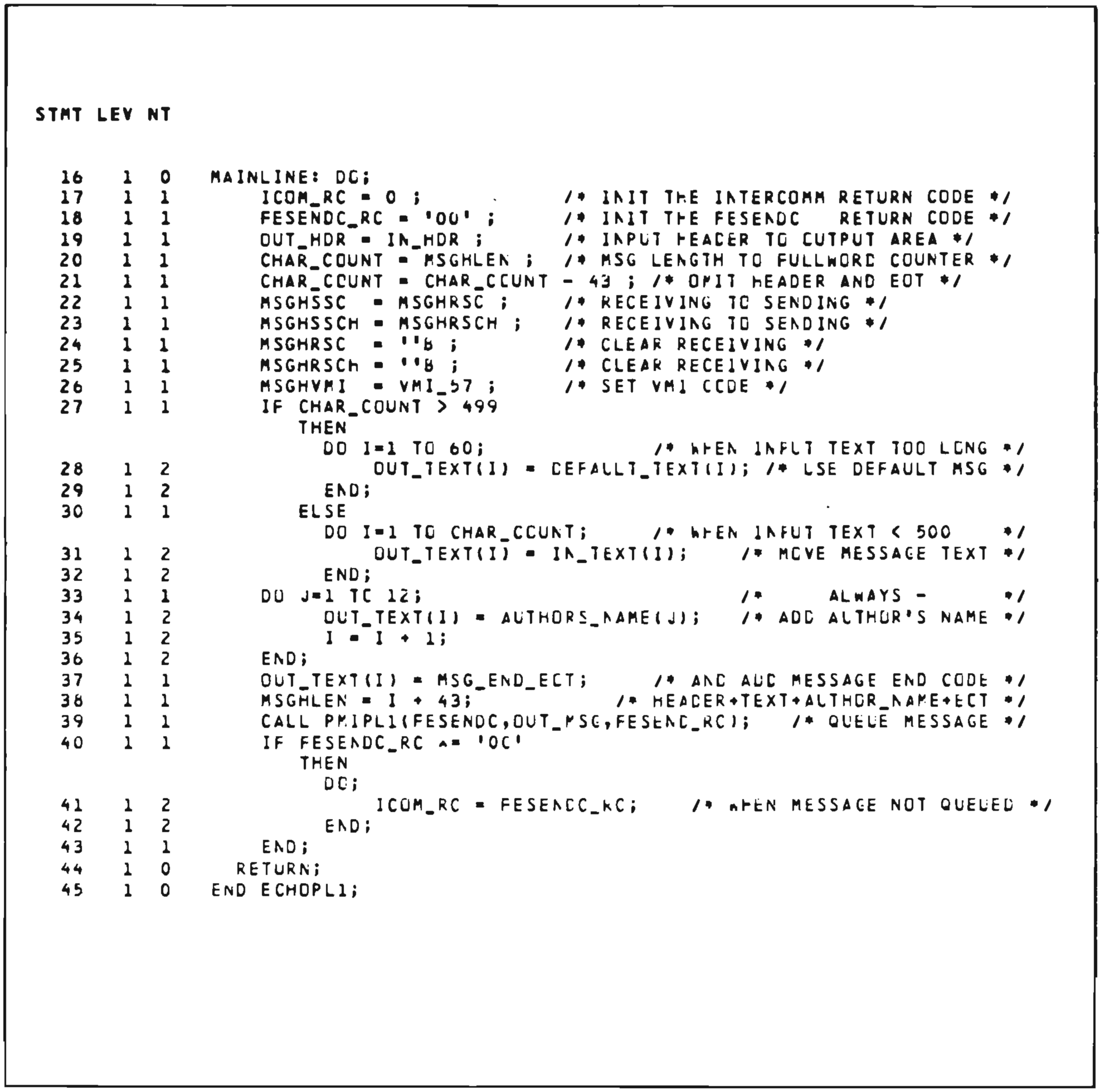
```
STMT LEV NT


              /*         DECLARE CONSTANTS AND AUTOMATIC VARIABLES.          */
   7   1  0  DCL  (I,J)              FIXED BIN(15);  /* CCUNTERS INTC TEXT  */
   8   1  0  DCL  CHAR_COUNT         FIXED BIN(31);  /* ACTUAL TEXT LENGTH  */
   9   1  0  DCL  FESENDC_RC         PIC'99',        /* FESENDC RETLRN CODE */
                  FESEND_RC          CHAR(2)  CEFINEC FESENDC_RC;
                               /* MUST BE CHAR FCR CALL PMIPL1(FESENCC,...) */
  10   1  0  CCL  AUTHORS_NAME2      CHAR(12)   INIT(' - M. DAVIES')  STATIC,
                  AUTHORS_NAME(12) CHAR(1)      DEFINEC   ALTHORS_NAME2;
  11   1  0  CCL  DEFAULT_TEXT2      CHAR(6C)   INIT('CRIGINAL DATA TOC LCNG - THIS TE
             XT HAS BEEN PLT IN ITS PLACE')     STATIC,
                  DEFAULT_TEXT(60) CHAR(1)      CEFINEC   CEFAULT_TEXT2;
  12   1  0  DCL  VMI_57         BIT(8) ALIGNEC  INIT('C1C10111')   STATIC;
  13   1  0  DCL  MSG_END        BIT(8) ALIGNEC  INIT('CC11C111')   STATIC,
                  MSG_END_EOT  CHAR(1)          DEFINEC   MSG_END;
             /*  'DEFINEC' CF MSG_END CODE AS CHAR SAVES EXTERNAL            */
             /*            SUBRCUTINE CALL TC MCVE BIT STRING.               */
             /*  NOTE: ABCVE USE CF DEFINED CAUSES CCMPILER RETURN CODE=8 */
             /*        BUT DOES NOT CAUSE LINKEDIT/EXECLTION PROBLEMS.       */
```

Figure 17. Echo Message Example; Reentrant PL/1
(Page 2 of 4)



```
STMT LEV NT


  14   1  0   DCL  PMIPL1        ENTRY EXTERNAL;   /* INTERCOMM INTERFACE */
             %INCLUDE PENTRY;*************************************************************
  15   1  0          DCL 1 PENTRY STATIC,                                      /* UPDATE */
                       2 ( /*IF OFFSET ODD,TRUE OFFSET=-(OFFSET+1)*/
                           INTSORTC                INIT(99),                   /* REL 10 */
                           DWSSNAP                 INIT(95),                   /* REL 10 */
                           MAPFREE                 INIT(91),
                           FECMRLSE                INIT(87),
                           FESEND                  INIT(83),
                           FESENDC                 INIT(79),
                           ALLOCATE                INIT(75),
                           ACCESS                  INIT(71),
                           MAPURGE                 INIT(67),
                           MAPCLR                  INIT(63),
                           MAPEND                  INIT(59),
                           MAPOUT                  INIT(55),
                           MAPIN                   INIT(51),
                           INTUNSTO                INIT(47),
                           INTSTORE                INIT(43),
                           INTFETCH                INIT(39),
                           FECMFDBK                INIT(35),
                           FECMDDQ                 INIT(31),
                           QWRITEX                 INIT(27),
                           QREADX                  INIT(23),
                           QWRITE                  INIT(19),
                           QREAD                   INIT(15),
                           QCLOSE                  INIT(11),
                           QOPEN                   INIT( 7),
                           QBUILD                  INIT( 3),
                           SELECT                  INIT( 4),
                           RELEASE                 INIT( 8),
                           READ                    INIT(12),
                           WRITE                   INIT(16),
                           GET                     INIT(20),
                           PUT                     INIT(24),
                           RELEX                   INIT(28),
                           FEOV                    INIT(32),
                           COBPUT                  INIT(68),
                           MSGCOL                  INIT(72),
                           COBSTORF                INIT(76),
                           CONVERSE                INIT(80),
                           DBINT                   INIT(84),
                           LOGPUT                  INIT(88),
                           PAGE                    INIT(92),
                           GETV                    INIT(96),
                           PUTV                    INIT(100) )
                     FIXED BIN(15);
             ****************           /* FOR REENTSBS CODES/ENTRY POINTS */
```

Figure 17. Echo Message Example; Reentrant PL/1 (Page 3 of 4)



```
STMT LEV NT


  16   1  0   MAINLINE: DO;
  17   1  1       ICOM_RC = 0 ;              /* INIT THE INTERCOMM RETURN CODE */
  18   1  1       FESENDC_RC = '00' ;        /* INIT THE FESENDC   RETURN CODE */
  19   1  1       OUT_HDR = IN_HDR ;         /* INPUT HEADER TO OUTPUT AREA */
  20   1  1       CHAR_COUNT = MSGHLEN ;     /* MSG LENGTH TO FULLWORD COUNTER */
  21   1  1       CHAR_COUNT = CHAR_COUNT - 43 ; /* OMIT HEADER AND EOT */
  22   1  1       MSGHSSC  = MSGHRSC ;       /* RECEIVING TO SENDING */
  23   1  1       MSGHSSCH = MSGHRSCH ;      /* RECEIVING TO SENDING */
  24   1  1       MSGHRSC  = ''B ;           /* CLEAR RECEIVING */
  25   1  1       MSGHRSCH = ''B ;           /* CLEAR RECEIVING */
  26   1  1       MSGHVMI  = VMI_57 ;        /* SET VMI CODE */
  27   1  1       IF CHAR_COUNT > 499
                     THEN
                       DO I=1 TO 60;                /* WHEN INPUT TEXT TOO LONG */
  28   1  2                OUT_TEXT(I) = DEFAULT_TEXT(I); /* USE DEFAULT MSG */
  29   1  2            END;
  30   1  1          ELSE
                       DO I=1 TO CHAR_COUNT;        /* WHEN INPUT TEXT < 500    */
  31   1  2                OUT_TEXT(I) = IN_TEXT(I);    /* MOVE MESSAGE TEXT */
  32   1  2            END;
  33   1  1       DO J=1 TO 12;                            /*     ALWAYS -      */
  34   1  2            OUT_TEXT(I) = AUTHORS_NAME(J);   /* ADD AUTHOR'S NAME */
  35   1  2            I = I + 1;
  36   1  2       END;
  37   1  1       OUT_TEXT(I) = MSG_END_EOT;      /* AND ADD MESSAGE END CODE */
  38   1  1       MSGHLEN = I + 43;           /* HEADER+TEXT+AUTHOR_NAME+EOT */
  39   1  1       CALL PMIPL1(FESENDC,OUT_MSG,FESENDC_RC);   /* QUEUE MESSAGE */
  40   1  1       IF FESENDC_RC ^= '00'
                     THEN
                       DO;
  41   1  2                ICOM_RC = FESENDC_RC;     /* WHEN MESSAGE NOT QUEUED */
  42   1  2            END;
  43   1  1       END;
  44   1  0     RETURN;
  45   1  0   END ECHOPL1;
```

Figure 17.   Echo Message Example; Reentrant PL/1
(Page 4 of 4)

3.4 PL/1 CODING CONVENTIONS AND TECHNIQUES

When coding a PL/1 subsystem, there are several PL/1 features to consider:

1. ON-units

   These may be used under Intercomm; however, note the following:

   a. Do not reference conditions that will be handled by Intercomm program check (SPIE/ESPIE) processing (for example, CONVERSION, FIXEDOVERFLOW, ZERODIVIDE); nor that concerning PL/1 I/O (for example, ENDFILE, KEY, TRANSMIT), all of which are handled by the Intercomm File Handler interface.

   b. For a production subsystem, ON-units may incur an inordinate amount of overhead - restrict their use to debugging, if possible.

2. BEGIN-blocks

   Beware of the overhead involved in block initialization procedures, both for storage and for processing time.

3. RECURSIVE procedures

   Use cautiously, considering storage allocations involved; do not call an internal PL/1 procedure from within a called procedure, or from within itself.

4. ALLOCATE/FREE statements

   Controlled and dynamically allocated based variables should not be used unless they can be allocated by PL/1 from the ISA supplied by Intercomm. If such variables are used, be careful to specify an ISA size large enough to include the allocated storage on the SPAC parameter of the SYCTTBL macro for the subsystem.

5. FETCH/RELEASE statements

   Do not use for dynamic loading of external procedures. Instead, CALL them as user subroutines using Intercomm controlled interfaces.

6. Multitasking

   Don't. If necessary, call an Assembler Language routine that issues a SUBTASK macro (see Intercomm Assembler Language Programmer's Guide).

7. Data conversion requiring subroutine calls

Avoid whenever possible (message IEL0906I at end of compile): check correct syntax on field editing PICTURE patterns, match variable definition attributes for simple data moves (when arithmetic conversion is not required). Define numeric input fields edited by the Edit Utility with PIC '...' clause. Define numeric fields in MMU maps to have the same form as in the associated file record and let MMU do the editing.

8. CONVERSION and ZERODIVIDE conditions

Prevent them by testing fields are numeric before arithmetic conversion, and not zero before division.

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

PL/1 subsystems and subroutines using Intercomm reentrant coding conventions are eligible for loading above the 16M line under XA and ESA if these recommendations are followed:

- The module should be linkedited with the AMODE=31,RMODE=ANY, NCAL and RENT (or REUS) parameters.

- For subsystems, the LOADNAM, LANG=RPL1, 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=PL1, BLDL=YES (default) and USAGE=REENT (default) parameters are required on the SUBMODS macro defining the subroutine to Intercomm in the REENTSBS table (see also Chapter 9).

- Ensure that the Intercomm interface routines SYCT400 (Subsystem Controller), PREPLI, PMIPL1, INTLOAD, and DYNLLOAD (for loaded subroutines) were reassembled with the XA global on in the Intercomm global table SETGLOBE.

- All parameters passed via direct calls to service routines or user subroutines must be in 24-Amode automatic storage (DSA). Constants (file names, map names, etc.) must be moved to automatic variables in the program's DSA before the call. The location of the parameter addresses is not checked by Intercomm for direct calls, however a program check will occur if a 31-Amode parameter is referenced by a 24-Amode (resident in Intercomm load module or dynamically loaded) program.

- All programs issuing direct calls to Intercomm service routines and user subroutines must be linked with the Intercomm interface routine INTLOAD (see Appendix A) which dynamically interfaces with the resident Amode-switching routine SWMODE which must be in the Intercomm load module.

- All parameters (except the PENTRY (REENTSBS) code) passed via calls to PMIPL1 must be in 24-Amode automatic storage (DSA). Constants (file names, map names, etc.) must be moved to automatic variables in the program's DSA before the call. PMIPL1 checks all parameter addresses (even those passed to user subroutines), and if not a 24-Amode address, PMIPL1 will force a program check (ISK 0,1) and therefore not execute the call. If only PMIPL1 is called, the loaded program need not be linked with INTLOAD as PMIPL1 performs mode-switching.

- For any calls to user subroutines, whether via PMIPL1 or called directly, the user subroutines must be defined via SUBMODS macros in the REENTSBS table which must be in the Intercomm load module. See Chapter 9 for defining the subroutines to INTLOAD when using direct calls (the caller must be linked with a modified INTLOAD which contains entries for the subroutines in addition to service routine entry points). See Appendix A for PL/1 subroutine linkediting.

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 DBMS 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 PL/1 subsystem to snap data areas from its own DSA; a PL/1 subroutine can snap areas from the calling subsystem's ISA (data areas passed as parameters to the subroutine). 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). The parameters for the DWSSNAP call are:

Parameter	Contents
SNCWname	The Snap Control Word, initialized to: 'ƀSƀƀ' (SNAP Option) for output to the system SNAPDD data set; 'ƀDƀƀ' (DISPLAY Option) for output back to the inputting terminal, 'ƀPƀƀ' (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 DSA which represents the start of the area to be snapped.
parm-address-end	A data name in the subsystem's/subroutine's DSA which represents the end (must be a higher address than start) of the area to be snapped.

Coding format:

```
          CALL  DWSSNAP(SNCWname[,term-id]
                [,parm-address-start[,parm-address-end]]);
```

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

```
          CALL  DWSSNAP(SNCWname);
```

will cause the entire ISA to be snapped.

```
          CALL  DWSSNAP(SNCWname,parm-address-start);
```

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

NOTE: the PL/1 compiler does not place data fields in the DSA in the order coded. Use the map of the DSA to determine delimiters when using address pairs, or insert a dummy field to provide an address-end label as in the subroutine example below.

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 PL/1 subroutine, the addresses passed to DWSSNAP as parms must be within the ISA of the main PL/1 subsystem. To pass addresses in the ISA of the subsystem from a subroutine, they must be part of the input parameters to the subroutine. For example:

```
SUBRTN: PROC (RECORD_PTR) OPTIONS(REENTRANT);
DCL RECORD_PTR POINTER;
DCL 1 RECORD-AREA BASED(RECORD_PTR),
      4 RECORD                       CHAR(166),
      4 RECORD_END_FILLER            CHAR(1);
DCL 1 SNCW CHAR(4);
      .

      .
      SNCW = 'b̸Db̸b̸';
      CALL DWSSNAP(SNCW,RECORD,RECORD_END_FILLER);
```

will cause a snap, to the inputting terminal, of the 166-character Record-Area passed to the subroutine by the subsystem as a parameter, provided the output does not exceed one screen (everything in excess of one screen will be lost). RECORD_END_FILLER is a delimiter for the snap.

3.7 BASED VERSUS NONBASED PARAMETERS

The PL1LNK parameter of the SYCTTBL macro defining the subsystem to Intercomm specifies whether PREPLI will pass the first three parameters to the subsystem as pointer variables or character strings (PL1LNK=NONBASED) for which the addresses of Locator/Descriptors for simple character strings are in the parameter list, or whether the parameters are passed as arithmetic variables which are the actual addresses of the areas as for Assembler Language programs (PL1LNK=BASED).

As seen in Figure 12, the default method of receiving the parameters to a PL/1 subsystem is as pointer variables, or as illustrated in Figure 17 as simple character strings. If using pointers, these areas may be defined as structures to enable individual data items within a parameter area to be referenced. If the first three parameters are pointers, the character strings or structures are defined as BASED upon the passed parameters as illustrated in Figure 12. Figure 17a illustrates a version of Figure 12, and shows the passed parameters defined as arithmetic variables (FIXED BIN(31)), that is, they are the actual addresses of the parameter areas. Again, the parameter areas are BASED upon the incoming parameters, however, it is necessary to set up each arithmetic variable as the character string address as illustrated in the MAINLINE code for the input message.

Declaring the input parameters as POINTERs is the easiest to use as a structure may be defined on each BASED pointer and the declaration is valid for message header reference and input parameter reference for direct calls and calls via PMIPL1. When declared simply as character strings, they may not be defined as structures (invalid data item references occur). When declared as arithmetic variables, addressing must be declared, and if MAPIN is called directly, six parameters must be passed (as though PMIPL1 was being called with the code for MAPIN as a seventh parameter) and the mapped input area may not be a BASED area.

```
 EXAMPLE2:  PROC (IN_MSG_ADDR,SPA_ADDR,SCT_ADDR,ICOM_RC)
                 OPTIONS(MAIN,REENTRANT);
   /*         DEFINE THE PASSED PARAMETERS:                         */
 DCL   IN_MSG_ADDR   FIXED BIN(31);                /* INPUT PARM 1 */
 DCL   SPA_ADDR      FIXED BIN(31);                /* INPUT PARM 2 */
 DCL   SCT_ADDR      FIXED BIN(31);                /* INPUT PARM 3 */
 DCL   ICOM_RC       FIXED BIN(31);                /* INPUT PARM 4 */
 DCL   IN_MSG     CHAR(4096) BASED(IN_MSG_PTR);    /* INPUT MSG    */
   /*                                                               */
   /*         DEFINE STATIC STORAGE AREAS:                          */
   /*    THESE AREAS SHOULD HAVE THE INITIAL ATTRIBUTE              */
   /*    AND NOT BE MODIFIED.                                       */
   /*                                                               */
 DCL  VMI_57      BIT(8) ALIGNED  INIT('01010111')    STATIC;
 DCL  RSC_OUTPUT  BIT(8) ALIGNED  INIT('11100100')    STATIC;
 DCL  RSCH_OUTPUT BIT(8) ALIGNED  INIT('11100100')    STATIC;
 DCL  FILE_NAME   CHAR(8)         INIT('MYFILE  ')    STATIC;
                                .
                                .
                                .
   /*                                                               */
   /*         DEFINE VARIABLE STORAGE AREAS:                        */
   /*    THESE AREAS WILL BE DEFINED IN AUTOMATIC STORAGE           */
   /*    AND WILL BE ASSIGNED FROM THE PROVIDED ISA.                */
   /*    THERE WILL BE ONE SET OF AREAS FOR EACH MESSAGE            */
   /*    THREAD INVOKED.                                            */
   /*                                                               */
 DCL  OUT_MSG              CHAR(2048);               /* OUTPUT MSG */
 DCL  I,J                  FIXED BIN(15);            /* COUNTERS   */
 DCL  FILE_RECOND_AREA     CHAR(200);                /* READ AREA  */
 DCL  ICOM_RETURN_VALUE    FIXED BIN(31);            /* RETURN CODE*/
                                .
                                .
                                .
   /*                                                               */
   /*         NOW DEFINE PROCESSING PROGRAM LOGIC.                  */
   /*                                                               */
1 MAINLINE: DO;
     ICOM_RC = 0;               /* INIT THE INTERCOMM RETURN CODE */
     IN_MSG_PTR = ADDR(IN_MSG_ADDR);       /* SET PARM AS ADDRESS */
     .
     .                                   Program Processing Logic
     .
     ICOM_RC = ICOM_RETURN_VALUE;         /* SET ICOM RETURN CODE */
     RETURN;
 END EXAMPLE2;
```

Figure 17a. Reentrant PL/1 Subsystem Structure using BASED arithmetic variable parameters.

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 %INCLUDE statements in the application program's dynamic storage area (automatic storage). 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 within an MMU symbolic structure definition as follows:

```
    4 CUSTMERL   FIXED BIN(15),      (length)
    4 CUSTMERT   CHAR(1),            (flag/attribute)
    4 CUSTMER    CHAR(25),           (data)
```

When defining maps for use by PL/1 subsystems, there is a special parameter, BASED, to be coded to indicate for symbolic map area generation whether the map area is (YES - default) or is not (NO) to be based on a pointer (PTR_mapname). If YES is coded, the symbolic map area for input message mapping may be acquired by MMU (requires a direct call to MAPIN with 5 parameters) and it replaces the input message area which was also based on a pointer (requires PL1LNK=NONBASED on subsystem SYCTTBL). The map area pointer is initialized after the MAPIN call and the acquired area is freed before RETURN to the Monitor as in the sample program in Chapter 10. When NO is coded, symbolic map areas are in the DSA.

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 <u>Message Mapping Utilities</u>.

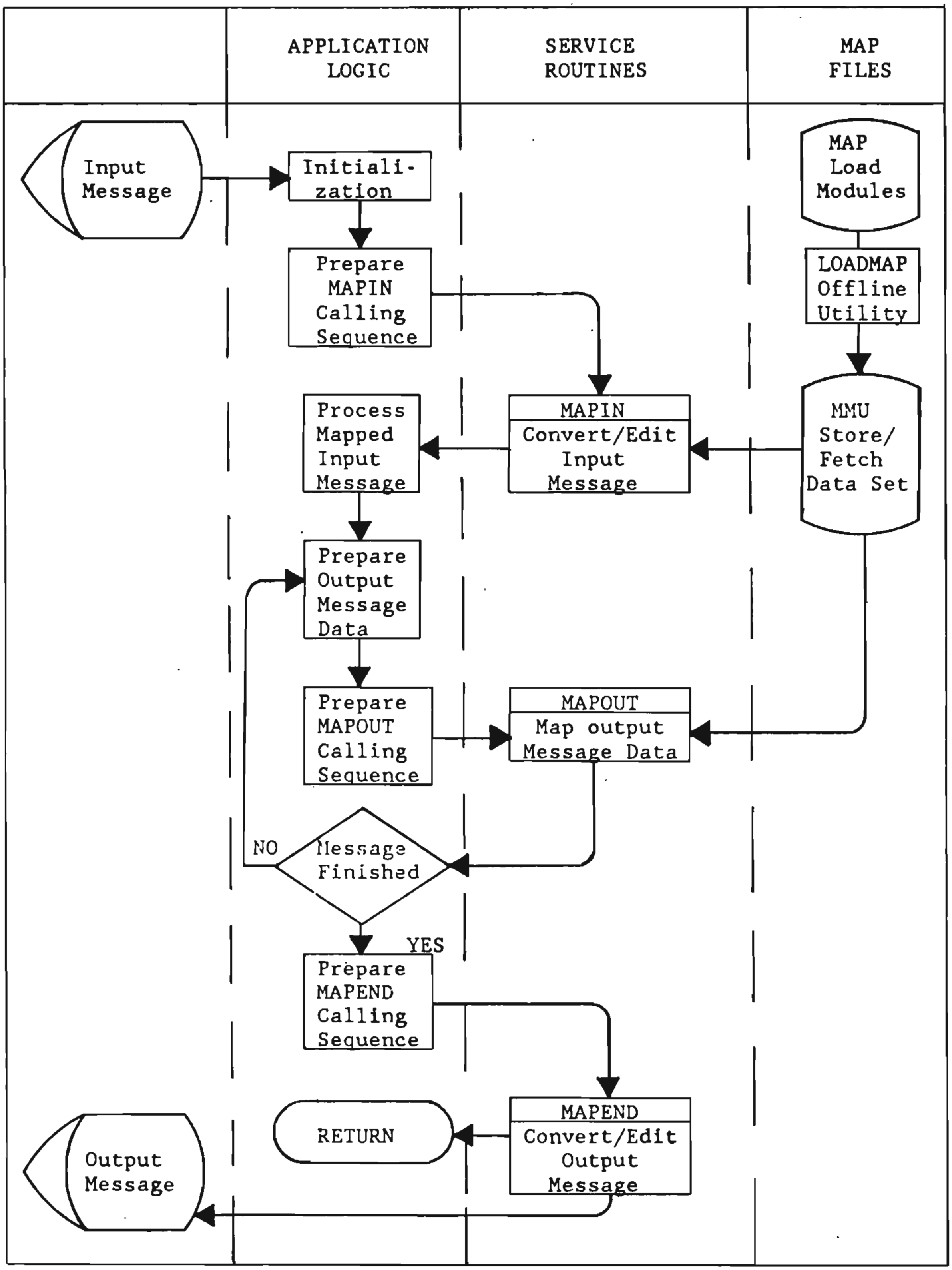

APPLICATION LOGIC
SERVICE ROUTINES
MAP FILES
Input Message
Initialization
MAP Load Modules
Prepare MAPIN Calling Sequence
LOADMAP Offline Utility
Process Mapped Input Message
MAPIN
Convert/Edit Input Message
MMU Store/ Fetch Data Set
Prepare Output Message Data
Prepare MAPOUT Calling Sequence
MAPOUT
Map output Message Data
NO
Message Finished
YES
Prepare MAPEND Calling Sequence
RETURN
MAPEND
Convert/Edit Output Message
Output Message

Figure 18. Message Processing Using MMU

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 PREPLI 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:

- <u>Fixed Format</u>

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

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

- <u>Variable Format</u>

  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 PL/1 file access functions and the Intercomm File Handler service routines is shown in Figure 20.

Function	PL/1 Requests	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 WRITE,REWRITE LOCATE	GETV PUTV 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 statements in a PL/1-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, Model 204, System 2K) 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 as automatic variables of reentrant programs, so that space can be assigned out of the ISA. To assure that they are fullword aligned, they should be defined following a "FIXED BIN (31)" field. To force the proper alignment, Figure 22 shows how these control areas may be declared for direct calls to File Handler routines.

```
DCL  1  FH_AREAS  ALIGNED,
        3 FH_DUMMY     FIXED BIN(31),
        3 EXTDSCT      CHAR(48),
        3 FHCW,
           5 FHCW1     CHAR(1),
           5 FHCW2     CHAR(1),
           5 FHCW3     CHAR(1),
           5 FHCW4     CHAR(1);
```

Figure 22. Defining File Handler Control Areas

If calling File Handler routines via PMIPL1, the FHCW must be declared as follows (see also sample program in Appendix D):

```
        3 FHCW UNALIGNED CHAR(4),
     1  FHCW_REDEF DEFINED FH_AREAS.FHCW,
           5 FHCW1     CHAR(1),
           5 FHCW2     CHAR(1),
           5 FHCW3     CHAR(1),
           5 FHCW4     CHAR(1);
```

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 ƀ. 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 PL/1 subsystem may call the File Handler service routines through the Intercomm interface module PMIPL1, and provide a routine-code name corresponding to the desired routine name, as described in the Intercomm %INCLUDE member PENTRY, or the routines may be called directly (see Chapter 3). The PMIPL1 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 (under XA or ESA), then all parameters (except the PENTRY code, if used) must be in Automatic storage (DSA), otherwise, the ddname may be in Static storage.

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 from Static to Automatic storage for calls from 31-Amode 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: ● 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 binary zeroes (X'00').

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 binary zeroes before being reused.

Coding format:

```
CALL SELECT(EXTDSCTname,FHCWname,ddname);

CALL RELEASE(EXTDSCTname,FHCWname);
```

Note: the ddname must be in Automatic storage (DSA) if the subsystem (subroutine) can be loaded above the 16M line under XA or ESA.

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.	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 subsystem:

- 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 ('ƀCƀƀ') and call RELEASE supplying the ddname of the file to be closed; use the following coding format:

  ```
  CALL RELEASE(ddname,FHCWname);
  ```

Note: the ddname must be in Automatic storage (DSA) if the subsystem (subroutine) can be loaded above the 16M line under XA or ESA.

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 'ƀXƀƀ' to the File Handler Control Word prior to calling GET or READ. Exclusive control does not apply to physical sequential (QSAM/BSAM) files.

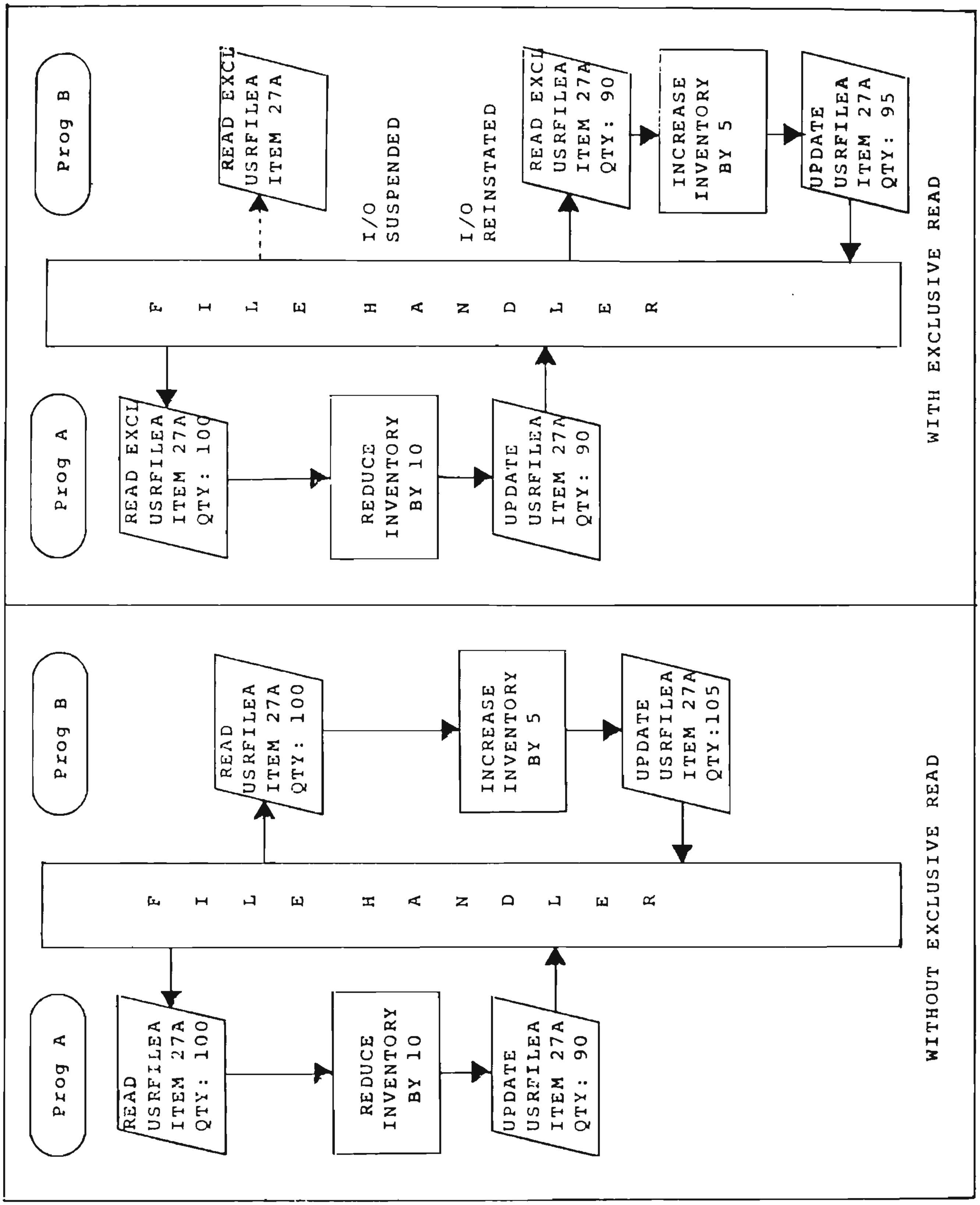


Figure 26. Exclusive Control Processing

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.

- 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 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) PROCESSING

6.5.1 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 GET(EXTDSCTname,FHCWname,record-area[,record-length]);

CALL READ(EXTDSCTname,FHCWname,record-area[,record-length]);

CALL PUT(EXTDSCTname,FHCWname,record-area[,record-length]);

CALL 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. 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   GET(EXTDSCTname,FHCWname,record-area);
```

to reposition and retrieve record with key equal or high:

```
CALL   GET(EXTDSCTname,FHCWname,record-area,key);
```

to update last GET:

```
CALL   PUT(EXTDSCTname,FHCWname,record-area);
```

to retrieve a specific record:

```
CALL   READ(EXTDSCTname,FHCWname,record-area,key);
```

to update last READ:

```
CALL   WRITE(EXTDSCTname,FHCWname,record-area);
```

to add a specific record:

```
CALL  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 READ(EXTDSCTname,FHCWname,record-area[,key],block-id);

CALL 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
b̸	READ w/o exclusive control, w/block-id	READ DIF
X	READ w/exclusive control, w/block-id	READ DIX
b̸	WRITE to update last READ, w/o block-id	WRITE DI/DIX
b̸	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
*b̸	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
*b̸	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	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 suppling 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 emply 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 GETV(EXTDSCTname,FHCWname,record-area);
```

Coding formats (continued):

For direct access

```
CALL GETV(EXTDSCTname,FHCWname,record-area,{rba});
                                           {key}
                                           {rrn}
```

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

```
CALL PUTV(EXTDSCTname,FHCWname,record-area);
```

For direct addition of a new record

```
CALL 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: Macro Instruction Reference or Macro Instructions for VSAM Data Sets.

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 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

	FHCW		
	Byte 1 (char)		Byte 4
Condition at Completion of Operation*	VSAM	ISAM	(hexadecimal)
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

*Characters in parentheses reference the type(s) of VSAM Call (Figure 32) which apply. A = all cases.

**Should not occur. The File Handler will force a program check condition to terminate the message in progress.

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 subsubsystem 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 occurence 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 seqment 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.

	Message Header Fields			Change/ Display Prefix
OUTPUT Message Type	MSGHRSCH	MSGHRSC	MSGHVMI	
Preformatted (device-dependent)	X'00'	C'U'	X'57' or C'P'	N/A
Variable Text Formatting:			X'50' or C'0'	
Single Segment Messages: 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
Multi-Segment Messages: 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'C'	
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'	
Fixed Field Formatting:	X'00'	C'H'	X'72' or C'S'	
Single-Segment Messages:				C'0'
Multi-Segment Messages: 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.

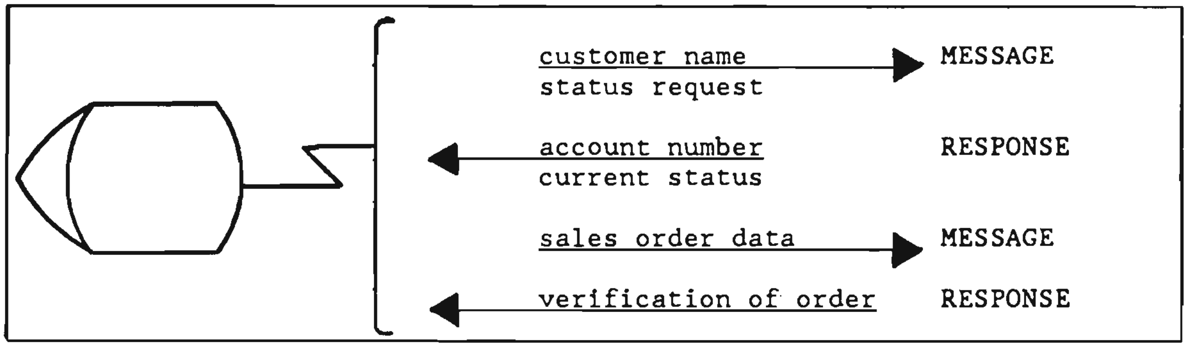


Figure 35. Typical Conversational Transactions

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

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 four methods of retention of input messages and corresponding results of processing.

Intercomm provides Front End conversational support to ensure that duplicate input 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 (Csect) is a 500-byte core-resident table. The user extention 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.

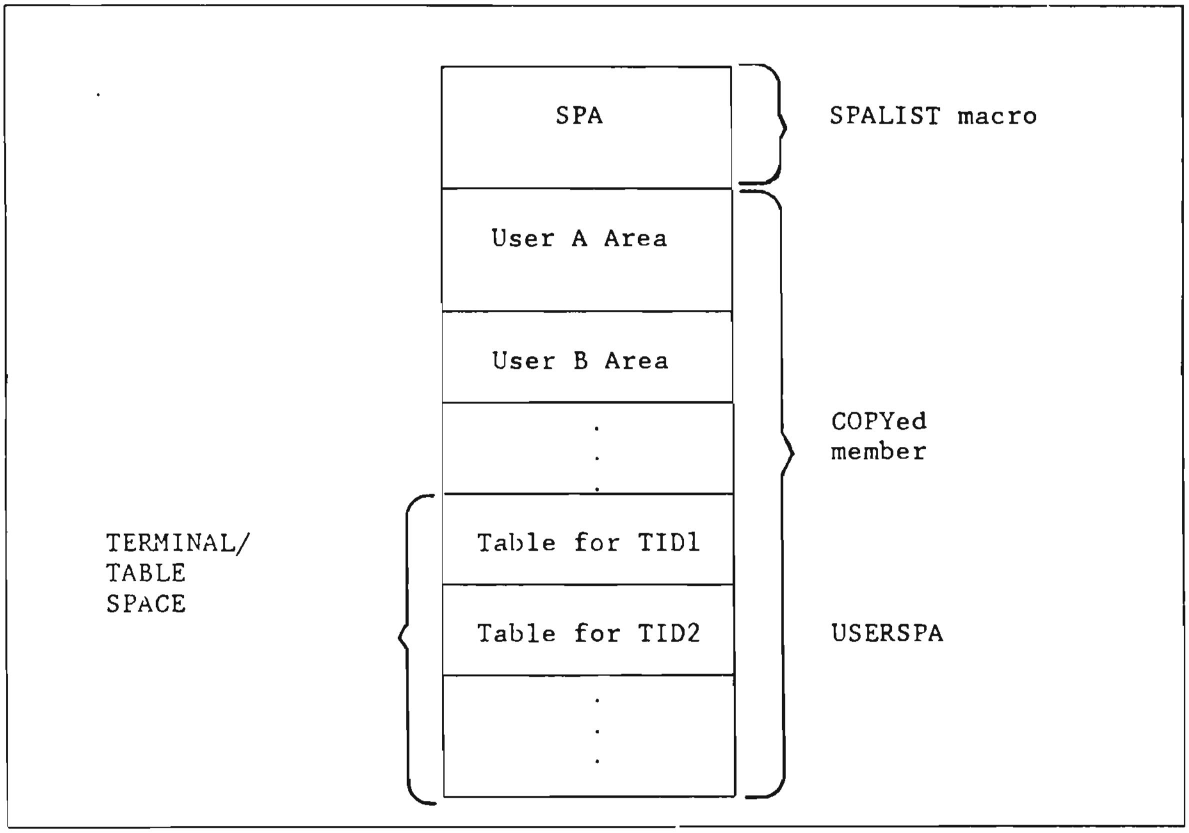


Figure 37. User and Terminal Table Space in the USERSPA

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 based structure 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 member to be included by all PL/1 routines using a '%INCLUDE OURSPA;' statement following the INTSPA statement.

```
DCL   1     FULLSPA     BASED(SPA),
        5   INTSPA      CHAR(500),
        5   OURSPA,
          6  COMMON_TABLE     CHAR(200),
          6  USERA_AREA       CHAR(50),
          6  USERB_AREA,
             8  COUNT_FIELD1     FIXED BIN(31),
             8  ON_OFF_SWITCH    CHAR(1),
             8  REST-OF-AREA     CHAR(45),
          6  USERC_AREA       CHAR(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.

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

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.

Initial Input:

STORE--create a new data string

Intermediate Input:

FETCH--retrieve existing data string

STORE--update string: new information merged with existing data

Final Input:

FETCH--retrieve existing data string

Process input and merge final information with existing data

Update necessary files and create final output message

UNSTORE--free data string storage

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

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.

```
Initial Input:

    QBUILD  -- Create a new queue

    QWRITE  -- Save input message and related data

    QCLOSE  -- Save the DDQ
-----------------------------------------------------------------------
 Intermediate Input:

    QOPEN   -- Open the queue
                                     )
    QREADX  -- Read the record       |        or QWRITE to add
               with intent to update >        to the queue
                                     |
    QWRITEX -- Update the record     )

    QCLOSE  -- Save the DDQ
-----------------------------------------------------------------------
 Final Input:

    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 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 storage (ISA area) 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 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.

<u>NOTE</u>: CONVERSE is <u>not</u> supported for subsystems loaded above the 16M line under XA or ESA.

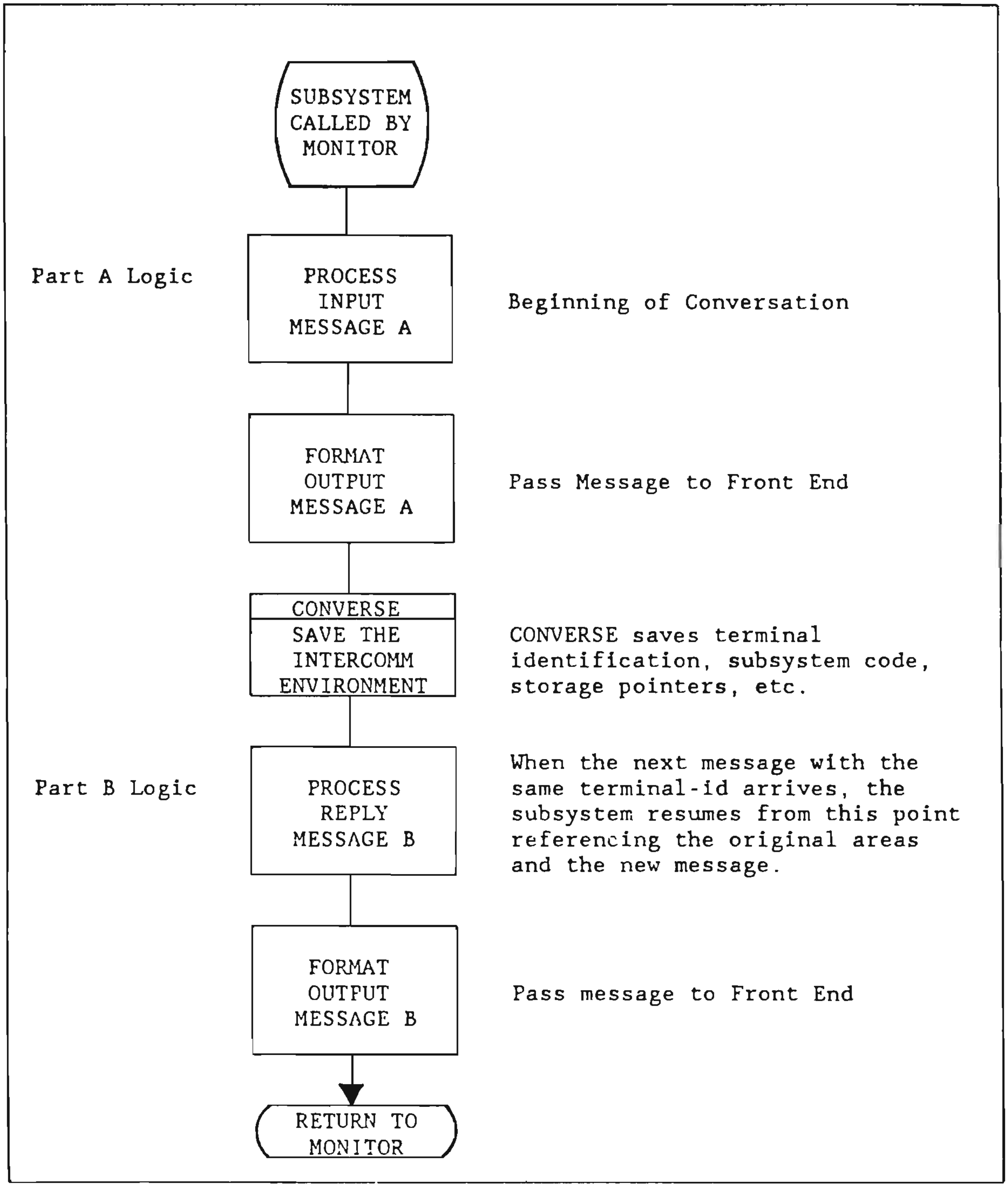

SUBSYSTEM CALLED BY MONITOR
Part A Logic
PROCESS INPUT MESSAGE A
Beginning of Conversation
FORMAT OUTPUT MESSAGE A
Pass Message to Front End
CONVERSE
SAVE THE INTERCOMM ENVIRONMENT
CONVERSE saves terminal identification, subsystem code, storage pointers, etc.
Part B Logic
PROCESS REPLY MESSAGE B
When the next message with the same terminal-id arrives, the subsystem resumes from this point referencing the original areas and the new message.
FORMAT OUTPUT MESSAGE B
Pass message to Front End
RETURN TO MONITOR

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 CONVERSE(word,time);
```

where:

word
: is the name of an aligned fullword (FIXED BIN(31)) in the subsystem's DSA required by CONVERSE for work space. The fullword must be CHAR(4) if CONVERSE is called via PMIPL1.

time
: is the name of an aligned fullword binary value (FIXED BIN(31)) 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, any subsystem may be in conversation with any number of terminals simultaneously.

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.

To use CONVERSE by a PL/1 subsystem, PL1LNK=NONBASED is required on the SYCTTBL macro definition (see Chapter 3), the input message parameter must be declared as a pointer, and the input message area must be BASED on that pointer.

Note that the CONVERSE routine may only be called from a 24-Amode subsystem. Due to complications arising in reestablishing internal pointers on return from the call to CONVERSE, it may not be called by a PL/1 or COBOL subroutine of the subsystem.

For example:

- Monitor calls PL/1 Subsystem AA which calls CONVERSE (valid sequence of program logic).
- Monitor calls PL/1 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). The calling PL/1 subsystem may then not reference the input message area or any of its data fields (except for data fields in its DSA 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 PL/1 subsystem, or the PL/1 subsystem may test it for the CONVERSE return code on return from the BAL subroutine.
- Monitor calls PL/1 Subsystem CC which calls PL/1 subroutine C1 which calls CONVERSE (invalid sequence of program logic).

The PL/1 subsystem may not use an old copy of the message header for a new output message.

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')	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.) CAUTION: The CONVERSE automatic time-out is not extended if a message is found in error by the Edit Utility.
17 (X'11')	No core available for CONVERSE control blocks; conversational mode not initiated.
18 (X'12')	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: *PMI*CONVERSE*ANTICIPATED MESSAGE NOT RECEIVED WITHIN USER SPECIFIED TIME INTERVAL

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 storage area (ISA) 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 SERVICE ROUTINE INTERFACING

PL/1 programs may call Intercomm service routines directly using the standard CALL statement. The service routines must be defined with ENTRY OPTIONS (ASM INTER) to generate an Assembler Language parameter list for the called routine. The member PLIENTRY is provided for copying into PL/1 programs (use %INCLUDE PLIENTRY), which defines all standard Intercomm routine entry points (entry point names are given in the REENTSBS illustration, Figure 43). Special facility entry points may be added by the user. For dynamically loaded programs, linking the program with INTLOAD (required if program may be loaded above the 16M line under XA or ESA) will reduce dynamic linkedit processing at Intercomm startup.

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

9.1.1 PL/1 Interface Routine (PMIPL1)

PMIPL1 (see also Chapter 3) is a PL/1 interface routine which may be called by PL/1 programs for service routine and user subroutine interface. When only PMIPL1 is called, dynamically loaded programs (even if loaded above the 16M line under XA or ESA) do not need to be linked with INTLOAD. The application program calls PMIPL1 specifying which routine is to be called (system service routine or user subroutine) and the appropriate parameters to pass to it. PMIPL1 preserves the PL/1 environment and performs mode switching if the caller is in 31-Amode. PMIPL1 also acquires a storage area in which it saves the entry parameters for the called program. PMIPL1 then calls the specified routine, and on return, PMIPL1 restores the caller's environment and returns to the calling program. Coding format:

```
CALL PMIPL1(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. All parameters passed to PMIPL1 must be in non-arithmetic format (locator/descriptors). PMIPL1 then passes the addresses in Assembler Language format except if the called subroutine is PL/1. All parameter addresses are validated for 24-Amode. If the calling subsystem (or subroutine) may be loaded above the 16M line under XA or ESA, then all parameters must be in the caller's DSA (have a 24-Amode address).

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 other service routines and user subroutines (in increments of 4). Figure 43 lists the routine-codes assigned as identifiers for Intercomm service routines in the released REENTSBS table. The member (of routine-codes) for copying into PL/1 subsystems and subroutines is named PENTRY (use %INCLUDE PENTRY) and is illustrated in Appendix B. See also Appendix D for sample coding using PMIPL1 and the PENTRY table. The routine-codes may be in Static storage for 31-Amode programs.

```
REENTSB1 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=INTSORTC     OFFSET -100,CODED AS 99
         SUBMODS NAME=DWSSNAP      OFFSET -96,CODED AS 95
         SUBMODS NAME=MAPFREE      OFFSET -92,CODED AS 91
         SUBMODS NAME=FECMRLSE     OFFSET -88,CODED AS 87
         SUBMODS NAME=FESEND       OFFSET -84,CODED AS 83
         SUBMODS NAME=FESENDC      OFFSET -80,CODED AS 79
         SUBMODS NAME=ALLOCATE     OFFSET -76,CODED AS 75
         SUBMODS NAME=ACCESS       OFFSET -72,CODED AS 71
         SUBMODS NAME=MAPURGE      OFFSET -68,CODED AS 67
         SUBMODS NAME=MAPCLR       OFFSET -64,CODED AS 63
         SUBMODS NAME=MAPEND       OFFSET -60,CODED AS 59
         SUBMODS NAME=MAPOUT       OFFEST -56,CODED AS 55
         SUBMODS NAME=MAPIN        OFFSET -52,CODED AS 51
         SUBMODS NAME=INTUNSTO     OFFSET -48,CODED AS 47
         SUBMODS NAME=INTSTORE     OFFSET -44,CODED AS 43
         SUBMODS NAME=INTFETCH     OFFSET -40,CODED AS 39
         SUBMODS NAME=FECMFDBK     OFFSET -36,CODED AS 35
         SUBMODS NAME=FECMDDQ      OFFSET -32,CODED AS 31
         SUBMODS NAME=QWRITEX      OFFSET -28,CODED AS 27
         SUBMODS NAME=QREADX       OFFSET -24,CODED AS 23
         SUBMODS NAME=QWRITE       OFFSET -20,CODED AS 19
         SUBMODS NAME=QREAD        OFFSET -16,CODED AS 15
         SUBMODS NAME=QCLOSE       OFFSET -12,CODED AS 11
         SUBMODS NAME=QOPEN        OFFSET -8,CODED AS 7
         SUBMODS NAME=QBUILD       OFFSET -4,CODED AS 3
         ENTRY REENTSBS
REENTSBS DS    0A                  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
                 .                          .
                 .                          .          (Codes 36-64
                 .                          .           are reserved)
                 .                          .
```

Figure 43. PMIPL1 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
REENTSB1  CSECT
          END
```

Figure 43. PMIPL1 Routine Pointers (REENTSBS) (Page 2 of 2)

9.2 INTERSUBSYSTEM QUEUING (COBPUT)

COBPUT (also used by COBOL programs) 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 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 Storage Area (DSA). 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 NOTE: 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 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)
NOTE:	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 Storage Area of the calling subsystem to a new dynamic area and calls Message Collection. Thus, the output message area within the Automatic storage 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 MSGCOL(message);
```

where:

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

MSGCOL return codes indicate the result of the queuing. The return code (stored in the Register 15 field of the caller's save area) may be accessed by the PLIRETV 'built-in-function'. (See Figure 46.) If MSGCOL is called via PMIPL1, a return-code field may be provided - see Apenddix D. 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
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.

9.4 FREE DYNAMIC (MESSAGE AREA) STORAGE (COBSTORF)

COBSTORF may be called to free some of the area utilized for the input message before it is passed to another subsystem, or to free the entire message when it is not to be freed by the Subsystem Controller when the subsystem returns. COBSTORF may also be used to free an area passed to a PL/1 subroutine which was dynamically acquired by a calling Assembler Language program.

Coding format:

```
CALL 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 (FIXED BIN(31)) 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). When freeing part of an input message, only the rightmost portion may be freed and the rounded remaining length must be stored in the first two bytes (MSGHLEN) of the message header. If freeing all of the input message area, the subsystem must return to the Monitor with the return code 900.

A further clarification is provided in the previous section on message queuing via MSGCOL.

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 DSA) 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 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 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'O'--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 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 PL/1 SUBSYSTEMS

All subroutines called by an application subsystem may be called directly or via PMIPL1. Under XA or ESA, passed parameter values must be in 24-Amode storage (such as the caller's DSA) if the routine is called via PMIPL1 or if it is resident or loaded in 24-Amode. 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 PL/1 subroutine.
- A data base interface routine.

If the routine is called directly, passed parameters must be appropriate to the language of the routine, for example, to pass a structured area to a PL/1 subroutine declare a pointer to the area and pass the pointer, whereas the label of the area may be passed to a BAL program.

The Intercomm return code area may be used as a parameter to pass a return code back to the calling PL/1 subsystem. 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. If a subroutine is called via PMIPL1, all parameters must have non-arithmetic attributes, therefore in this case the Intercomm return code area may not be used.

9.7.1 Defining User Subroutines to Intercomm

Except as noted in Section 9.7.4, a user-coded subroutine (Assembler Language or PL/1) 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 PENTRY (see Appendix B) for easy access by subsystems when calling PMIPL1, or add the name to PLIENTRY if it is a BAL routine. 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 PL/1 program, standard linkage conventions are used. PMIPL1 (if used) issues a MODCNTRL macro to link to non-resident Assembler subroutines. At entry, register 13 points to a save area in the caller's DSA.

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 save area may not otherwise be used by a called subroutine. An Assembler subroutine may not call a PL/1 subroutine (unless code is provided to pass the caller's PL/1 environment).

9.7.3 Interfacing to User-coded PL/1 Subroutines

A reentrant PL/1 subroutine is coded like a PL/1 subsystem in that it uses OPTIONS (REENTRANT) and a Dynamic Storage Area (in calling programs ISA - do not use the MAIN option), and it may call PMIPL1 to interface to Intercomm service routines and other user subroutines, or it may use direct calls. Non-resident reentrant PL/1 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. See Appendix A for subroutine linkedit considerations.

9.7.4 Interfacing When Caller or Subroutine is Non-Resident

When all calls are made via PMIPL1, all called routines (Intercomm and user) must be defined in the REENTSBS table via SUBMODS macros as described earlier in Sections 9.1.1 and 9.7.1. If the called routine is reentrant BAL and resident in the Intercomm load module (NAME parameter used on SUBMODS coding), PMIPL1 calls the routine directly passing the address of the caller's save area in register 13. If the routine is non-resident or not reentrant BAL (LNAME parameter used on SUBMODS macro), PMIPL1 links to the subroutine interface module DYNLLOAD which loads the called routine if necessary before giving it control, again passing the original caller's save area address and registers 2-12. DYNLLOAD performs mode switching if the called routine is loaded above the 16M line under Release 10. Return is via DYNLLOAD to PMIPL1 which then returns to the caller by using a previously saved return address.

When using direct calls between resident routines, the linkedit of the Intercomm load module resolves the external references, resident subroutines do not need to be defined in REENTSBS. To link to a loaded subroutine (must be defined in REENTSBS), a resident PL/1 program must either call it via PMIPL1 or call a resident BAL interface routine passing the name of the desired subroutine. The interface then issues a MODCNTRL macro to link to DYNLLOAD. If the desired subroutine is PL/1, the interface routine must pass the caller's registers 2 through 13. See the sample interface program in Appendix E.

For non-resident PL/1 programs using direct calls, linking the loadable program with INTLOAD will resolve all Intercomm routine entry points (REENTSBS not needed). Otherwise, dynamic linkedit must resolve the called entry point addresses at Intercomm startup, which adds startup processing overhead. Dynamic linkedit is required for IBMB.... internal PL/1 subroutines linked in the Intercomm load module. Dynamic linkedit can also be used to resolve calls from 24-Amode programs to user subroutines in the Intercomm load module. Calls to loadable subroutines (which must be defined in REENTSBS) can be made via PMIPL1 or an interface routine (resident or linked with loaded calling PL/1 program) as described above. If the calling program may be loaded above the 16M line, the interface program and INTLOAD must be linked with it (along with PLIV).

Under Release 10, the need for a BAL interface routine (or PMIPL1 calls) for loadable user subroutines (or for dynamic linkedit resolution for resident subroutines) can be eliminated for dynamically loaded PL/1 programs using direct calls. Define the user subroutines to REENTSBS by coding SUBMODS macros for them in the copy member USRSUBS. Always use the LNAME parameter even if defining a resident reentrant BAL subroutine. Then reassemble and link REENTSBS and reassemble and link INTLOAD so that they both copy the revised USRSUBS and thus entries are generated within INTLOAD for the directly called subroutines. Then link the loadable program with the revised INTLOAD. INTLOAD links directly to DYNLLOAD for 24-Amode callers or via the resident interface SWMODE for 31-Amode callers (required). In both cases, the PL/1 environment is preserved for called PL/1 subroutines. Note that as new subroutines are defined in USRSUBS and copied to INTLOAD for new calls, older programs which do not call the new programs do not have to be relinked with the latest revised INTLOAD. Dynamic linkedit may still be used for resident IBMB... routines as they can be entered directly in 31-Amode (see also Appendix A).

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 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 Automatic storage (DSA) if the calling program is loaded above the 16M line under XA or ESA.

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 FESENDC 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 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 (HDFF) 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 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)

To sort an in-core table, the INTSORT Facility (entry point INTSORTC for PL/1) is provided. Such a table might be data stored in a Store/Fetch string or file data record via online transactions or offline processing. The table can have any number of fixed-length entries up to 32767, and each entry can have a total size of 1 to 255 bytes. The key to be sorted on can be anywhere within the entry, but must be in the same place, and of the same length, in each entry. Coding format:

```
      CALL 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 (up to 32767) in binary format.

entry-length is a 4-byte (fullword aligned) area containing the size of each entry (up to 255) 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 zero if at the beginning of the table entry; 1 if it starts in the second position of the table entry, etc.).

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).

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
X'04'	Number of entries less than 1 or more than 32767.
X'08'	Length of an entry is less than 1 or greater than 255.
X'12'	No Table name (address) supplied.
X'16'	Key-offset greater than 254.
X'20'	The key-length plus key-offset exceeds maximum (255) entry-length.

For all non-zero return codes, 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)

Code names for the routines are provided in the members PLIENTRY and PENTRY (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 for Assembler Language programmers 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 called directly from PL/1 programs (declare entry name as ENTRY OPTIONS (ASM INTER), or add the entry name to USRSUBS with a SUBMODS macro (use NAME parameter only) and add the name and offset code to PENTRY if PMIPL1 called.

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, SSTEST)
- 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 a hexadecimal field to printable character (LAYOUT)
- 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.

NOTE: GETDATE may only be used under Release 10.

Chapter 10

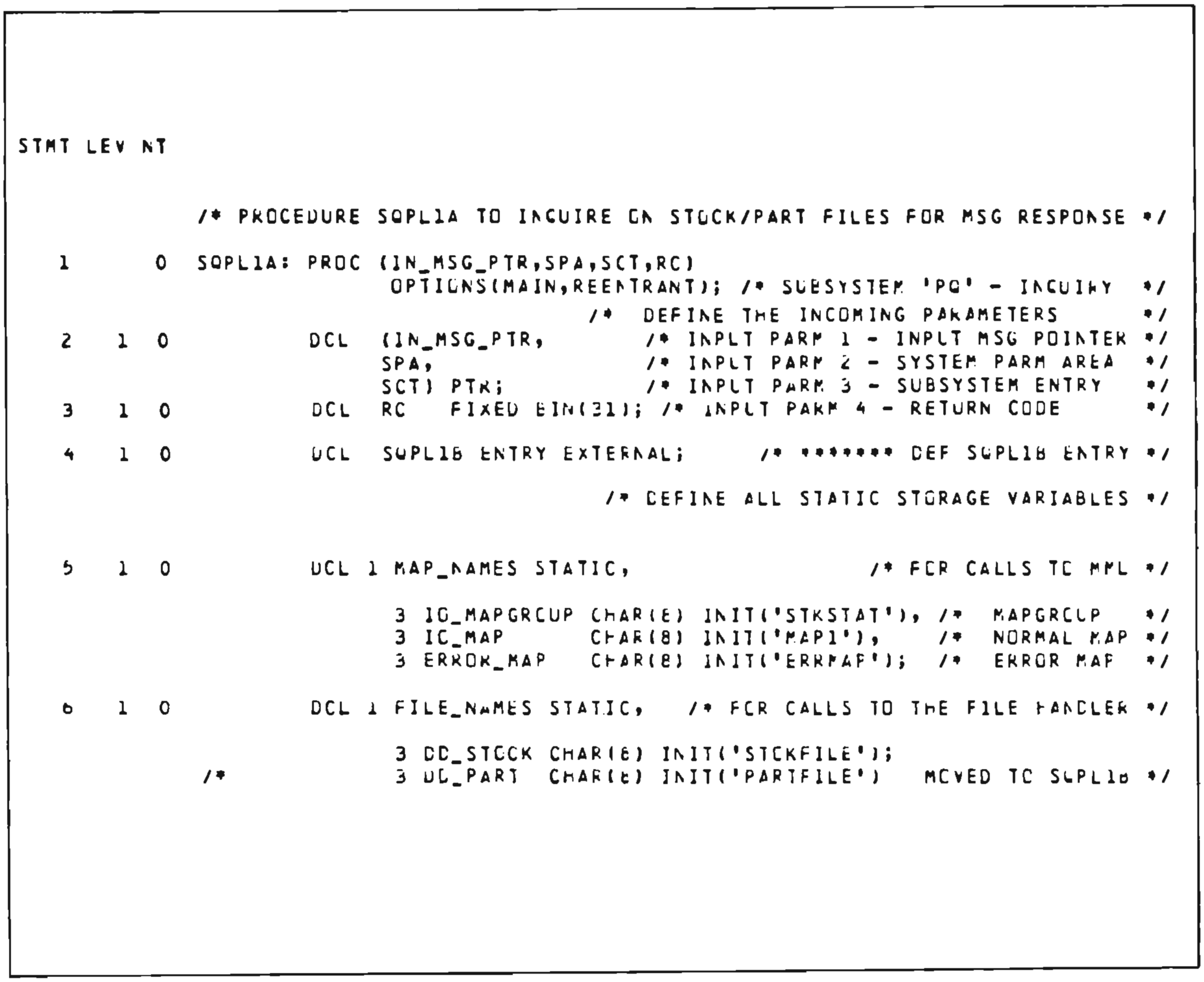
SAMPLE PROCESSING PROGRAMS

The sample program SQPL1A, shown in Figure 48, demonstrates coding of a PL/1 subsystem which is either resident or dynamically loadable above or below the 16M line (if XA or ESA). The program processes an inquiry transaction (TPL1) 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 PL/1 subroutine SQPL1B, illustrated in Figure 49, which also may reside above or below the 16M line. 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.

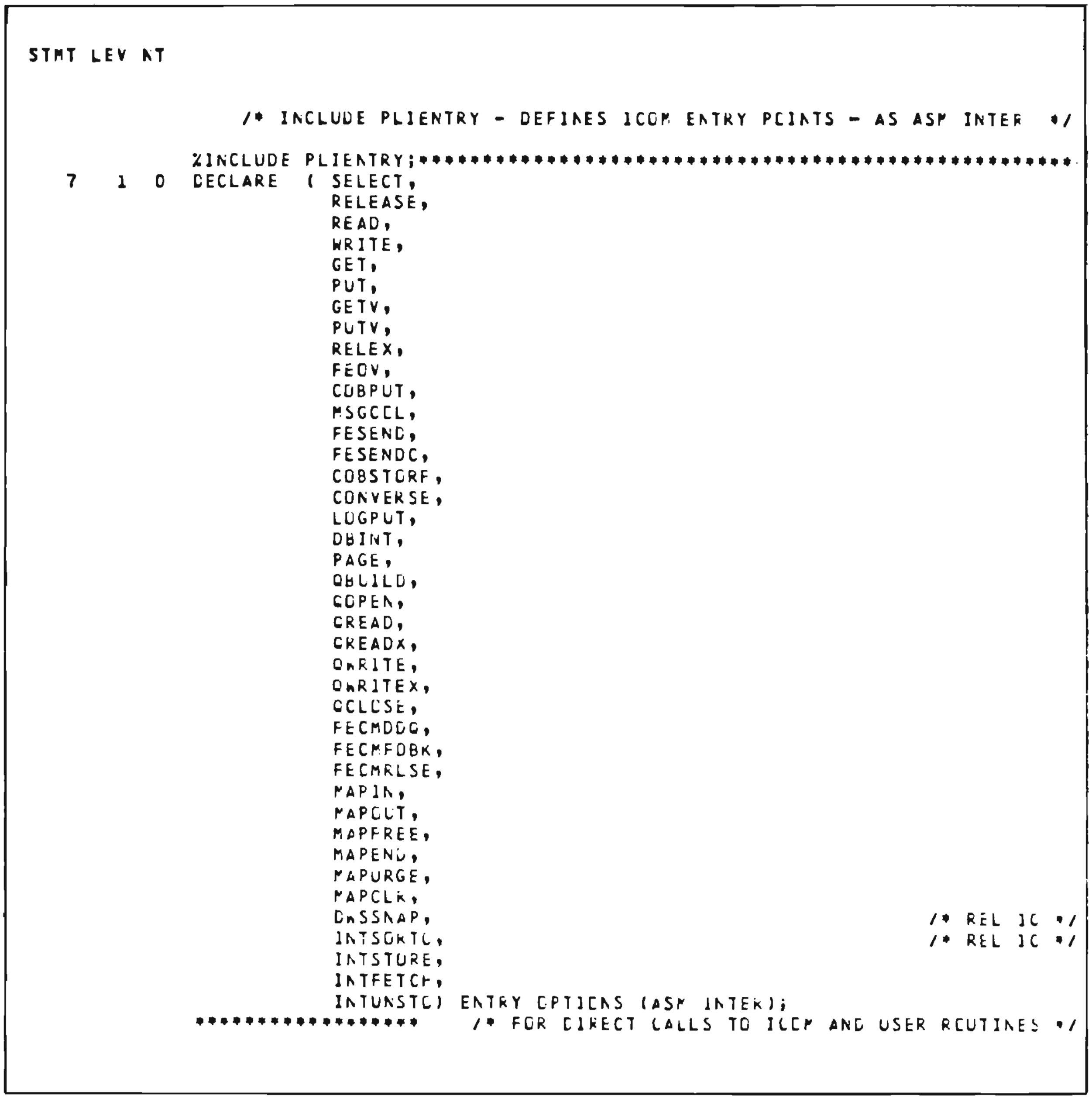
The PLIENTRY and PLMSGHD source text members defining the service routine entries and Intercomm message header fields are % INCLUDE'd from the source text members by the PL/1 compiler. The PLILOGCH source text member used for terminal attribute and command override for MMU processing, and the symbolic map areas, are also copied into the program. Note that the MMU symbolic map areas are BASED on PTR_mapname and that the pointers are set up in the program (MAPIN called directly with five parameters). Note also that the first three input parameters to the program are declared as pointers.

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 PQ. 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 PL/1 subroutine) coded for the same purpose but using the Edit and Output Utilities, a COBPUT call, and Test Mode for testing.



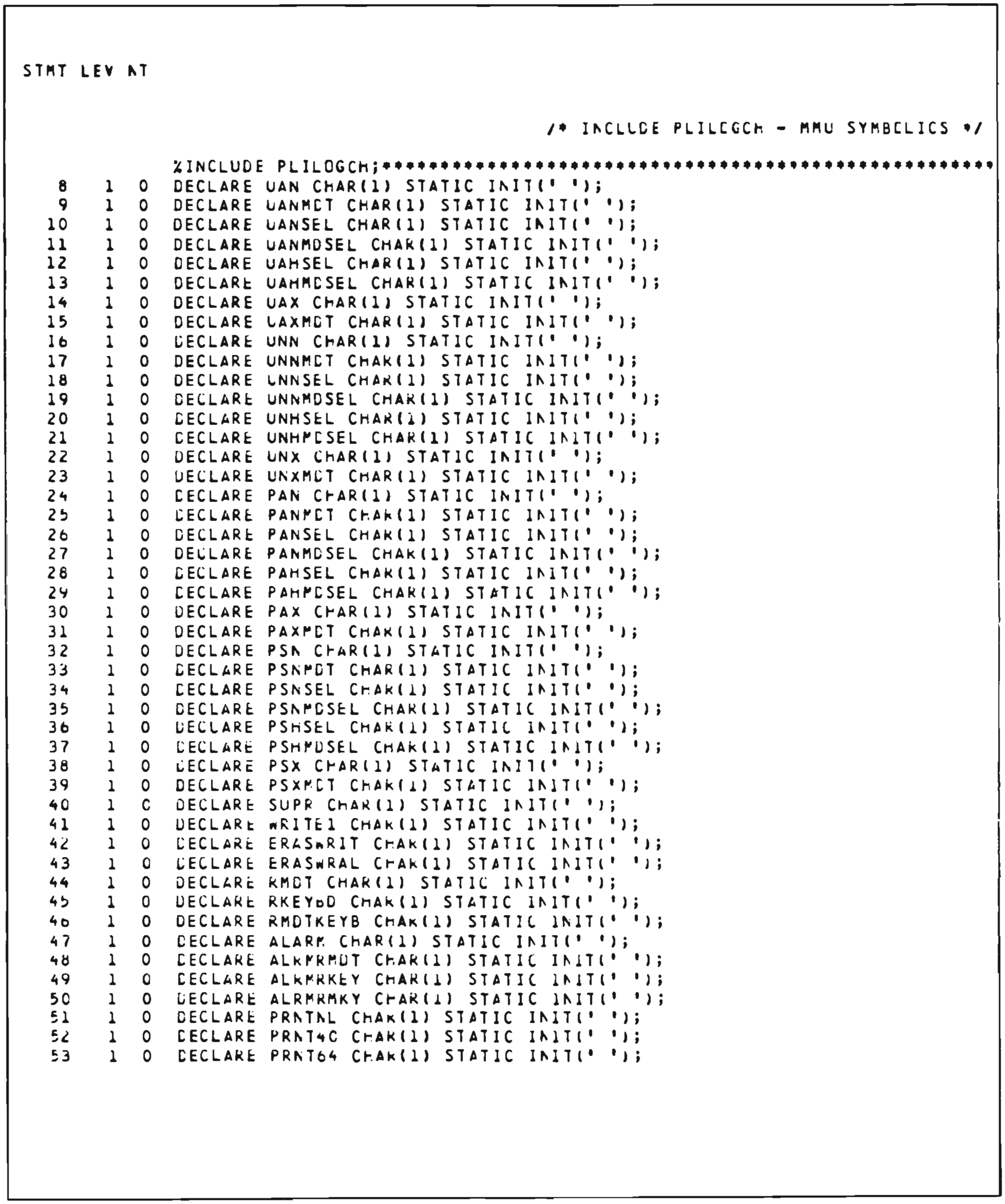
```
STMT LEV NT


                 /* PROCEDURE SQPL1A TO INQUIRE ON STOCK/PART FILES FOR MSG RESPONSE */

   1       0  SQPL1A: PROC (IN_MSG_PTR,SPA,SCT,RC)
                             OPTIONS(MAIN,REENTRANT); /* SUBSYSTEM 'PQ' - INQUIRY  */
                                                  /*  DEFINE THE INCOMING PARAMETERS      */
   2   1  0             DCL   (IN_MSG_PTR,         /* INPUT PARM 1 - INPUT MSG POINTER */
                              SPA,                 /* INPUT PARM 2 - SYSTEM PARM AREA  */
                              SCT) PTR;            /* INPUT PARM 3 - SUBSYSTEM ENTRY   */
   3   1  0             DCL   RC    FIXED BIN(31); /* INPUT PARM 4 - RETURN CODE       */

   4   1  0             DCL   SQPL1B ENTRY EXTERNAL;      /* ******** DEF SQPL1B ENTRY */

                                                   /* DEFINE ALL STATIC STORAGE VARIABLES */


   5   1  0             DCL 1 MAP_NAMES STATIC,                         /* FOR CALLS TO MPL */

                              3 IO_MAPGROUP CHAR(8) INIT('STKSTAT'), /*  MAPGROUP    */
                              3 IO_MAP      CHAR(8) INIT('MAP1'),     /*  NORMAL MAP */
                              3 ERROR_MAP   CHAR(8) INIT('ERRMAP');   /*  ERROR MAP   */

   6   1  0             DCL 1 FILE_NAMES STATIC,   /* FOR CALLS TO THE FILE HANDLER */

                              3 DD_STOCK CHAR(8) INIT('STCKFILE');
                 /*           3 DD_PART  CHAR(8) INIT('PARTFILE')   MOVED TO SQPL1B */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 1 of 18)



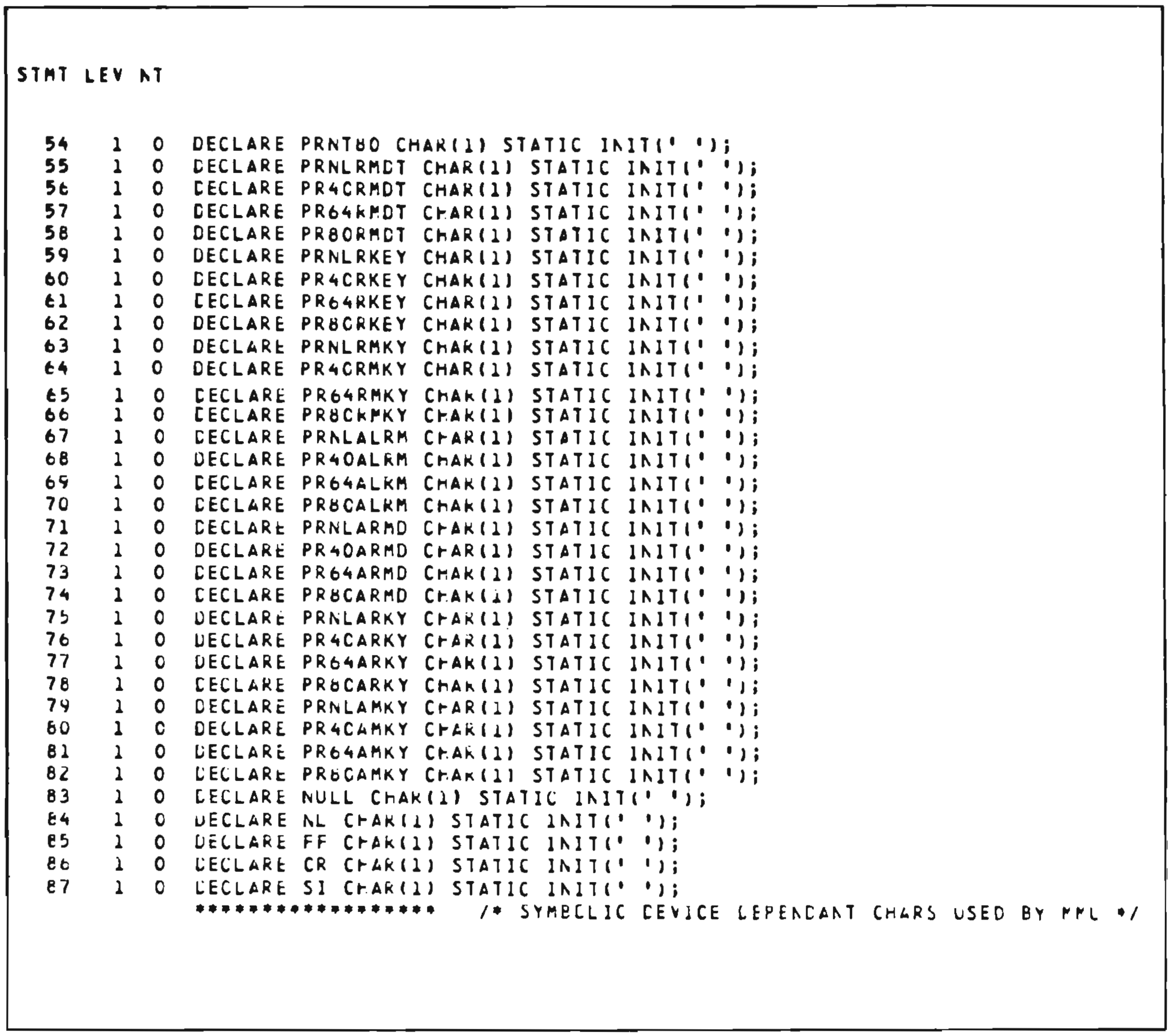
```
STMT LEV NT

                    /* INCLUDE PLIENTRY - DEFINES ICOM ENTRY POINTS - AS ASM INTER  */

               %INCLUDE PLIENTRY;************************************************************
   7   1   0   DECLARE  ( SELECT,
                          RELEASE,
                          READ,
                          WRITE,
                          GET,
                          PUT,
                          GETV,
                          PUTV,
                          RELEX,
                          FEOV,
                          COBPUT,
                          MSGCOL,
                          FESEND,
                          FESENDC,
                          COBSTORF,
                          CONVERSE,
                          LOGPUT,
                          DBINT,
                          PAGE,
                          QBUILD,
                          QOPEN,
                          QREAD,
                          QREADX,
                          QWRITE,
                          QWRITEX,
                          QCLOSE,
                          FECMDDQ,
                          FECMFDBK,
                          FECMRLSE,
                          MAPIN,
                          MAPOUT,
                          MAPFREE,
                          MAPEND,
                          MAPURGE,
                          MAPCLR,
                          DWSSNAP,                                          /* REL 10 */
                          INTSORTC,                                         /* REL 10 */
                          INTSTORE,
                          INTFETCH,
                          INTUNSTO) ENTRY OPTIONS (ASM INTER);
               ******************    /* FOR DIRECT CALLS TO ICOM AND USER ROUTINES */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 2 of 18)



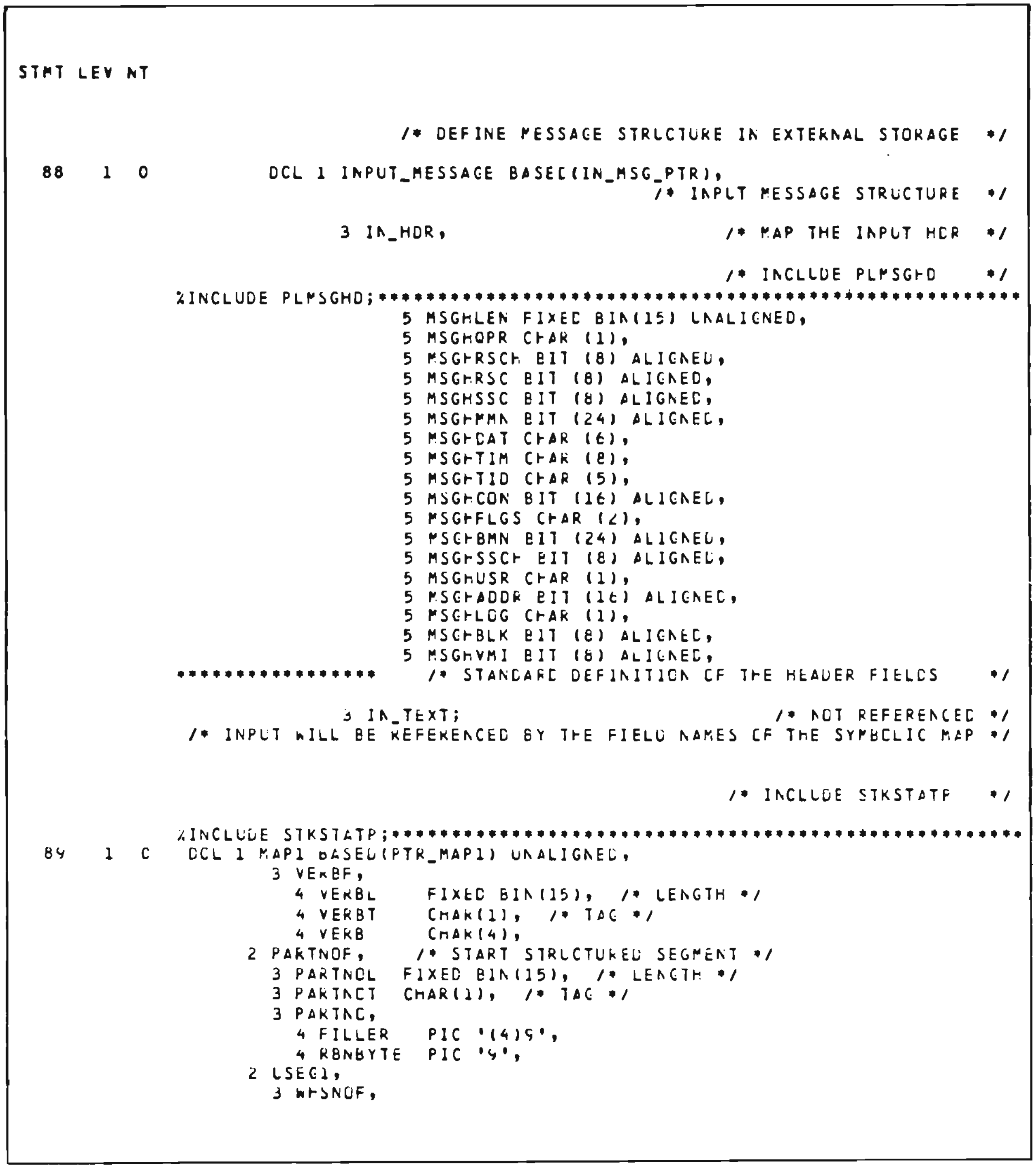
```
STMT LEV NT

                                          /* INCLUDE PLILOGCH - MMU SYMBOLICS */

                  %INCLUDE PLILOGCH;*******************************************************
   8    1   0     DECLARE UAN CHAR(1) STATIC INIT(' ');
   9    1   0     DECLARE UANMDT CHAR(1) STATIC INIT(' ');
  10    1   0     DECLARE UANSEL CHAR(1) STATIC INIT(' ');
  11    1   0     DECLARE UANMDSEL CHAR(1) STATIC INIT(' ');
  12    1   0     DECLARE UAHSEL CHAR(1) STATIC INIT(' ');
  13    1   0     DECLARE UAHMDSEL CHAR(1) STATIC INIT(' ');
  14    1   0     DECLARE UAX CHAR(1) STATIC INIT(' ');
  15    1   0     DECLARE UAXMDT CHAR(1) STATIC INIT(' ');
  16    1   0     DECLARE UNN CHAR(1) STATIC INIT(' ');
  17    1   0     DECLARE UNNMDT CHAR(1) STATIC INIT(' ');
  18    1   0     DECLARE UNNSEL CHAR(1) STATIC INIT(' ');
  19    1   0     DECLARE UNNMDSEL CHAR(1) STATIC INIT(' ');
  20    1   0     DECLARE UNHSEL CHAR(1) STATIC INIT(' ');
  21    1   0     DECLARE UNHMDSEL CHAR(1) STATIC INIT(' ');
  22    1   0     DECLARE UNX CHAR(1) STATIC INIT(' ');
  23    1   0     DECLARE UNXMDT CHAR(1) STATIC INIT(' ');
  24    1   0     DECLARE PAN CHAR(1) STATIC INIT(' ');
  25    1   0     DECLARE PANMDT CHAR(1) STATIC INIT(' ');
  26    1   0     DECLARE PANSEL CHAR(1) STATIC INIT(' ');
  27    1   0     DECLARE PANMDSEL CHAR(1) STATIC INIT(' ');
  28    1   0     DECLARE PAHSEL CHAR(1) STATIC INIT(' ');
  29    1   0     DECLARE PAHMDSEL CHAR(1) STATIC INIT(' ');
  30    1   0     DECLARE PAX CHAR(1) STATIC INIT(' ');
  31    1   0     DECLARE PAXMDT CHAR(1) STATIC INIT(' ');
  32    1   0     DECLARE PSN CHAR(1) STATIC INIT(' ');
  33    1   0     DECLARE PSNMDT CHAR(1) STATIC INIT(' ');
  34    1   0     DECLARE PSNSEL CHAR(1) STATIC INIT(' ');
  35    1   0     DECLARE PSNMDSEL CHAR(1) STATIC INIT(' ');
  36    1   0     DECLARE PSHSEL CHAR(1) STATIC INIT(' ');
  37    1   0     DECLARE PSHMDSEL CHAR(1) STATIC INIT(' ');
  38    1   0     DECLARE PSX CHAR(1) STATIC INIT(' ');
  39    1   0     DECLARE PSXMDT CHAR(1) STATIC INIT(' ');
  40    1   0     DECLARE SUPR CHAR(1) STATIC INIT(' ');
  41    1   0     DECLARE WRITE1 CHAR(1) STATIC INIT(' ');
  42    1   0     DECLARE ERASWRIT CHAR(1) STATIC INIT(' ');
  43    1   0     DECLARE ERASWRAL CHAR(1) STATIC INIT(' ');
  44    1   0     DECLARE RMDT CHAR(1) STATIC INIT(' ');
  45    1   0     DECLARE RKEYBD CHAR(1) STATIC INIT(' ');
  46    1   0     DECLARE RMDTKEYB CHAR(1) STATIC INIT(' ');
  47    1   0     DECLARE ALARM CHAR(1) STATIC INIT(' ');
  48    1   0     DECLARE ALRMRMDT CHAR(1) STATIC INIT(' ');
  49    1   0     DECLARE ALRMRKEY CHAR(1) STATIC INIT(' ');
  50    1   0     DECLARE ALRMRMKY CHAR(1) STATIC INIT(' ');
  51    1   0     DECLARE PRNTNL CHAR(1) STATIC INIT(' ');
  52    1   0     DECLARE PRNT40 CHAR(1) STATIC INIT(' ');
  53    1   0     DECLARE PRNT64 CHAR(1) STATIC INIT(' ');
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 3 of 18)



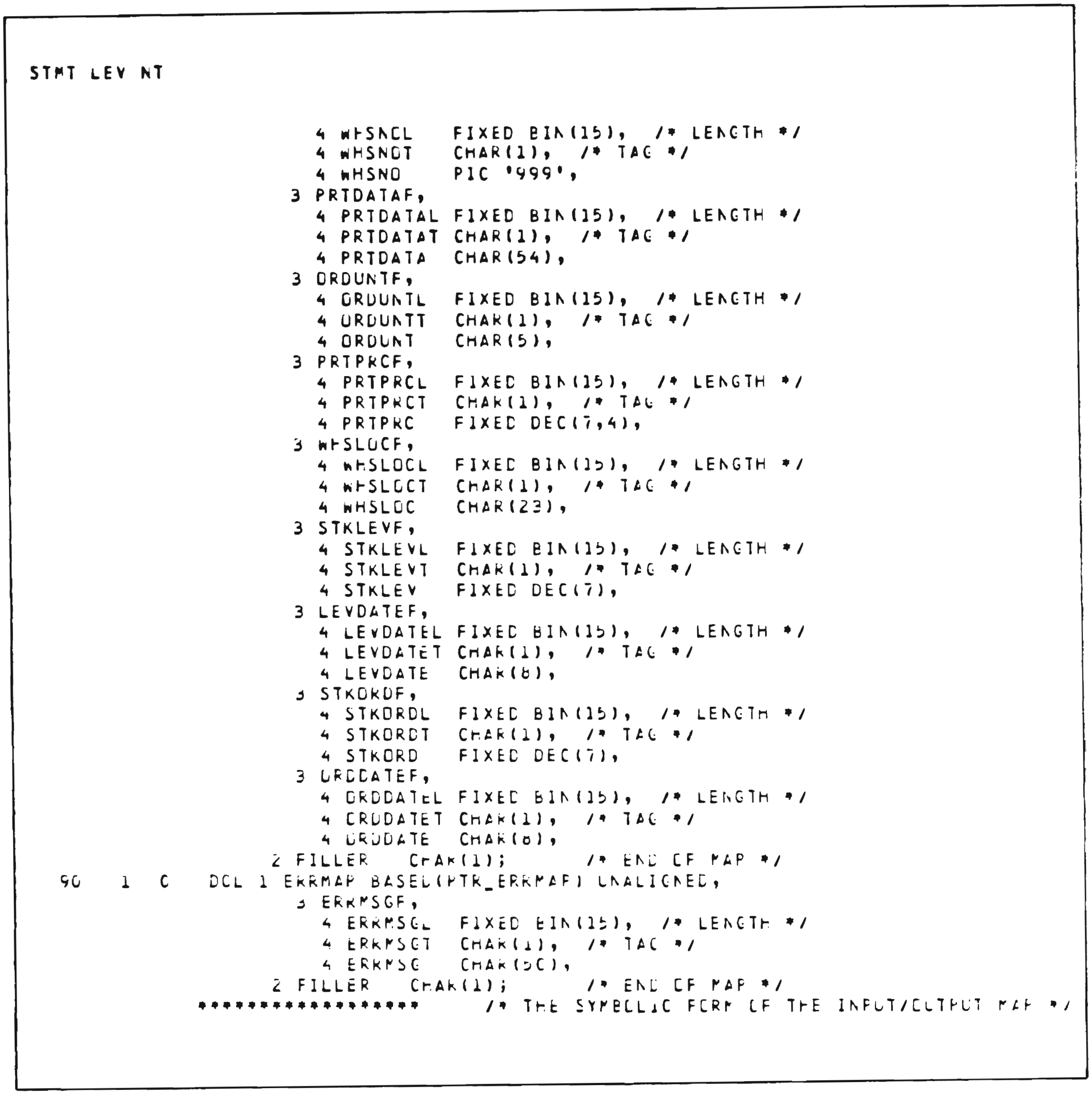
```
STMT LEV NT


  54    1  0  DECLARE PRNT80 CHAR(1) STATIC INIT(' ');
  55    1  0  DECLARE PRNLRMDT CHAR(1) STATIC INIT(' ');
  56    1  0  DECLARE PR40RMDT CHAR(1) STATIC INIT(' ');
  57    1  0  DECLARE PR64RMDT CHAR(1) STATIC INIT(' ');
  58    1  0  DECLARE PR80RMDT CHAR(1) STATIC INIT(' ');
  59    1  0  DECLARE PRNLRKEY CHAR(1) STATIC INIT(' ');
  60    1  0  DECLARE PR40RKEY CHAR(1) STATIC INIT(' ');
  61    1  0  DECLARE PR64RKEY CHAR(1) STATIC INIT(' ');
  62    1  0  DECLARE PR80RKEY CHAR(1) STATIC INIT(' ');
  63    1  0  DECLARE PRNLRMKY CHAR(1) STATIC INIT(' ');
  64    1  0  DECLARE PR40RMKY CHAR(1) STATIC INIT(' ');
  65    1  0  DECLARE PR64RMKY CHAR(1) STATIC INIT(' ');
  66    1  0  DECLARE PR80RMKY CHAR(1) STATIC INIT(' ');
  67    1  0  DECLARE PRNLALRM CHAR(1) STATIC INIT(' ');
  68    1  0  DECLARE PR40ALRM CHAR(1) STATIC INIT(' ');
  69    1  0  DECLARE PR64ALRM CHAR(1) STATIC INIT(' ');
  70    1  0  DECLARE PR80ALRM CHAR(1) STATIC INIT(' ');
  71    1  0  DECLARE PRNLARMD CHAR(1) STATIC INIT(' ');
  72    1  0  DECLARE PR40ARMD CHAR(1) STATIC INIT(' ');
  73    1  0  DECLARE PR64ARMD CHAR(1) STATIC INIT(' ');
  74    1  0  DECLARE PR80ARMD CHAR(1) STATIC INIT(' ');
  75    1  0  DECLARE PRNLARKY CHAR(1) STATIC INIT(' ');
  76    1  0  DECLARE PR40ARKY CHAR(1) STATIC INIT(' ');
  77    1  0  DECLARE PR64ARKY CHAR(1) STATIC INIT(' ');
  78    1  0  DECLARE PR80ARKY CHAR(1) STATIC INIT(' ');
  79    1  0  DECLARE PRNLAMKY CHAR(1) STATIC INIT(' ');
  80    1  0  DECLARE PR40AMKY CHAR(1) STATIC INIT(' ');
  81    1  0  DECLARE PR64AMKY CHAR(1) STATIC INIT(' ');
  82    1  0  DECLARE PR80AMKY CHAR(1) STATIC INIT(' ');
  83    1  0  DECLARE NULL CHAR(1) STATIC INIT(' ');
  84    1  0  DECLARE NL CHAR(1) STATIC INIT(' ');
  85    1  0  DECLARE FF CHAR(1) STATIC INIT(' ');
  86    1  0  DECLARE CR CHAR(1) STATIC INIT(' ');
  87    1  0  DECLARE SI CHAR(1) STATIC INIT(' ');
              ******************    /* SYMBOLIC DEVICE DEPENDANT CHARS USED BY MML */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 4 of 18)



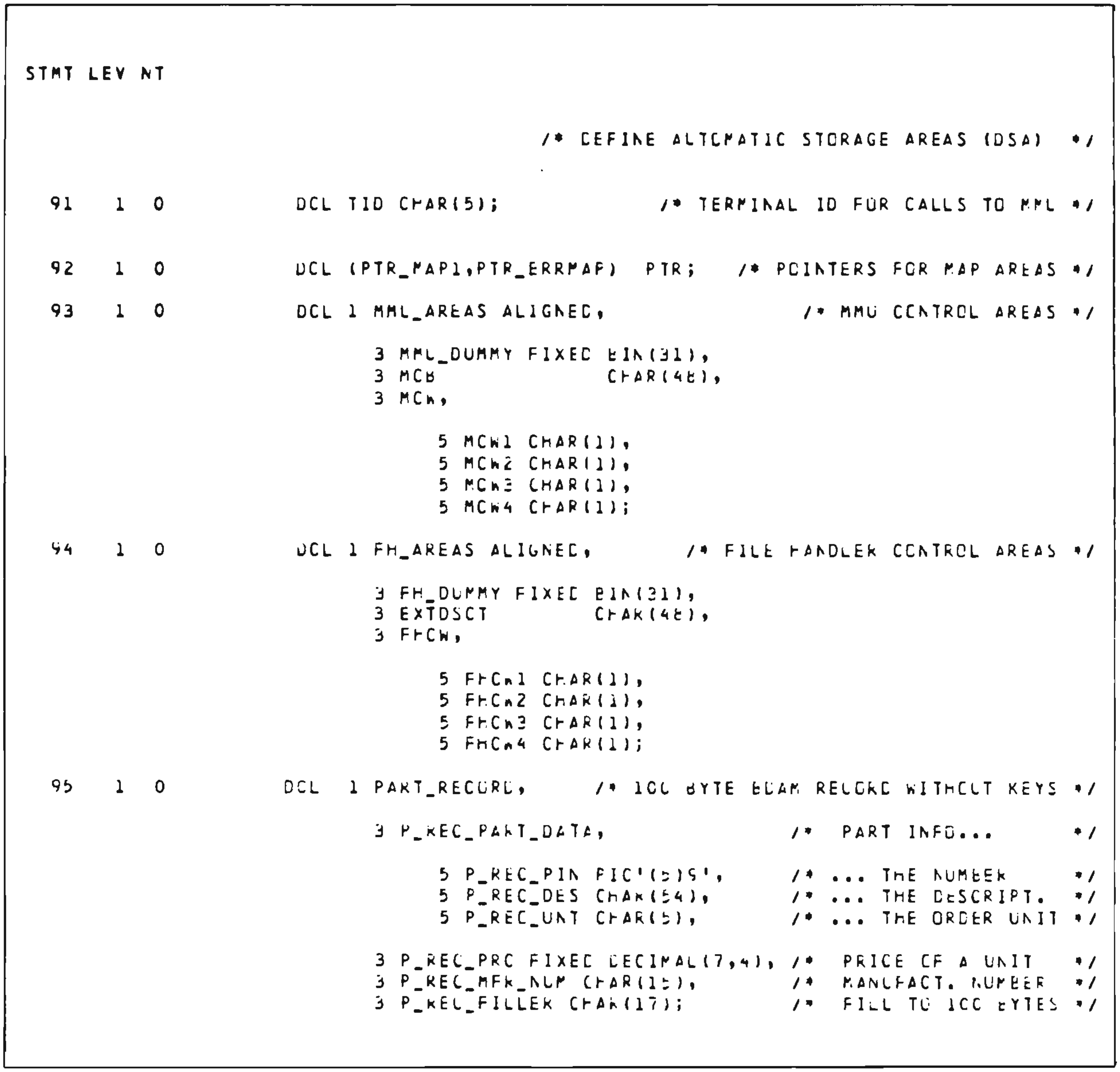
```
STMT LEV NT

                               /* DEFINE MESSAGE STRUCTURE IN EXTERNAL STORAGE  */

  88   1  0           DCL 1 INPUT_MESSAGE BASED(IN_MSG_PTR),
                                                   /* INPUT MESSAGE STRUCTURE  */

                            3 IN_HDR,                    /* MAP THE INPUT HDR  */

                                                         /* INCLUDE PLMSGHD    */
            %INCLUDE PLMSGHD;***************************************************
                               5 MSGHLEN FIXED BIN(15) UNALIGNED,
                               5 MSGHQPR CHAR (1),
                               5 MSGHRSCH BIT (8) ALIGNED,
                               5 MSGHRSC BIT (8) ALIGNED,
                               5 MSGHSSC BIT (8) ALIGNED,
                               5 MSGHMMN BIT (24) ALIGNED,
                               5 MSGHDAT CHAR (6),
                               5 MSGHTIM CHAR (8),
                               5 MSGHTID CHAR (5),
                               5 MSGHCON BIT (16) ALIGNED,
                               5 MSGHFLGS CHAR (2),
                               5 MSGHBMN BIT (24) ALIGNED,
                               5 MSGHSSCH BIT (8) ALIGNED,
                               5 MSGHUSR CHAR (1),
                               5 MSGHADDR BIT (16) ALIGNED,
                               5 MSGHLOG CHAR (1),
                               5 MSGHBLK BIT (8) ALIGNED,
                               5 MSGHVMI BIT (8) ALIGNED,
            ****************    /* STANDARD DEFINITION OF THE HEADER FIELDS    */

                            3 IN_TEXT;                      /* NOT REFERENCED */
             /* INPUT WILL BE REFERENCED BY THE FIELD NAMES OF THE SYMBOLIC MAP */


                                                         /* INCLUDE STKSTATP    */

            %INCLUDE STKSTATP;**************************************************
  89   1  0  DCL 1 MAP1 BASED(PTR_MAP1) UNALIGNED,
                    3 VERBF,
                      4 VERBL     FIXED BIN(15),  /* LENGTH */
                      4 VERBT     CHAR(1),  /* TAG */
                      4 VERB      CHAR(4),
                  2 PARTNOF,     /* START STRUCTURED SEGMENT */
                    3 PARTNOL   FIXED BIN(15),  /* LENGTH */
                    3 PARTNOT   CHAR(1),  /* TAG */
                    3 PARTNO,
                      4 FILLER    PIC '(4)9',
                      4 RBNBYTE   PIC '9',
                  2 LSEG1,
                    3 WHSNOF,
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 5 of 18)



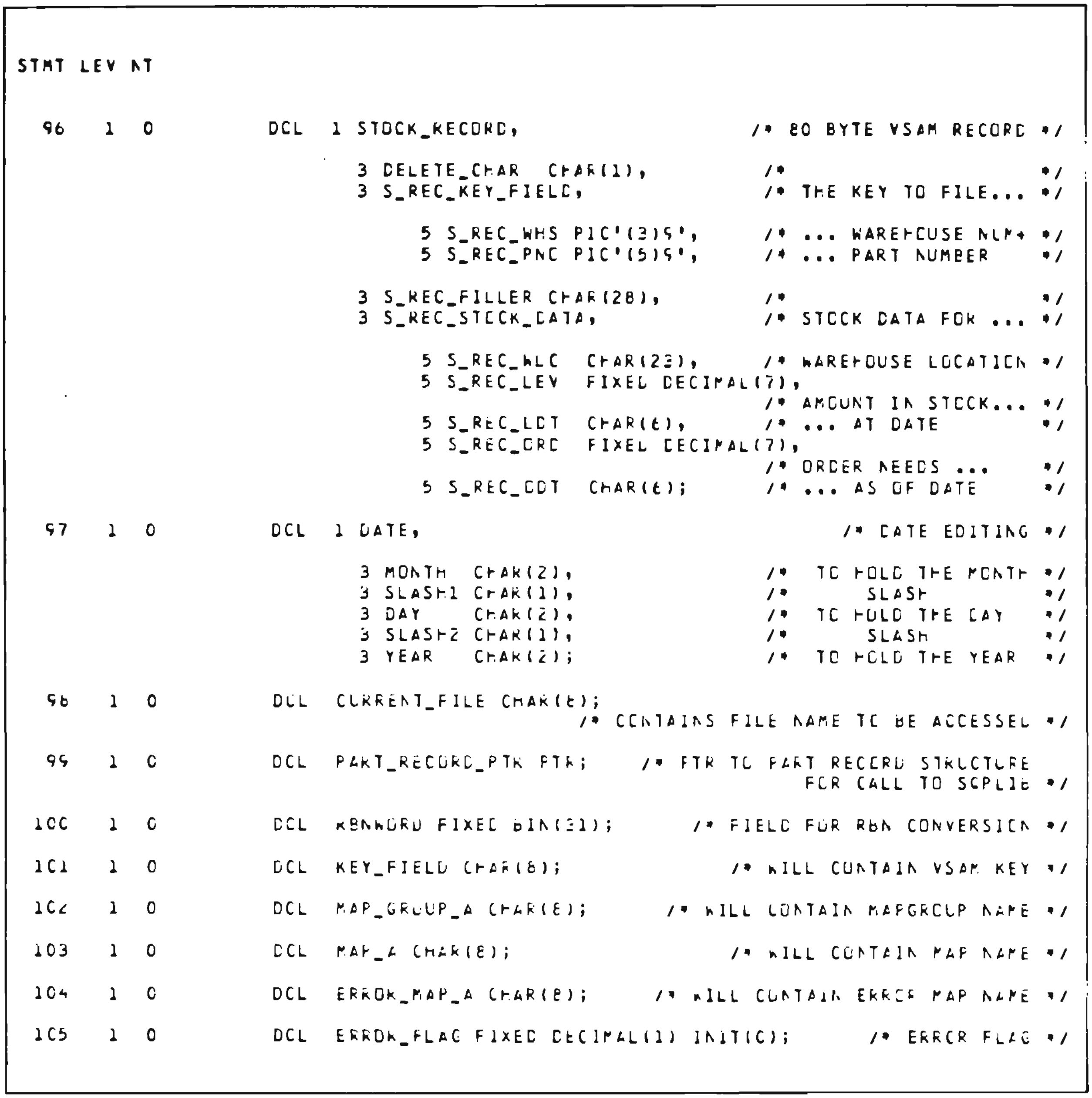
```
STMT LEV NT

                                  4 WHSNOL    FIXED BIN(15),  /* LENGTH */
                                  4 WHSNOT    CHAR(1),  /* TAG */
                                  4 WHSNO     PIC '999',
                                3 PRTDATAF,
                                  4 PRTDATAL FIXED BIN(15),  /* LENGTH */
                                  4 PRTDATAT CHAR(1),  /* TAG */
                                  4 PRTDATA  CHAR(54),
                                3 ORDUNTF,
                                  4 ORDUNTL   FIXED BIN(15),  /* LENGTH */
                                  4 ORDUNTT   CHAR(1),  /* TAG */
                                  4 ORDUNT    CHAR(5),
                                3 PRTPRCF,
                                  4 PRTPRCL   FIXED BIN(15),  /* LENGTH */
                                  4 PRTPRCT   CHAR(1),  /* TAG */
                                  4 PRTPRC    FIXED DEC(7,4),
                                3 WHSLOCF,
                                  4 WHSLOCL   FIXED BIN(15),  /* LENGTH */
                                  4 WHSLOCT   CHAR(1),  /* TAG */
                                  4 WHSLOC    CHAR(23),
                                3 STKLEVF,
                                  4 STKLEVL   FIXED BIN(15),  /* LENGTH */
                                  4 STKLEVT   CHAR(1),  /* TAG */
                                  4 STKLEV    FIXED DEC(7),
                                3 LEVDATEF,
                                  4 LEVDATEL FIXED BIN(15),  /* LENGTH */
                                  4 LEVDATET CHAR(1),  /* TAG */
                                  4 LEVDATE   CHAR(8),
                                3 STKORDF,
                                  4 STKORDL   FIXED BIN(15),  /* LENGTH */
                                  4 STKORDT   CHAR(1),  /* TAG */
                                  4 STKORD    FIXED DEC(7),
                                3 ORDDATEF,
                                  4 ORDDATEL FIXED BIN(15),  /* LENGTH */
                                  4 ORDDATET CHAR(1),  /* TAG */
                                  4 ORDDATE   CHAR(8),
                              2 FILLER    CHAR(1);        /* END OF MAP */
  90    1   0    DCL 1 ERRMAP BASED(PTR_ERRMAP) UNALIGNED,
                                3 ERRMSGF,
                                  4 ERRMSGL   FIXED BIN(15),  /* LENGTH */
                                  4 ERRMSGT   CHAR(1),  /* TAG */
                                  4 ERRMSG    CHAR(50),
                              2 FILLER    CHAR(1);        /* END OF MAP */
            *******************      /* THE SYMBOLIC FORM OF THE INPUT/OUTPUT MAP */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 6 of 18)



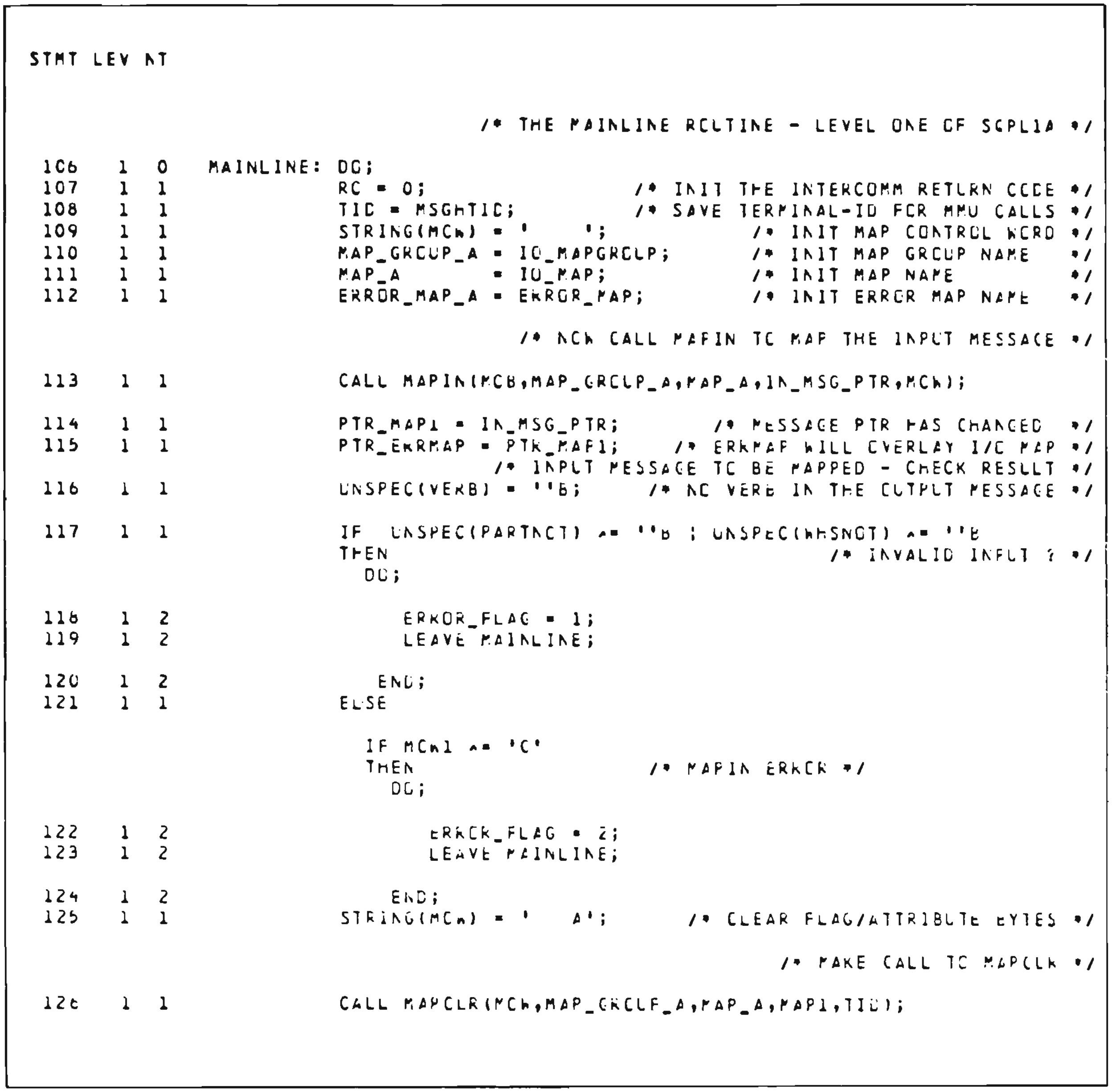
```
STMT LEV NT

                                     /* DEFINE AUTOMATIC STORAGE AREAS (DSA)  */

  91    1  0         DCL TID CHAR(5);            /* TERMINAL ID FOR CALLS TO MML */

  92    1  0         DCL (PTR_MAP1,PTR_ERRMAP)  PTR;   /* POINTERS FOR MAP AREAS */

  93    1  0         DCL 1 MML_AREAS ALIGNED,               /* MMU CONTROL AREAS */

                           3 MML_DUMMY FIXED BIN(31),
                           3 MCB             CHAR(48),
                           3 MCW,

                               5 MCW1 CHAR(1),
                               5 MCW2 CHAR(1),
                               5 MCW3 CHAR(1),
                               5 MCW4 CHAR(1);

  94    1  0         DCL 1 FH_AREAS ALIGNED,       /* FILE HANDLER CONTROL AREAS */

                           3 FH_DUMMY FIXED BIN(31),
                           3 EXTDSCT        CHAR(48),
                           3 FHCW,

                               5 FHCW1 CHAR(1),
                               5 FHCW2 CHAR(1),
                               5 FHCW3 CHAR(1),
                               5 FHCW4 CHAR(1);

  95    1  0        DCL  1 PART_RECORD,     /* 100 BYTE BDAM RECORD WITHOUT KEYS */

                           3 P_REC_PART_DATA,                /*  PART INFO...      */

                               5 P_REC_PIN PIC'(5)9',      /* ... THE NUMBER     */
                               5 P_REC_DES CHAR(54),       /* ... THE DESCRIPT.  */
                               5 P_REC_UNT CHAR(5),        /* ... THE ORDER UNIT */

                           3 P_REC_PRC FIXED DECIMAL(7,4), /*  PRICE OF A UNIT   */
                           3 P_REC_MFR_NUM CHAR(15),       /*  MANUFACT. NUMBER  */
                           3 P_REC_FILLER CHAR(17);        /*  FILL TO 100 BYTES */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 7 of 18)



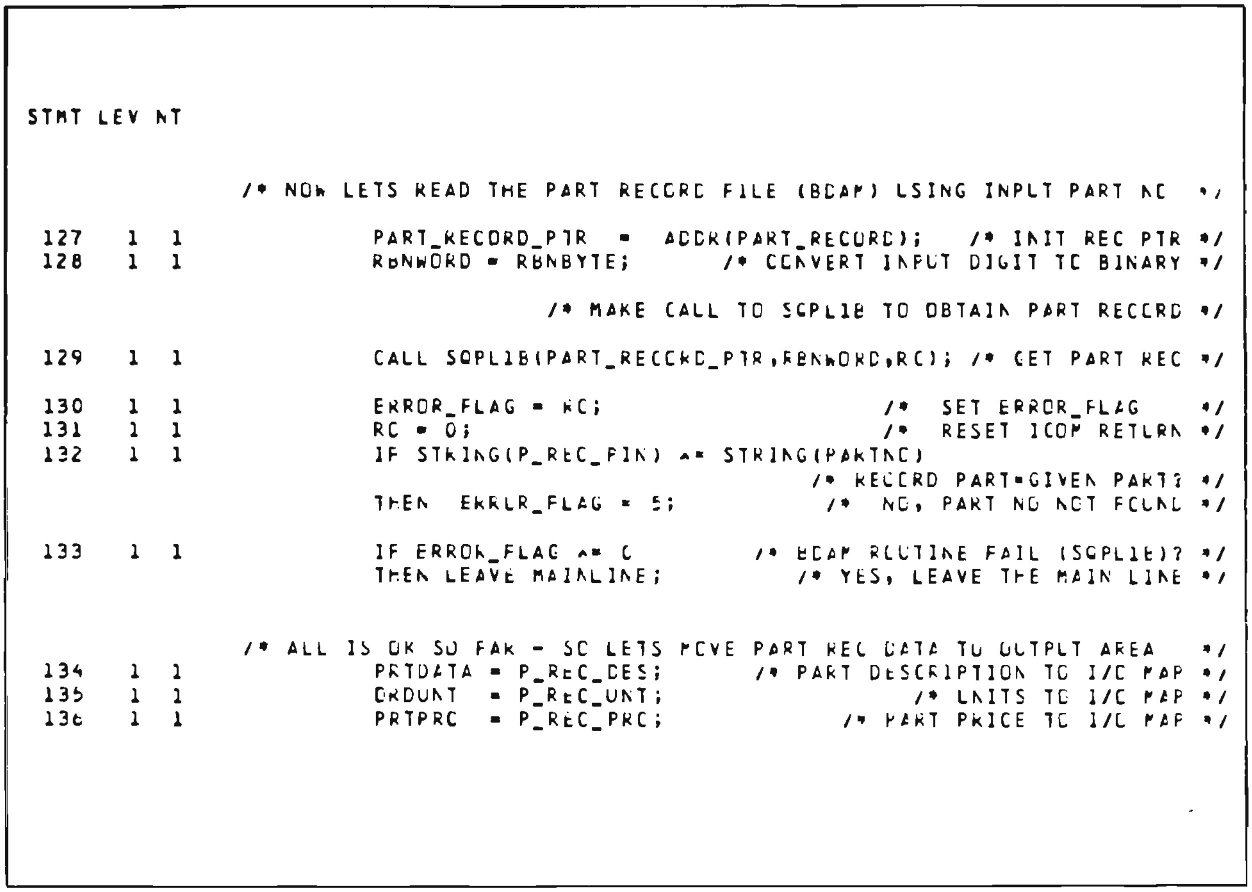
```
STMT LEV NT

  96   1  0         DCL  1 STOCK_RECORD,                    /* 80 BYTE VSAM RECORD */

                           3 DELETE_CHAR  CHAR(1),          /*                     */
                           3 S_REC_KEY_FIELD,               /* THE KEY TO FILE...  */

                                5 S_REC_WHS PIC'(3)9',      /* ... WAREHOUSE NUM+  */
                                5 S_REC_PNC PIC'(5)9',      /* ... PART NUMBER     */

                           3 S_REC_FILLER CHAR(28),         /*                     */
                           3 S_REC_STOCK_DATA,              /* STOCK DATA FOR ...  */

                                5 S_REC_WLC  CHAR(23),      /* WAREHOUSE LOCATION  */
                                5 S_REC_LEV  FIXED DECIMAL(7),
                                                            /* AMOUNT IN STOCK...  */
                                5 S_REC_LDT  CHAR(6),       /* ... AT DATE         */
                                5 S_REC_ORD  FIXED DECIMAL(7),
                                                            /* ORDER NEEDS ...     */
                                5 S_REC_ODT  CHAR(6);       /* ... AS OF DATE      */

  97   1  0         DCL  1 DATE,                                    /* DATE EDITING */

                           3 MONTH  CHAR(2),                /*  TO HOLD THE MONTH  */
                           3 SLASH1 CHAR(1),                /*      SLASH          */
                           3 DAY    CHAR(2),                /*  TO HOLD THE DAY    */
                           3 SLASH2 CHAR(1),                /*      SLASH          */
                           3 YEAR   CHAR(2);                /*  TO HOLD THE YEAR   */

  98   1  0         DCL  CURRENT_FILE CHAR(8);
                                        /* CONTAINS FILE NAME TO BE ACCESSED */

  99   1  0         DCL  PART_RECORD_PTR PTR;     /* PTR TO PART RECORD STRUCTURE
                                                              FOR CALL TO SQPL1B */

 100   1  0         DCL  RBNWORD FIXED BIN(31);       /* FIELD FOR RBN CONVERSION */

 101   1  0         DCL  KEY_FIELD CHAR(8);               /* WILL CONTAIN VSAM KEY */

 102   1  0         DCL  MAP_GROUP_A CHAR(8);      /* WILL CONTAIN MAPGROUP NAME */

 103   1  0         DCL  MAP_A CHAR(8);                   /* WILL CONTAIN MAP NAME */

 104   1  0         DCL  ERROR_MAP_A CHAR(8);     /* WILL CONTAIN ERROR MAP NAME */

 105   1  0         DCL  ERROR_FLAG FIXED DECIMAL(1) INIT(0);        /* ERROR FLAG */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 8 of 18)



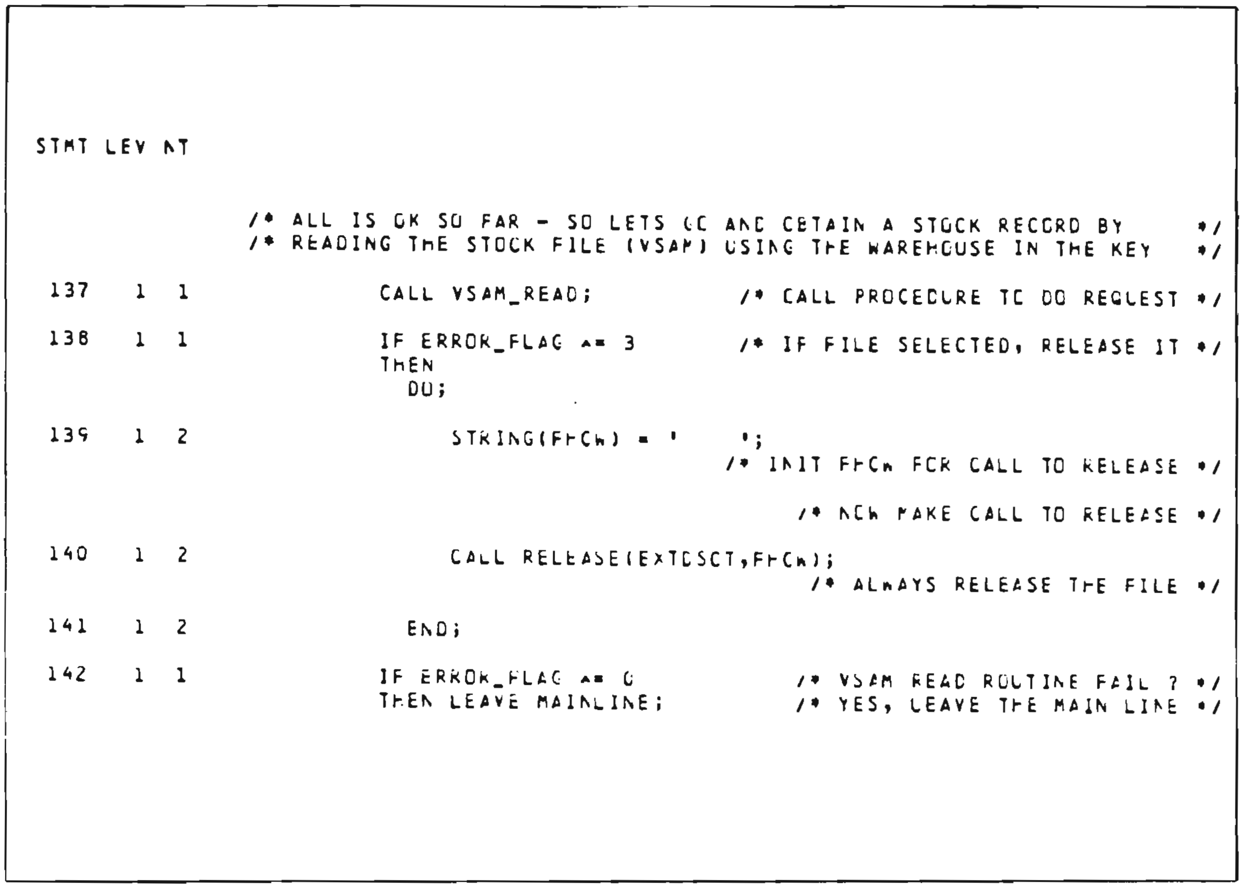
```
STMT LEV NT

                                    /* THE MAINLINE ROUTINE - LEVEL ONE OF SQPL1A */

 106    1  0   MAINLINE: DO;
 107    1  1             RC = 0;                  /* INIT THE INTERCOMM RETURN CODE */
 108    1  1             TID = MSGHTID;           /* SAVE TERMINAL-ID FOR MMU CALLS */
 109    1  1             STRING(MCW) = '    ';           /* INIT MAP CONTROL WORD */
 110    1  1             MAP_GROUP_A = IO_MAPGROUP;      /* INIT MAP GROUP NAME   */
 111    1  1             MAP_A       = IO_MAP;           /* INIT MAP NAME         */
 112    1  1             ERROR_MAP_A = ERROR_MAP;        /* INIT ERROR MAP NAME   */

                                      /* NOW CALL MAPIN TO MAP THE INPUT MESSAGE */

 113    1  1             CALL MAPIN(MCB,MAP_GROUP_A,MAP_A,IN_MSG_PTR,MCW);

 114    1  1             PTR_MAP1 = IN_MSG_PTR;        /* MESSAGE PTR HAS CHANGED  */
 115    1  1             PTR_ERRMAP = PTR_MAP1;     /* ERRMAP WILL OVERLAY I/O MAP */
                                   /* INPUT MESSAGE TO BE MAPPED - CHECK RESULT */
 116    1  1             UNSPEC(VERB) = ''B;       /* NO VERB IN THE OUTPUT MESSAGE */

 117    1  1             IF  UNSPEC(PARTNOT) ¬= ''B | UNSPEC(WHSNOT) ¬= ''B
                         THEN                                  /* INVALID INPUT ? */
                           DO;

 118    1  2                 ERROR_FLAG = 1;
 119    1  2                 LEAVE MAINLINE;

 120    1  2               END;
 121    1  1             ELSE

                           IF MCW1 ¬= 'C'
                           THEN                     /* MAPIN ERROR */
                             DO;

 122    1  2                   ERROR_FLAG = 2;
 123    1  2                   LEAVE MAINLINE;

 124    1  2                 END;
 125    1  1             STRING(MCW) = '   A';       /* CLEAR FLAG/ATTRIBUTE BYTES */

                                                           /* MAKE CALL TO MAPCLR */

 126    1  1             CALL MAPCLR(MCW,MAP_GROUP_A,MAP_A,MAP1,TID);
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 9 of 18)



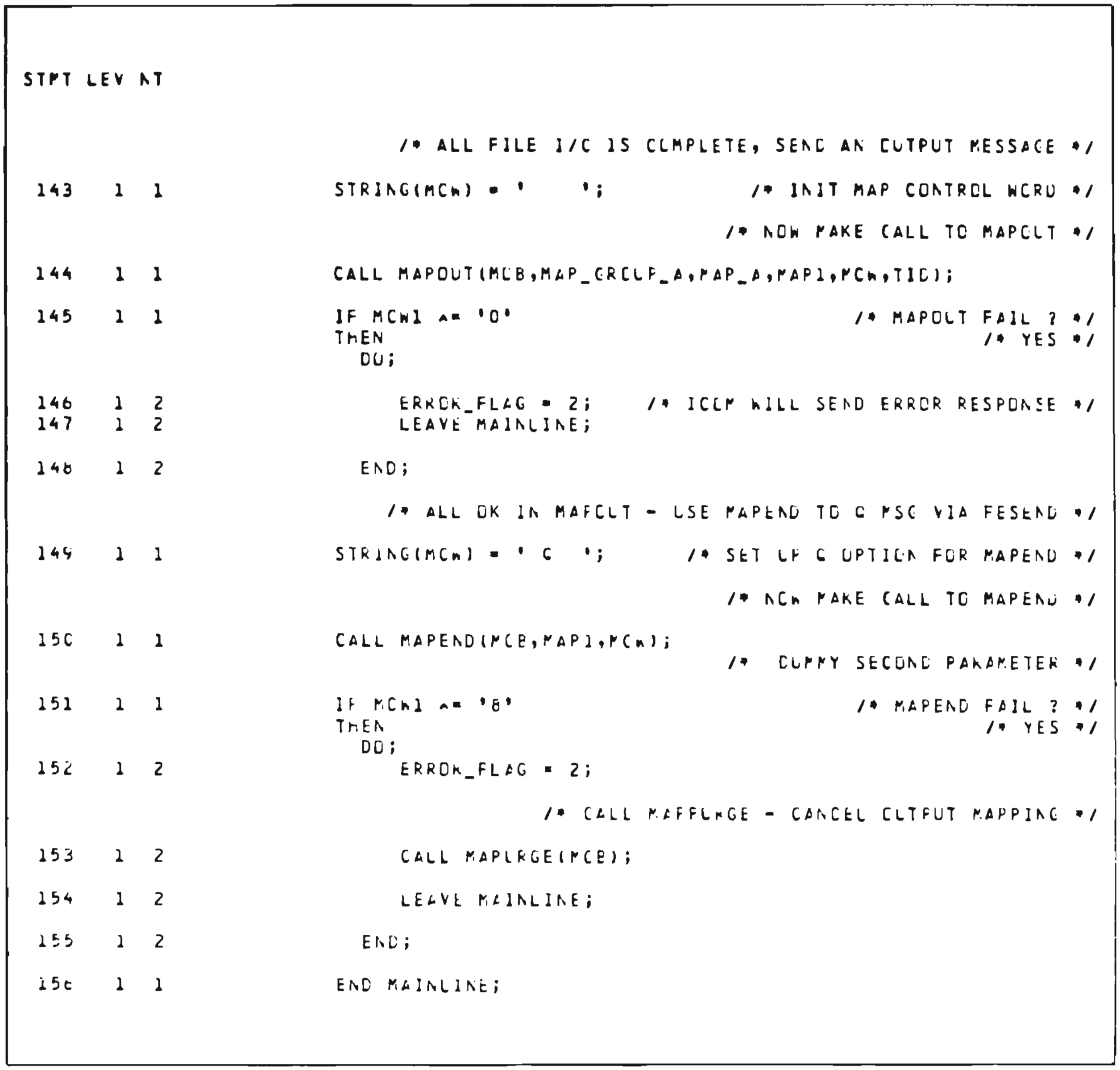
```
STMT LEV NT

               /* NOW LETS READ THE PART RECORD FILE (BDAM) USING INPUT PART NO   */

 127   1  1             PART_RECORD_PTR   =  ADDR(PART_RECORD);   /* INIT REC PTR */
 128   1  1             RBNWORD = RBNBYTE;       /* CONVERT INPUT DIGIT TO BINARY */

                                    /* MAKE CALL TO SQPL1B TO OBTAIN PART RECORD */

 129   1  1             CALL SQPL1B(PART_RECORD_PTR,RBNWORD,RC); /* GET PART REC */

 130   1  1             ERROR_FLAG = RC;                    /*  SET ERROR_FLAG    */
 131   1  1             RC = 0;                             /*  RESET ICOM RETURN */
 132   1  1             IF STRING(P_REC_PIN) ¬= STRING(PARTNO)
                                                       /* RECORD PART=GIVEN PART? */
                        THEN  ERROR_FLAG = 5;           /*  NO, PART NO NOT FOUND */

 133   1  1             IF ERROR_FLAG ¬= 0         /* BDAM ROUTINE FAIL (SQPL1B)? */
                        THEN LEAVE MAINLINE;          /* YES, LEAVE THE MAIN LINE */


               /* ALL IS OK SO FAR - SO LETS MOVE PART REC DATA TO OUTPUT AREA   */
 134   1  1             PRTDATA = P_REC_DES;       /* PART DESCRIPTION TO I/O MAP */
 135   1  1             ORDUNT  = P_REC_UNT;                  /* UNITS TO I/O MAP */
 136   1  1             PRTPRC  = P_REC_PRC;             /* PART PRICE TO I/O MAP */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 10 of 18)



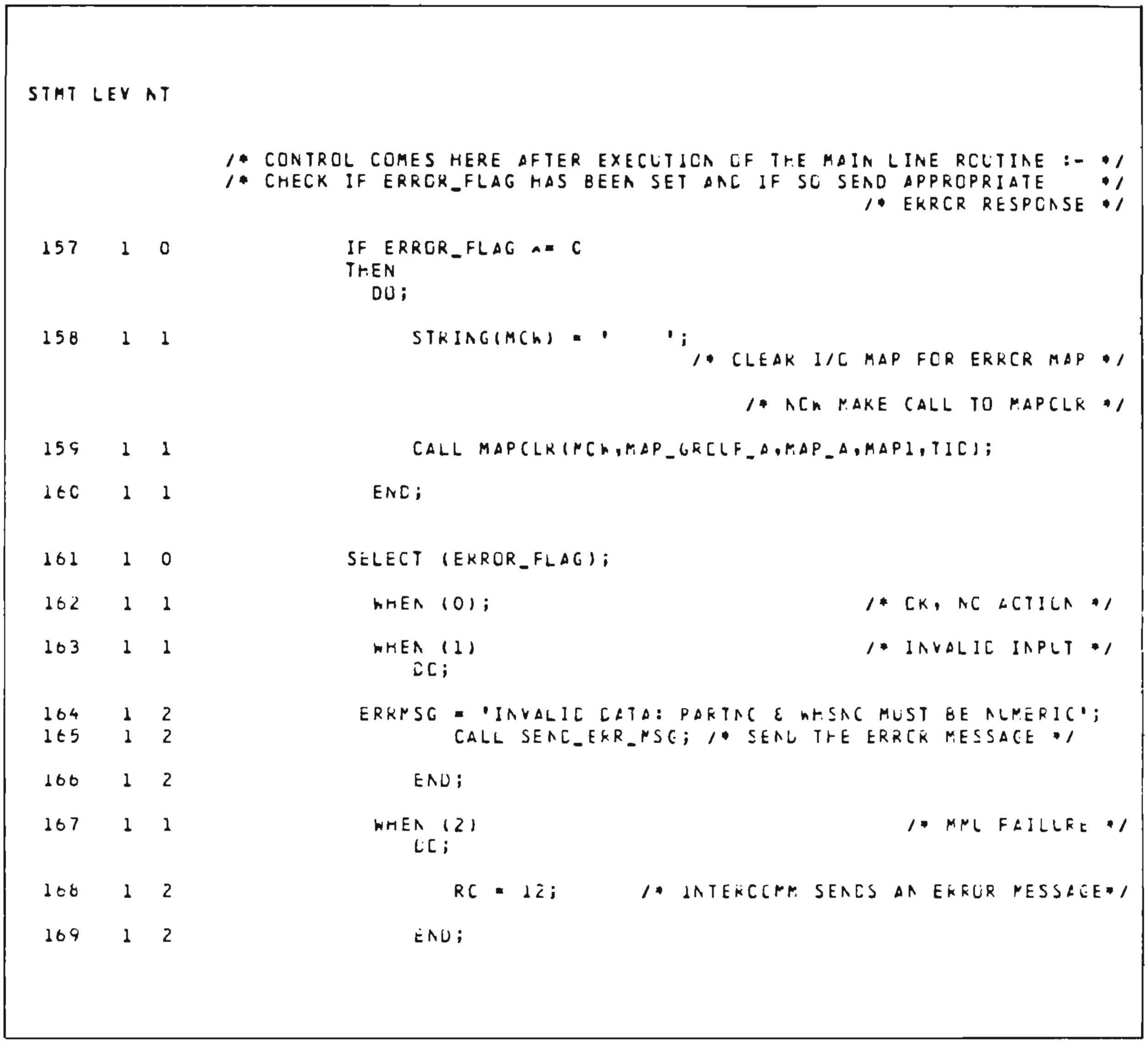
```
 STMT LEV NT


                     /* ALL IS OK SO FAR - SO LETS GO AND OBTAIN A STOCK RECORD BY        */
                     /* READING THE STOCK FILE (VSAM) USING THE WAREHOUSE IN THE KEY      */

  137    1   1                 CALL VSAM_READ;             /* CALL PROCEDURE TO DO REQUEST */

  138    1   1                 IF ERROR_FLAG ¬= 3          /* IF FILE SELECTED, RELEASE IT */
                               THEN
                                 DO;

  139    1   2                     STRING(FFCW) = '      ';
                                                          /* INIT FFCW FOR CALL TO RELEASE */

                                                              /* NOW MAKE CALL TO RELEASE */

  140    1   2                     CALL RELEASE(EXTDSCT,FFCW);
                                                               /* ALWAYS RELEASE THE FILE */

  141    1   2                   END;

  142    1   1                 IF ERROR_FLAG ¬= 0              /* VSAM READ ROUTINE FAIL ? */
                               THEN LEAVE MAINLINE;            /* YES, LEAVE THE MAIN LINE */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 11 of 18)



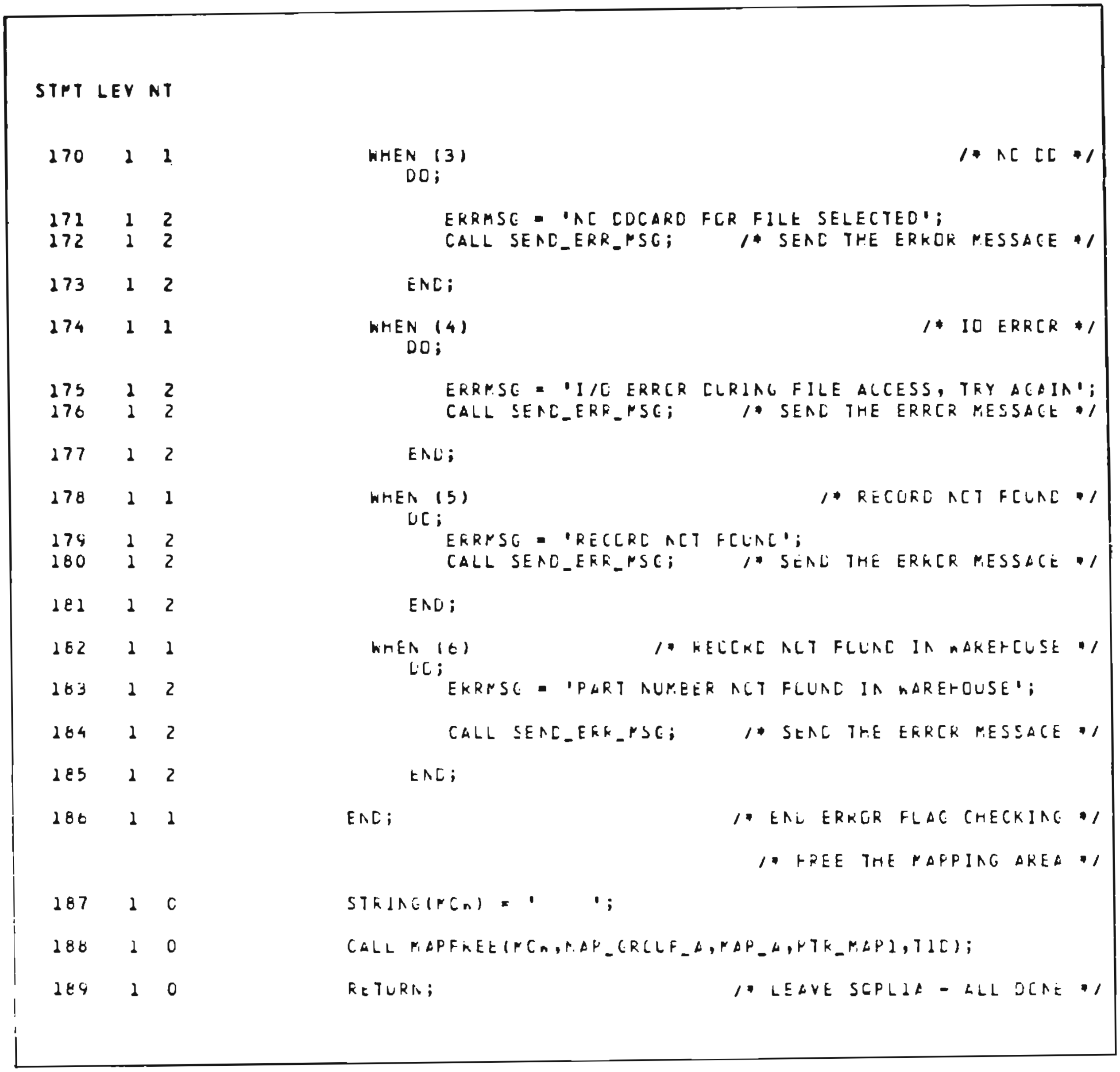
```
STMT LEV NT

                                    /* ALL FILE I/O IS COMPLETE, SEND AN OUTPUT MESSAGE */

 143   1  1            STRING(MCW) = '    ';             /* INIT MAP CONTROL WORD */

                                                           /* NOW MAKE CALL TO MAPOUT */

 144   1  1            CALL MAPOUT(MCB,MAP_GROUP_A,MAP_A,MAP1,MCW,TID);

 145   1  1            IF MCW1 ^= '0'                          /* MAPOUT FAIL ? */
                       THEN                                             /* YES */
                         DO;

 146   1  2               ERROR_FLAG = 2;     /* IOCP WILL SEND ERROR RESPONSE */
 147   1  2               LEAVE MAINLINE;

 148   1  2            END;

                                   /* ALL OK IN MAPOUT - USE MAPEND TO Q MSG VIA FESEND */

 149   1  1            STRING(MCW) = ' Q  ';      /* SET UP Q OPTION FOR MAPEND */

                                                           /* NOW MAKE CALL TO MAPEND */

 150   1  1            CALL MAPEND(MCB,MAP1,MCW);
                                                          /*  DUMMY SECOND PARAMETER */

 151   1  1            IF MCW1 ^= '8'                          /* MAPEND FAIL ? */
                       THEN                                             /* YES */
                         DO;
 152   1  2               ERROR_FLAG = 2;

                                          /* CALL MAPPURGE - CANCEL OUTPUT MAPPING */

 153   1  2               CALL MAPURGE(MCB);

 154   1  2               LEAVE MAINLINE;

 155   1  2             END;

 156   1  1            END MAINLINE;
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 12 of 18)



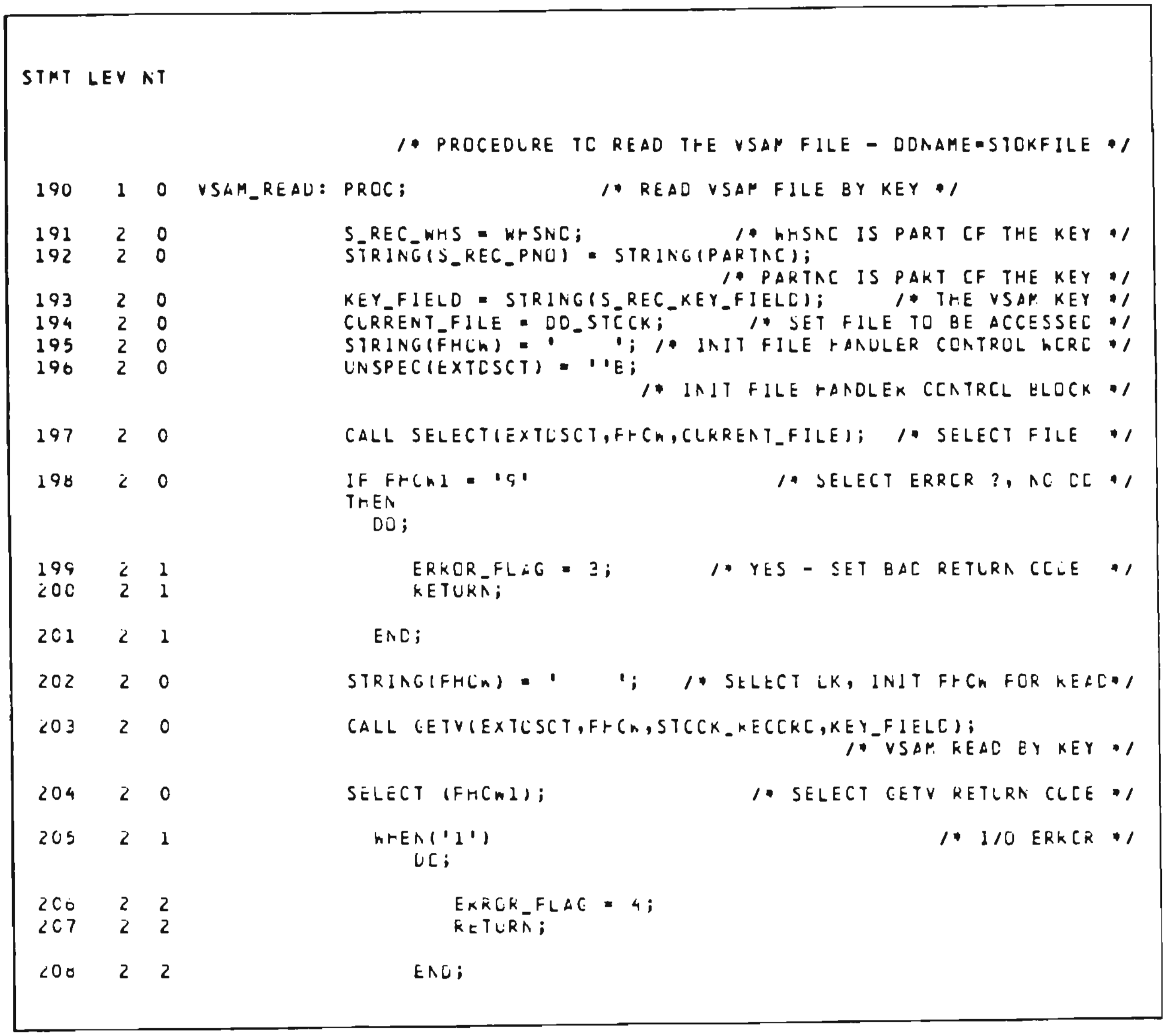
```
STMT LEV NT


                /* CONTROL COMES HERE AFTER EXECUTION OF THE MAIN LINE ROUTINE :- */
                /* CHECK IF ERROR_FLAG HAS BEEN SET AND IF SO SEND APPROPRIATE    */
                                                               /* ERROR RESPONSE */

 157   1  0                  IF ERROR_FLAG ^= 0
                             THEN
                               DO;

 158   1  1                        STRING(MOW) = '     ';
                                                  /* CLEAR I/O MAP FOR ERROR MAP */

                                                      /* NOW MAKE CALL TO MAPCLR */

 159   1  1                        CALL MAPCLR(MOW,MAP_GROUP_A,MAP_A,MAP1,TIO);

 160   1  1                    END;


 161   1  0                  SELECT (ERROR_FLAG);

 162   1  1                    WHEN (0);                           /* OK, NO ACTION */

 163   1  1                    WHEN (1)                            /* INVALID INPUT */
                                  DO;

 164   1  2                  ERRMSG = 'INVALID DATA: PARTNO & WHSNO MUST BE NUMERIC';
 165   1  2                          CALL SEND_ERR_MSG; /* SEND THE ERROR MESSAGE */

 166   1  2                       END;

 167   1  1                    WHEN (2)                                /* MML FAILURE */
                                  DO;

 168   1  2                          RC = 12;      /* INTERCOMM SENDS AN ERROR MESSAGE*/

 169   1  2                       END;
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 13 of 18)



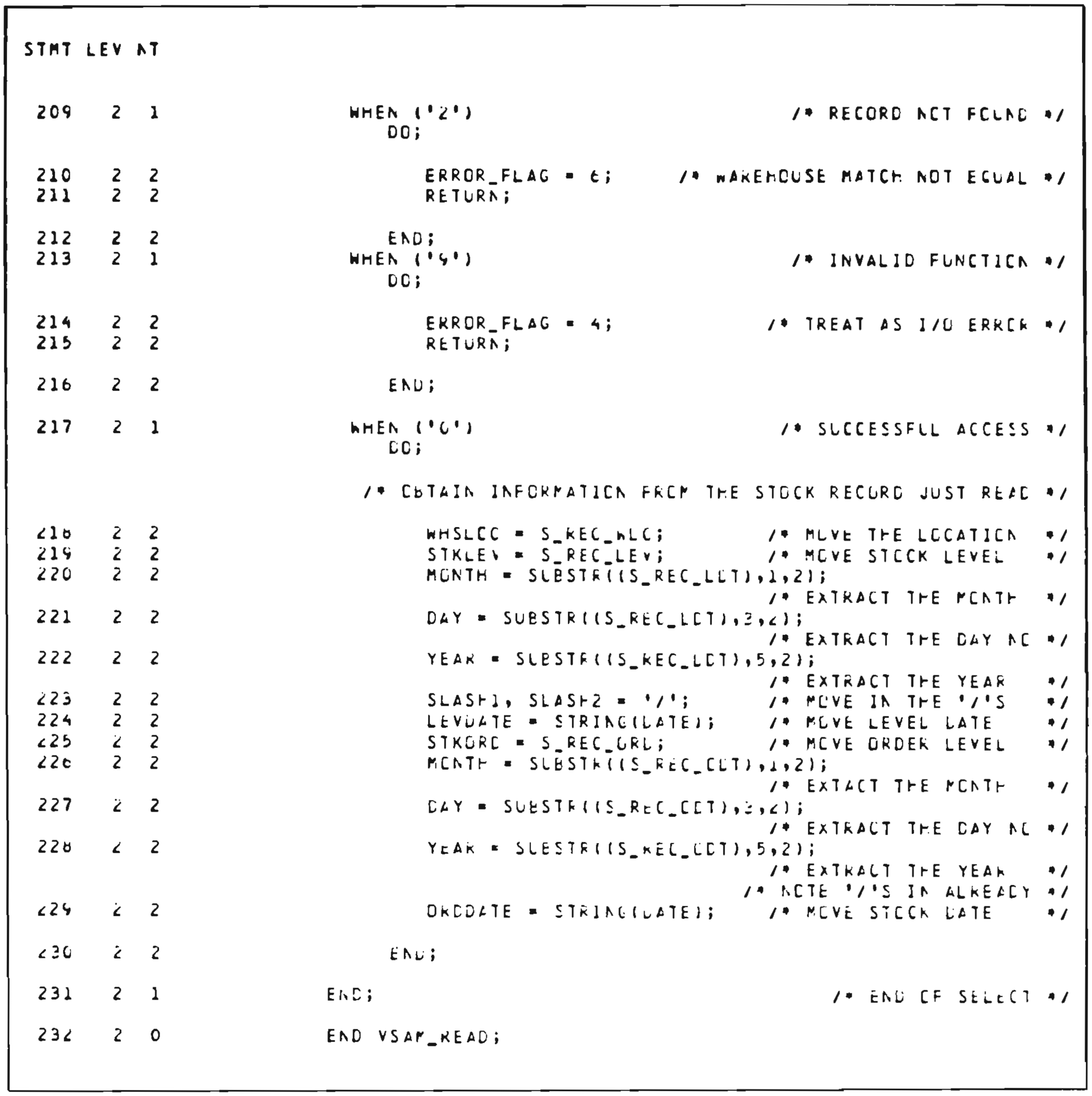
```
STMT LEV NT

 170   1  1            WHEN (3)                                    /* NO DD */
                          DO;

 171   1  2               ERRMSG = 'NO DDCARD FOR FILE SELECTED';
 172   1  2               CALL SEND_ERR_MSG;      /* SEND THE ERROR MESSAGE */

 173   1  2            END;

 174   1  1            WHEN (4)                                  /* IO ERROR */
                          DO;

 175   1  2               ERRMSG = 'I/O ERROR DURING FILE ACCESS, TRY AGAIN';
 176   1  2               CALL SEND_ERR_MSG;      /* SEND THE ERROR MESSAGE */

 177   1  2            END;

 178   1  1            WHEN (5)                            /* RECORD NOT FOUND */
                          DO;
 179   1  2               ERRMSG = 'RECORD NOT FOUND';
 180   1  2               CALL SEND_ERR_MSG;      /* SEND THE ERROR MESSAGE */

 181   1  2            END;

 182   1  1            WHEN (6)          /* RECORD NOT FOUND IN WAREHOUSE */
                          DO;
 183   1  2               ERRMSG = 'PART NUMBER NOT FOUND IN WAREHOUSE';

 184   1  2               CALL SEND_ERR_MSG;      /* SEND THE ERROR MESSAGE */

 185   1  2            END;

 186   1  1         END;                           /* END ERROR FLAG CHECKING */

                                                  /* FREE THE MAPPING AREA */

 187   1  0         STRING(MCW) = '    ';

 188   1  0         CALL MAPFREE(MCW,MAP_GROUP_A,MAP_A,PTR_MAP1,TID);

 189   1  0         RETURN;                      /* LEAVE SQPL1A - ALL DONE */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 14 of 18)



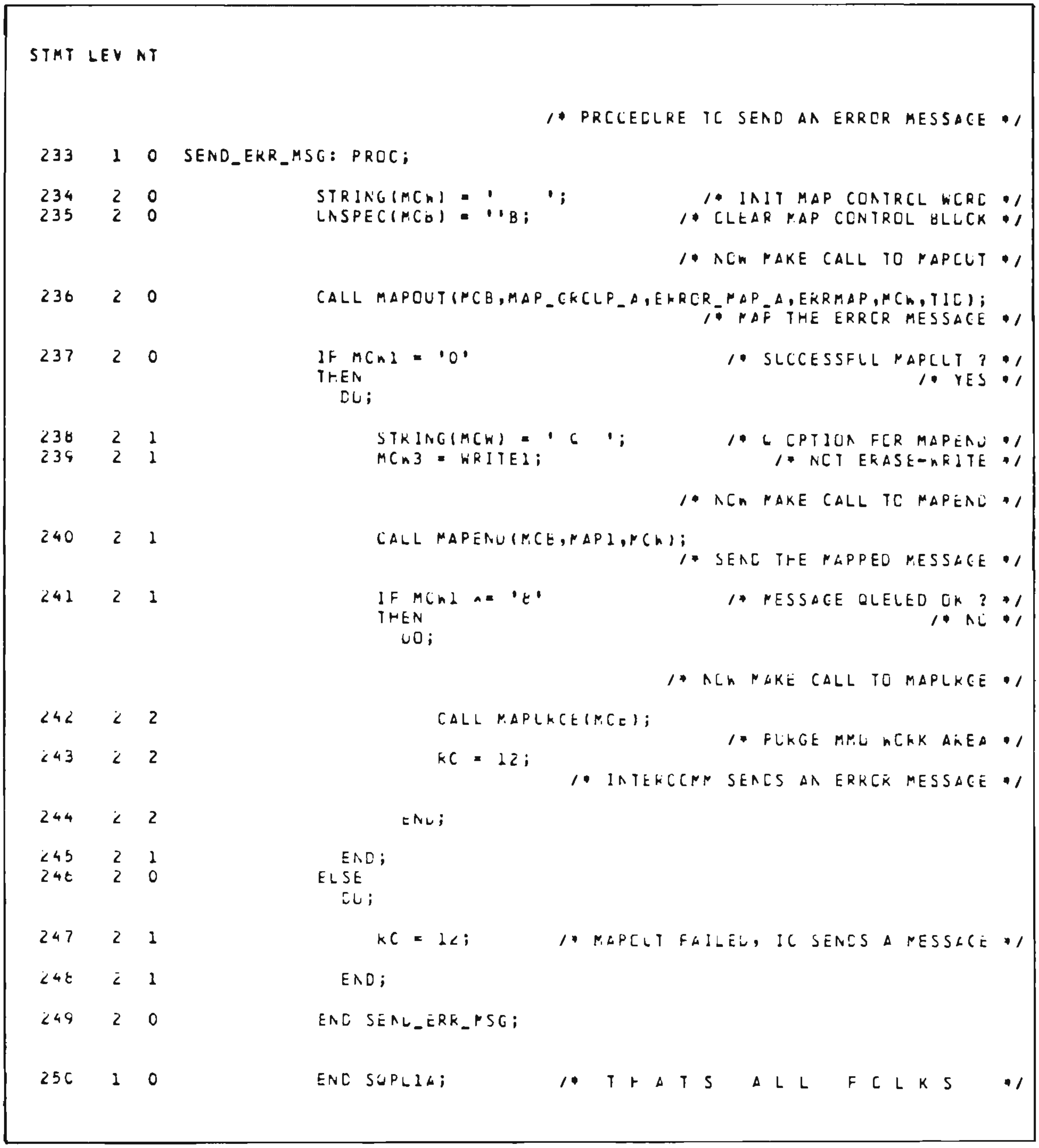
```
STMT LEV NT


                                  /* PROCEDURE TO READ THE VSAM FILE - DDNAME=STOKFILE */

 190   1  0  VSAM_READ: PROC;                 /* READ VSAM FILE BY KEY */

 191   2  0              S_REC_WHS = WHSNO;              /* WHSNO IS PART OF THE KEY */
 192   2  0              STRING(S_REC_PNO) = STRING(PARTNO);
                                                         /* PARTNO IS PART OF THE KEY */
 193   2  0              KEY_FIELD = STRING(S_REC_KEY_FIELD);      /* THE VSAM KEY */
 194   2  0              CURRENT_FILE = DD_STOCK;        /* SET FILE TO BE ACCESSED */
 195   2  0              STRING(FHCW) = '    '; /* INIT FILE HANDLER CONTROL WORD */
 196   2  0              UNSPEC(EXTDSCT) = ''B;
                                              /* INIT FILE HANDLER CONTROL BLOCK */

 197   2  0              CALL SELECT(EXTDSCT,FHCW,CURRENT_FILE);  /* SELECT FILE  */

 198   2  0              IF FHCW1 = '9'                  /* SELECT ERROR ?, NO DD */
                         THEN
                           DO;

 199   2  1                  ERROR_FLAG = 3;         /* YES - SET BAD RETURN CODE  */
 200   2  1                  RETURN;

 201   2  1                END;

 202   2  0              STRING(FHCW) = '    ';   /* SELECT OK, INIT FHCW FOR READ*/

 203   2  0              CALL GETV(EXTDSCT,FHCW,STOCK_RECORD,KEY_FIELD);
                                                               /* VSAM READ BY KEY */

 204   2  0              SELECT (FHCW1);                /* SELECT GETV RETURN CODE */

 205   2  1                WHEN('1')                                 /* I/O ERROR */
                             DO;

 206   2  2                    ERROR_FLAG = 4;
 207   2  2                    RETURN;

 208   2  2                  END;
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 15 of 18)



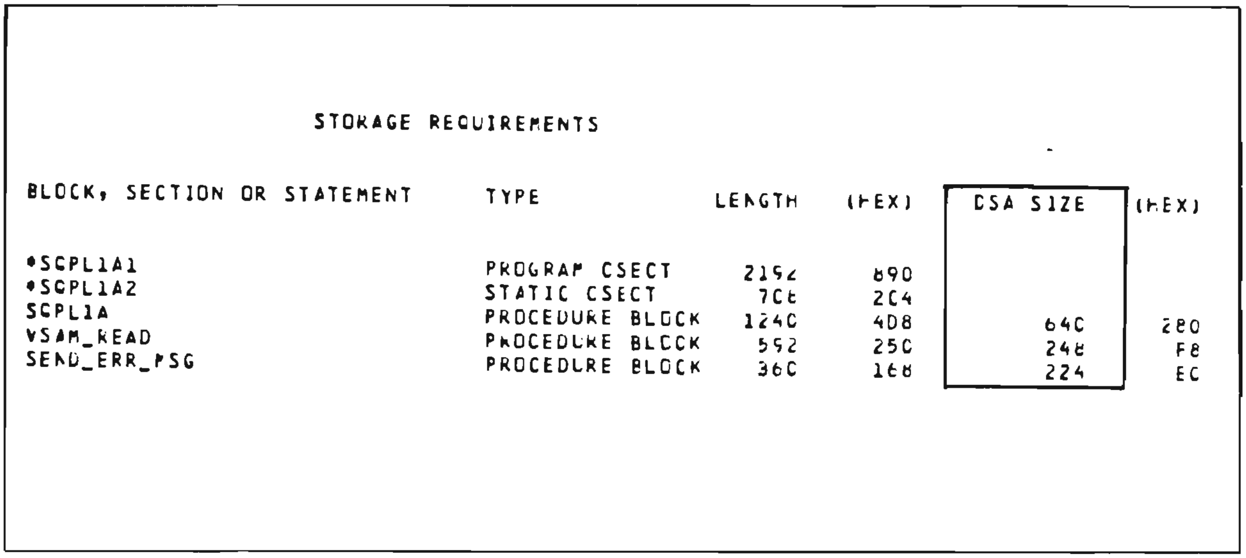
```
STMT LEV NT

 209   2  1               WHEN ('2')                         /* RECORD NOT FOUND */
                             DO;

 210   2  2                     ERROR_FLAG = 6;     /* WAREHOUSE MATCH NOT EQUAL */
 211   2  2                     RETURN;

 212   2  2                  END;
 213   2  1               WHEN ('9')                         /* INVALID FUNCTION */
                             DO;

 214   2  2                     ERROR_FLAG = 4;            /* TREAT AS I/O ERROR */
 215   2  2                     RETURN;

 216   2  2                  END;

 217   2  1               WHEN ('0')                        /* SUCCESSFUL ACCESS */
                             DO;

                           /* OBTAIN INFORMATION FROM THE STOCK RECORD JUST READ */

 218   2  2                     WHSLOC = S_REC_WLC;        /* MOVE THE LOCATION   */
 219   2  2                     STKLEV = S_REC_LEV;        /* MOVE STOCK LEVEL    */
 220   2  2                     MONTH = SUBSTR((S_REC_LDT),1,2);
                                                           /* EXTRACT THE MONTH   */
 221   2  2                     DAY = SUBSTR((S_REC_LDT),3,2);
                                                           /* EXTRACT THE DAY NO  */
 222   2  2                     YEAR = SUBSTR((S_REC_LDT),5,2);
                                                           /* EXTRACT THE YEAR    */
 223   2  2                     SLASH1, SLASH2 = '/';      /* MOVE IN THE '/'S    */
 224   2  2                     LEVDATE = STRING(DATE);    /* MOVE LEVEL DATE     */
 225   2  2                     STKORD = S_REC_ORL;        /* MOVE ORDER LEVEL    */
 226   2  2                     MONTH = SUBSTR((S_REC_ODT),1,2);
                                                           /* EXTACT THE MONTH    */
 227   2  2                     DAY = SUBSTR((S_REC_ODT),3,2);
                                                           /* EXTRACT THE DAY NO  */
 228   2  2                     YEAR = SUBSTR((S_REC_ODT),5,2);
                                                           /* EXTRACT THE YEAR    */
                                                         /* NOTE '/'S IN ALREADY */
 229   2  2                     ORDDATE = STRING(DATE);    /* MOVE STOCK DATE     */

 230   2  2                  END;

 231   2  1             END;                                    /* END OF SELECT */

 232   2  0             END VSAM_READ;
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 16 of 18)



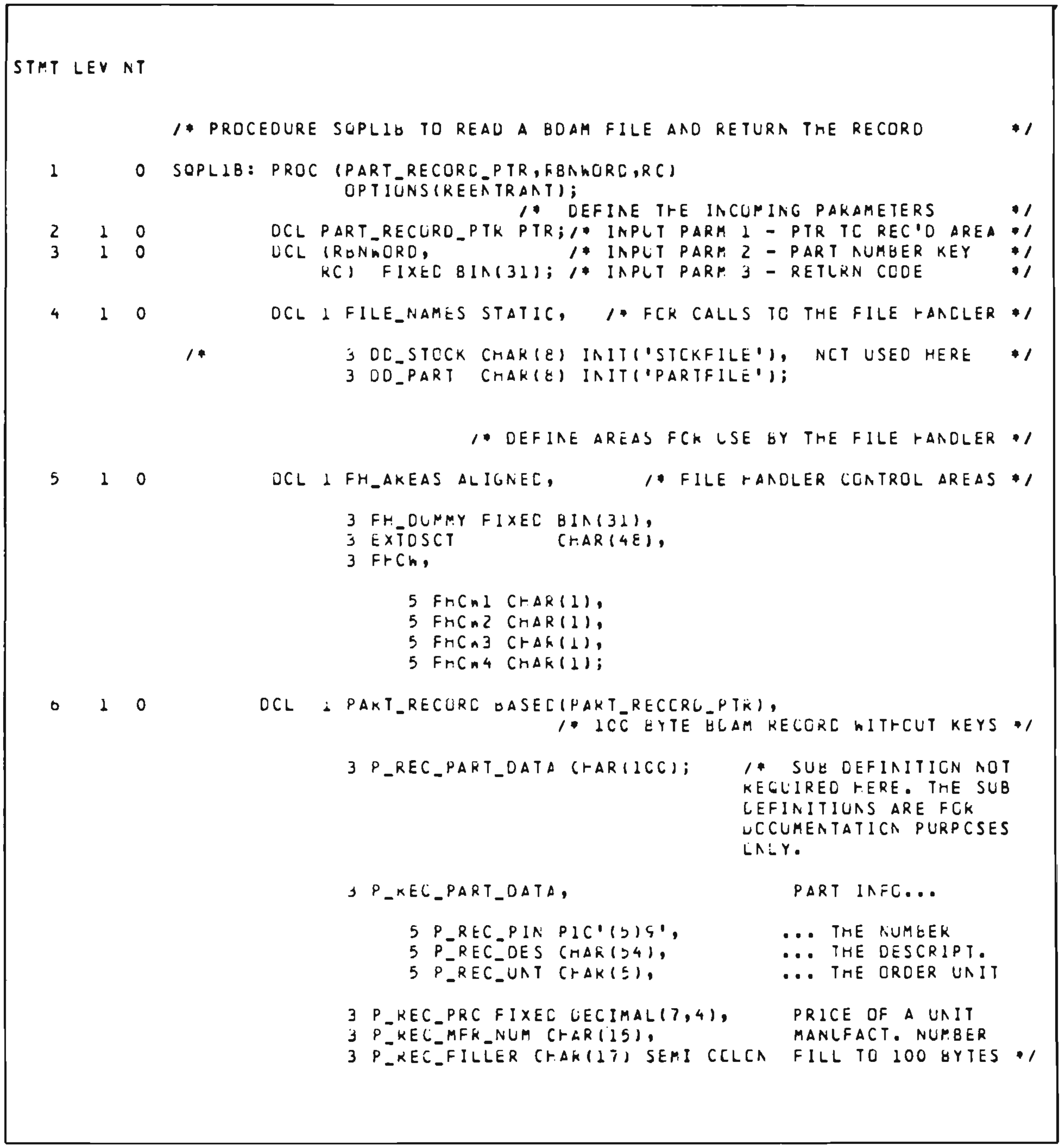
```
STMT LEV NT


                                                /* PROCEDURE TO SEND AN ERROR MESSAGE */

 233   1  0  SEND_ERR_MSG: PROC;

 234   2  0             STRING(MCW) = '    ';          /* INIT MAP CONTROL WORD */
 235   2  0             UNSPEC(MCB) = ''B;           /* CLEAR MAP CONTROL BLOCK */

                                                       /* NOW MAKE CALL TO MAPOUT */

 236   2  0             CALL MAPOUT(MCB,MAP_GROUP_A,ERROR_MAP_A,ERRMAP,MCW,TID);
                                                         /* MAP THE ERROR MESSAGE */

 237   2  0             IF MCW1 = '0'                      /* SUCCESSFUL MAPOUT ? */
                        THEN                                                /* YES */
                          DO;

 238   2  1                  STRING(MCW) = ' C  ';        /* C OPTION FOR MAPEND */
 239   2  1                  MCW3 = WRITE1;                  /* NOT ERASE-WRITE */

                                                       /* NOW MAKE CALL TO MAPEND */

 240   2  1                  CALL MAPEND(MCB,MAP1,MCW);
                                                       /* SEND THE MAPPED MESSAGE */

 241   2  1                  IF MCW1 ^= '8'              /* MESSAGE QUEUED OK ? */
                             THEN                                           /* NO */
                               DO;

                                                      /* NOW MAKE CALL TO MAPURGE */

 242   2  2                       CALL MAPURGE(MCB);
                                                         /* PURGE MMU WORK AREA */
 243   2  2                       RC = 12;
                                                /* INTERCOMM SENDS AN ERROR MESSAGE */

 244   2  2                     END;

 245   2  1                 END;
 246   2  0             ELSE
                          DO;

 247   2  1                  RC = 12;          /* MAPOUT FAILED, IC SENDS A MESSAGE */

 248   2  1                 END;

 249   2  0             END SEND_ERR_MSG;


 250   1  0             END SQPL1A;         /*   T H A T S   A L L   F O L K S     */
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 17 of 18)



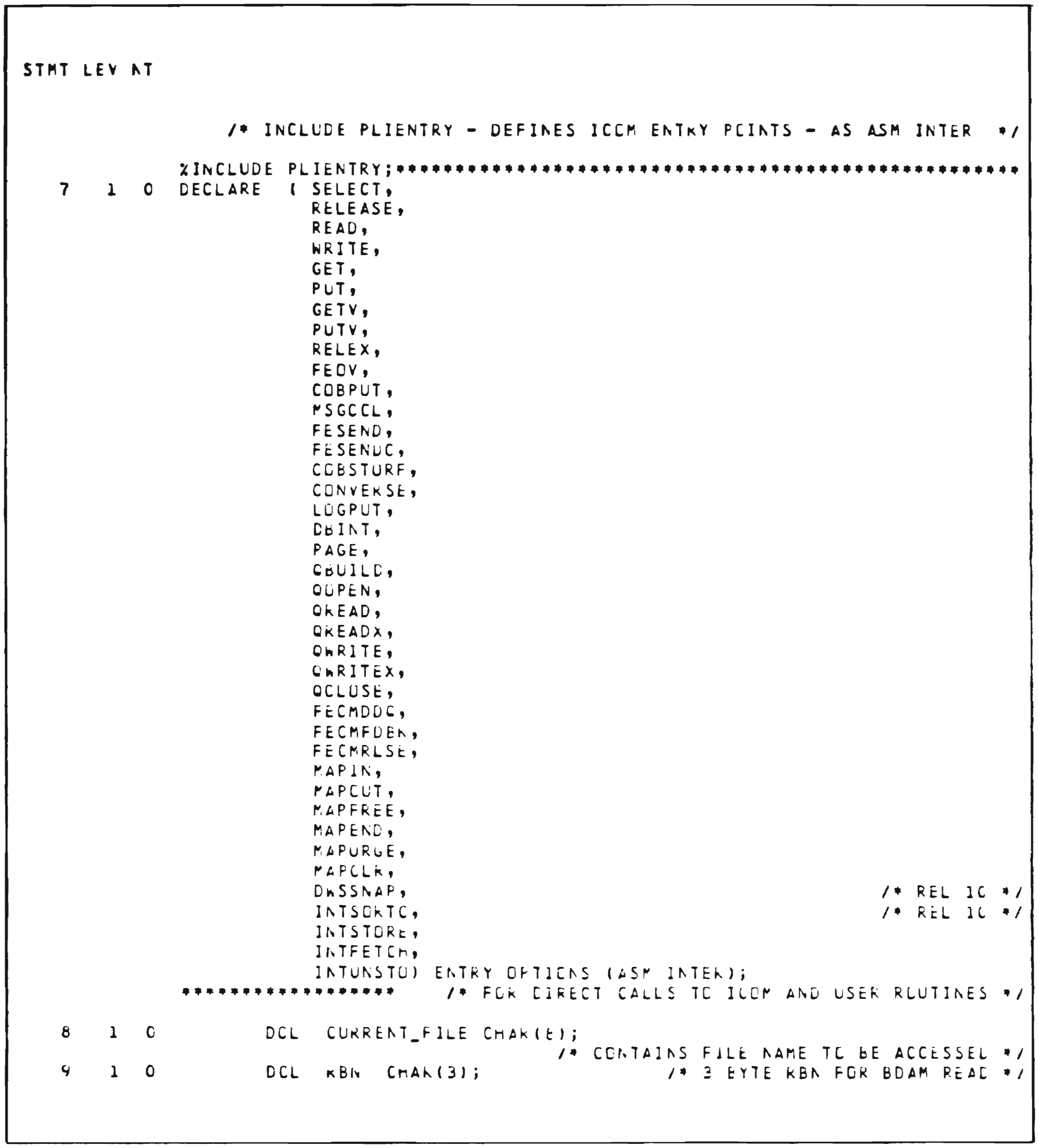
```
                    STORAGE REQUIREMENTS

BLOCK, SECTION OR STATEMENT      TYPE                LENGTH    (HEX)    DSA SIZE   (HEX)

*SQPL1A1                         PROGRAM CSECT        2192      890
*SQPL1A2                         STATIC CSECT          708      2C4
SQPL1A                           PROCEDURE BLOCK      1240      4D8          640     280
VSAM_READ                        PROCEDURE BLOCK       592      250          248      F8
SEND_ERR_MSG                     PROCEDURE BLOCK       360      168          224      E0
```

Figure 48. Sample PL/1 Subsystem SQPL1A (Page 18 of 18)



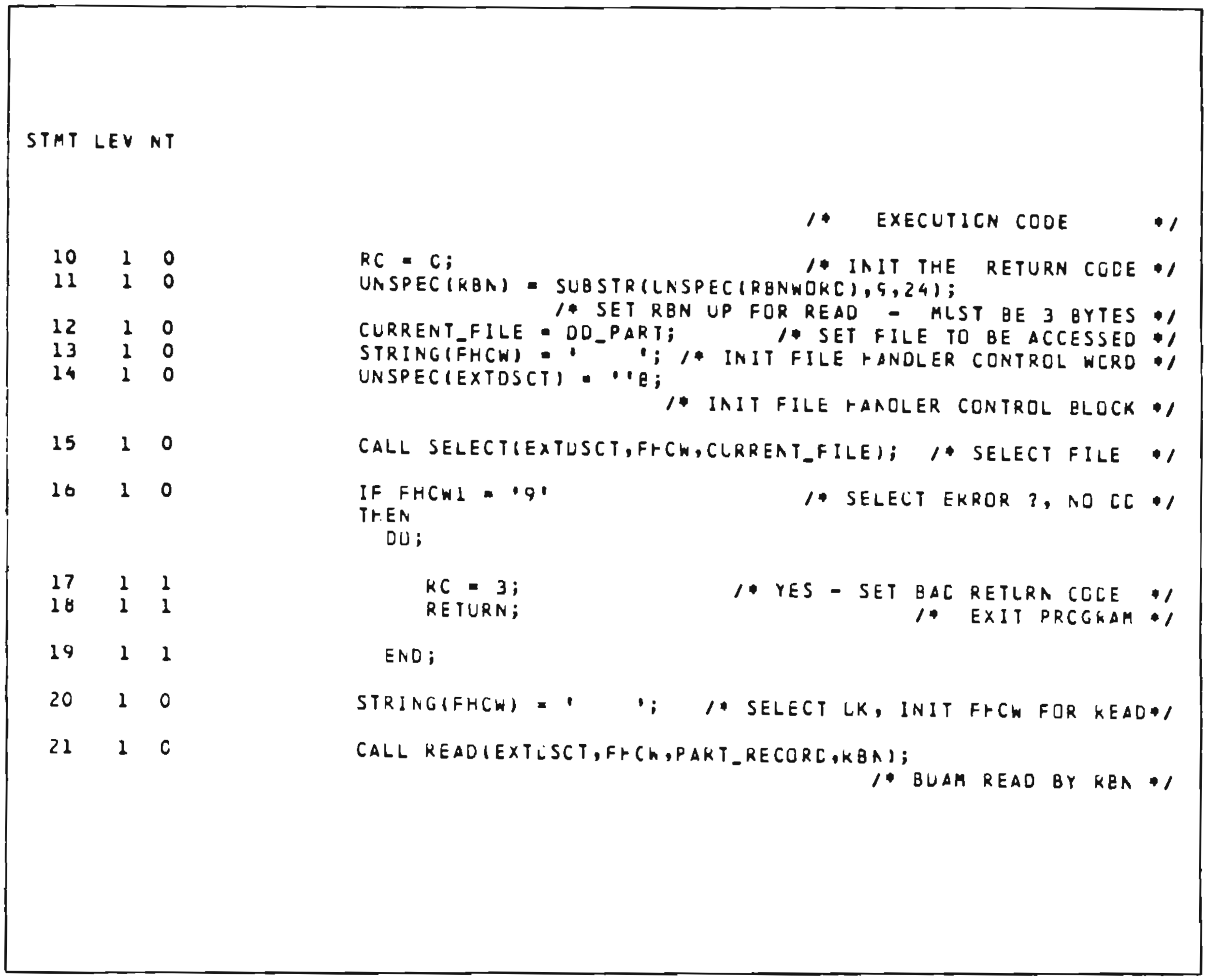
```
STMT LEV NT


                /* PROCEDURE SQPL1B TO READ A BDAM FILE AND RETURN THE RECORD       */

   1       0    SQPL1B: PROC (PART_RECORD_PTR,RBNWORD,RC)
                              OPTIONS(REENTRANT);
                                     /*  DEFINE THE INCOMING PARAMETERS            */
   2   1   0            DCL PART_RECORD_PTR PTR;/* INPUT PARM 1 - PTR TO REC'D AREA */
   3   1   0            DCL (RBNWORD,             /* INPUT PARM 2 - PART NUMBER KEY    */
                            RC)  FIXED BIN(31); /* INPUT PARM 3 - RETURN CODE        */

   4   1   0            DCL 1 FILE_NAMES STATIC,    /* FOR CALLS TO THE FILE HANDLER */

                 /*           3 DD_STOCK CHAR(8) INIT('STCKFILE'),  NOT USED HERE    */
                              3 DD_PART  CHAR(8) INIT('PARTFILE');


                                          /* DEFINE AREAS FOR USE BY THE FILE HANDLER */

   5   1   0            DCL 1 FH_AREAS ALIGNED,         /* FILE HANDLER CONTROL AREAS */

                              3 FH_DUMMY FIXED BIN(31),
                              3 EXTDSCT         CHAR(48),
                              3 FHCW,

                                  5 FHCW1 CHAR(1),
                                  5 FHCW2 CHAR(1),
                                  5 FHCW3 CHAR(1),
                                  5 FHCW4 CHAR(1);

   6   1   0           DCL  1 PART_RECORD BASED(PART_RECORD_PTR),
                                                 /* 100 BYTE BDAM RECORD WITHOUT KEYS */

                              3 P_REC_PART_DATA CHAR(100);      /*  SUB DEFINITION NOT
                                                               REQUIRED HERE. THE SUB
                                                               DEFINITIONS ARE FOR
                                                               DOCUMENTATION PURPOSES
                                                               ONLY.

                              3 P_REC_PART_DATA,                  PART INFO...

                                  5 P_REC_PIN PIC'(5)9',         ... THE NUMBER
                                  5 P_REC_DES CHAR(54),          ... THE DESCRIPT.
                                  5 P_REC_UNT CHAR(5),           ... THE ORDER UNIT

                              3 P_REC_PRC FIXED DECIMAL(7,4),     PRICE OF A UNIT
                              3 P_REC_MFR_NUM CHAR(15),           MANUFACT. NUMBER
                              3 P_REC_FILLER CHAR(17) SEMI COLON  FILL TO 100 BYTES */
```

Figure 49. Sample PL/1 Subroutine SQPL1B (Page 1 of 4)



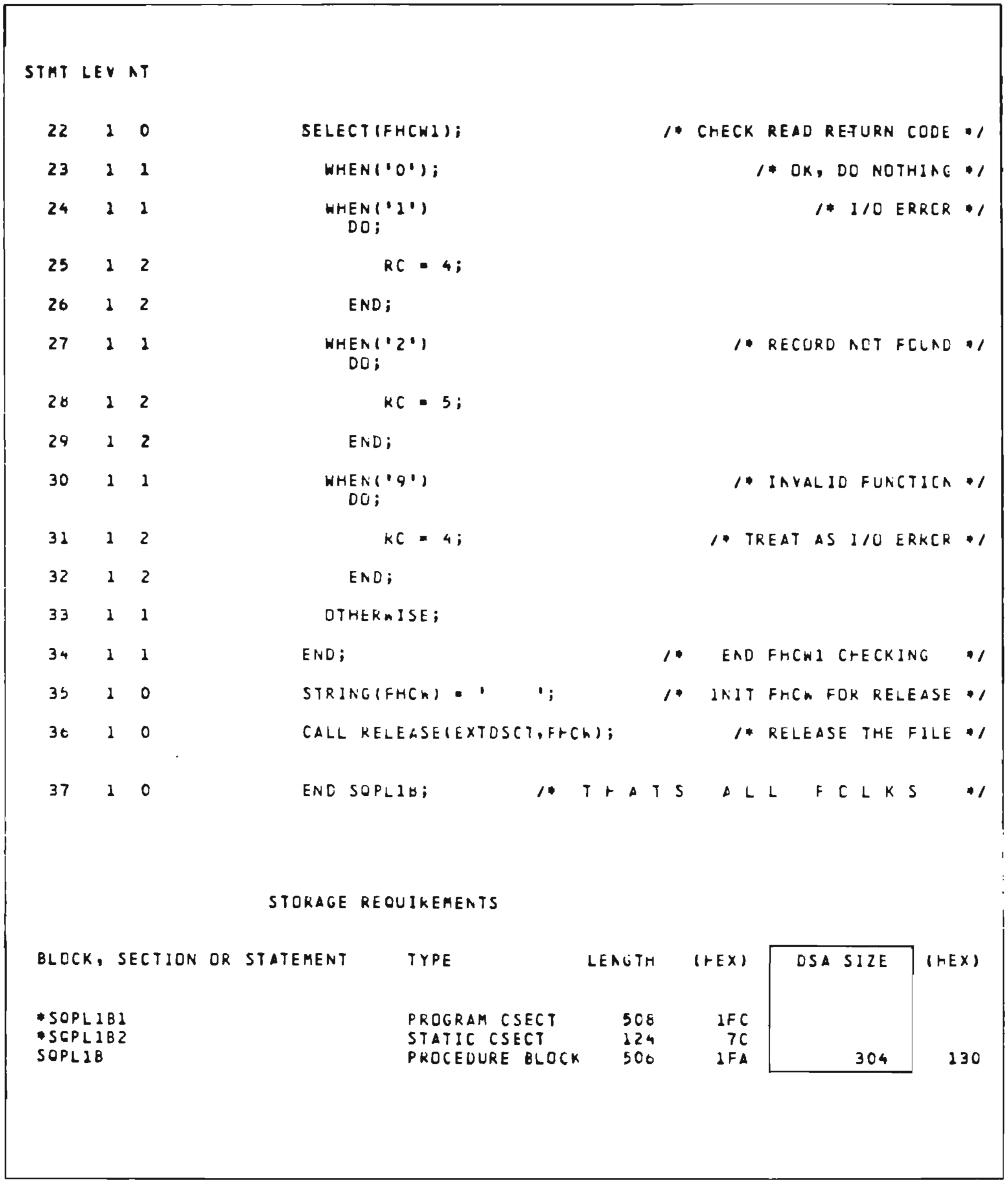
```
STMT LEV NT

                 /* INCLUDE PLIENTRY - DEFINES ICOM ENTRY POINTS - AS ASM INTER  */

             %INCLUDE PLIENTRY;****************************************************************
   7   1  0  DECLARE  ( SELECT,
                        RELEASE,
                        READ,
                        WRITE,
                        GET,
                        PUT,
                        GETV,
                        PUTV,
                        RELEX,
                        FEOV,
                        COBPUT,
                        MSGCOL,
                        FESEND,
                        FESENDC,
                        COBSTORF,
                        CONVERSE,
                        LOGPUT,
                        DBINT,
                        PAGE,
                        QBUILD,
                        QOPEN,
                        QREAD,
                        QREADX,
                        QWRITE,
                        QWRITEX,
                        QCLOSE,
                        FECMDDC,
                        FECMFDBK,
                        FECMRLSE,
                        MAPIN,
                        MAPOUT,
                        MAPFREE,
                        MAPEND,
                        MAPURGE,
                        MAPCLR,
                        DWSSNAP,                                           /* REL 10 */
                        INTSORTC,                                          /* REL 10 */
                        INTSTORE,
                        INTFETCH,
                        INTUNSTO) ENTRY OPTIONS (ASM INTER);
             ******************   /* FOR DIRECT CALLS TO ICOM AND USER ROUTINES */

   8   1  0         DCL  CURRENT_FILE CHAR(8);
                                            /* CONTAINS FILE NAME TO BE ACCESSED */
   9   1  0         DCL  RBN  CHAR(3);                /* 3 BYTE RBN FOR BDAM READ */
```

Figure 49. Sample PL/1 Subroutine SQPL1B (Page 2 of 4)



```
STMT LEV NT

                                                  /*   EXECUTION CODE       */

  10   1  0          RC = 0;                      /* INIT THE  RETURN CODE */
  11   1  0          UNSPEC(RBN) = SUBSTR(UNSPEC(RBNWORD),9,24);
                               /* SET RBN UP FOR READ  -  MUST BE 3 BYTES */
  12   1  0          CURRENT_FILE = DD_PART;        /* SET FILE TO BE ACCESSED */
  13   1  0          STRING(FHCW) = '    '; /* INIT FILE HANDLER CONTROL WORD */
  14   1  0          UNSPEC(EXTDSCT) = ''B;
                                   /* INIT FILE HANDLER CONTROL BLOCK */

  15   1  0          CALL SELECT(EXTDSCT,FHCW,CURRENT_FILE);  /* SELECT FILE   */

  16   1  0          IF FHCW1 = '9'                 /* SELECT ERROR ?, NO CC */
                     THEN
                       DO;

  17   1  1              RC = 3;               /* YES - SET BAD RETURN CODE  */
  18   1  1              RETURN;                          /*  EXIT PROGRAM */

  19   1  1            END;

  20   1  0          STRING(FHCW) = '    ';   /* SELECT OK, INIT FHCW FOR READ*/

  21   1  0          CALL READ(EXTDSCT,FHCW,PART_RECORD,RBN);
                                                      /* BDAM READ BY RBN */
```

Figure 49. Sample PL/1 Subroutine SQPL1B (Page 3 of 4)



```
STMT LEV NT

  22   1  0              SELECT(FHCW1);                 /* CHECK READ RETURN CODE */

  23   1  1                WHEN('0');                           /* OK, DO NOTHING */

  24   1  1                WHEN('1')                                 /* I/O ERROR */
                             DO;

  25   1  2                     RC = 4;

  26   1  2                  END;

  27   1  1                WHEN('2')                          /* RECORD NOT FOUND */
                             DO;

  28   1  2                     RC = 5;

  29   1  2                  END;

  30   1  1                WHEN('9')                          /* INVALID FUNCTION */
                             DO;

  31   1  2                     RC = 4;                     /* TREAT AS I/O ERROR */

  32   1  2                  END;

  33   1  1                OTHERWISE;

  34   1  1              END;                           /*   END FHCW1 CHECKING   */

  35   1  0              STRING(FHCW) = '    ';         /*  INIT FHCW FOR RELEASE */

  36   1  0              CALL RELEASE(EXTDSCT,FHCW);          /* RELEASE THE FILE */

  37   1  0              END SQPL1B;          /*  T H A T S   A L L   F O L K S   */



                         STORAGE REQUIREMENTS

  BLOCK, SECTION OR STATEMENT      TYPE            LENGTH   (HEX)   DSA SIZE   (HEX)

  *SQPL1B1                         PROGRAM CSECT     508     1FC
  *SQPL1B2                         STATIC CSECT      124      7C
  SQPL1B                           PROCEDURE BLOCK   506     1FA        304     130
```

Figure 49. Sample PL/1 Subroutine SQPL1B (Page 4 of 4)

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 Message and Codes. Additional debugging facilities such as dispatcher trace reports, thread dumps and indicative dumps are 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 PL/1 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 PL/1 subroutine and reassemble and linkedit REENTSBS which copies USRSUBS (see Figure 50). Also reassemble INTLOAD to copy the same USRSUBS if the program is loadable, uses direct calls, and is linked with INTLOAD.

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. Create the symbolic map copy member(s) to be included in the program and place them on SYMPL1 (PL/1 V1) or on the library with the program (PL/1 V2). See Message Mapping Utilities.

6. Prepare input test message data set(s) using the CREATSIM utility as illustrated in Figure 52. The first 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 PL/1 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.

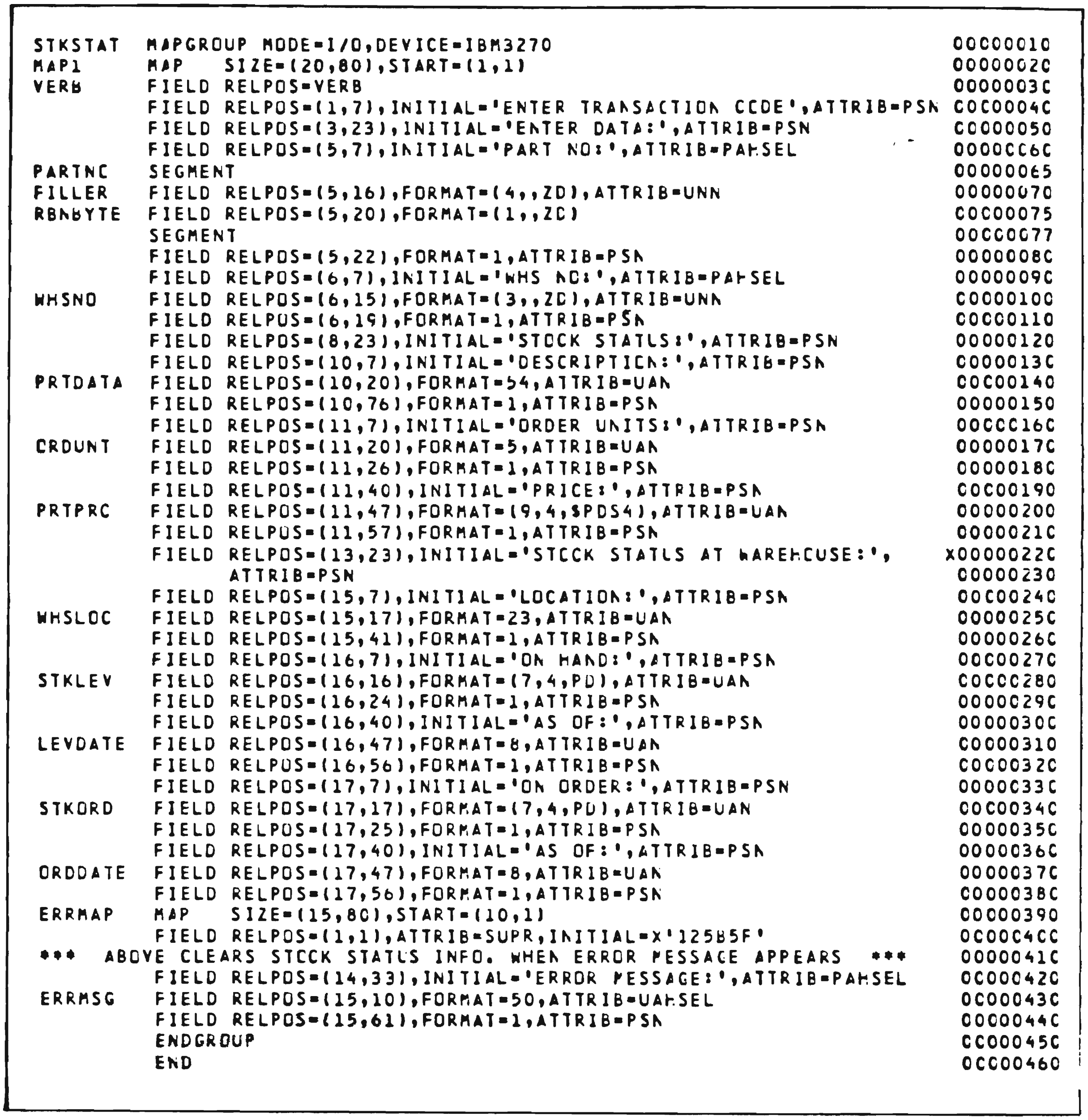
The resulting SIM3270 printouts for the simulated execution of the sample inquiry subsystem are illustrated in Figure 54. Note that the underlined positions on each screen display indicate attribute byte positions; codes are described under the display. On an actual terminal, the attribute byte position appears as a blank to the terminal operator. See <u>Message Mapping Utilities</u> 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 SUBSYTEM
//*
//STEP1       EXEC  LIBELINK,Q=TEST,NAME=INTSCT,LMOD=INTSCT
//LIB.SYSIN   DD    *
./ ADD NAME=USRSCTS
./ NUMBER     NEW1=100,INCR=100
USRSCTS       DS    0H
PQ            SYCTTBL SUBH=P,SUBC=Q,SBSP=SQPL1A,LANG=RPL1,OVLY=0,        X
                    NUMCL=10,MNCL=2,TCTV=60,SPAC=4096
/*
//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=TPL1,SSCH=P,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=SQPL1B,TYPE=PL1,DELTIME=30
/*
//ASM.SYSIN   DD    DSN=INT.SYMREL(REENTSBS),DISP=SHR
//*
//STEP4       EXEC  ASMPCL,Q=TEST,NAME=INTLOAD,LMOD=INTLOAD
//ASM.SYSIN   DD    DSN=INT.SYMREL(INTLOAD),DISP=SHR
//
```

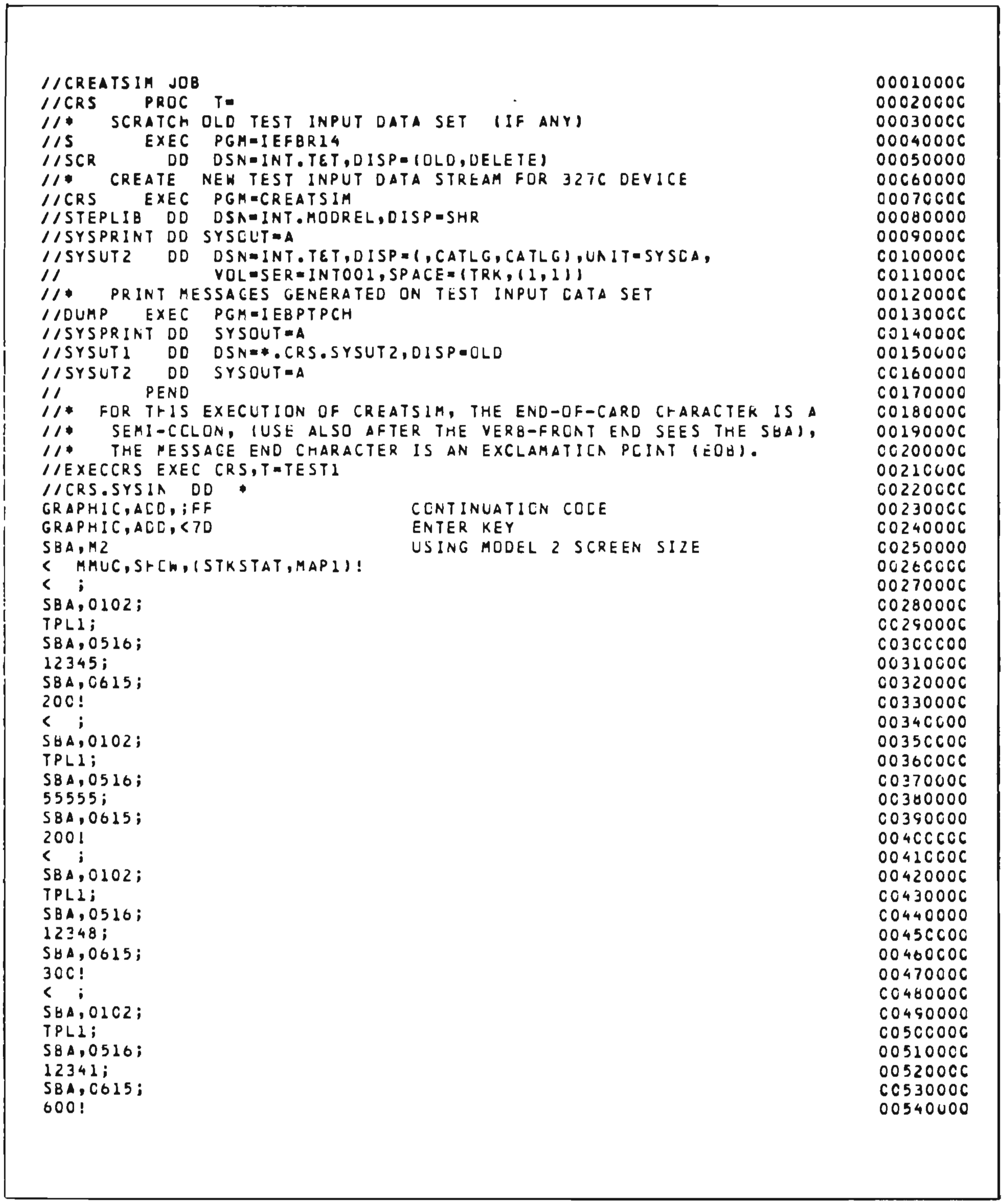
Figure 50. Table Updates to Implement Simulation Mode Testing



```
STKSTAT  MAPGROUP MODE=I/O,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=PANSEL            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=PANSEL             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,$PDS4),ATTRIB=UAN             00000200
         FIELD RELPOS=(11,57),FORMAT=1,ATTRIB=PSN                       00000210
         FIELD RELPOS=(13,23),INITIAL='STOCK STATUS AT WAREHOUSE:',    X00000220
               ATTRIB=PSN                                               00000230
         FIELD RELPOS=(15,7),INITIAL='LOCATION:',ATTRIB=PSN             00000240
WHSLOC   FIELD RELPOS=(15,17),FORMAT=23,ATTRIB=UAN                      00000250
         FIELD RELPOS=(15,41),FORMAT=1,ATTRIB=PSN                       00000260
         FIELD RELPOS=(16,7),INITIAL='ON HAND:',ATTRIB=PSN              00000270
STKLEV   FIELD RELPOS=(16,16),FORMAT=(7,4,PD),ATTRIB=UAN                00000280
         FIELD RELPOS=(16,24),FORMAT=1,ATTRIB=PSN                       00000290
         FIELD RELPOS=(16,40),INITIAL='AS OF:',ATTRIB=PSN               00000300
LEVDATE  FIELD RELPOS=(16,47),FORMAT=8,ATTRIB=UAN                       00000310
         FIELD RELPOS=(16,56),FORMAT=1,ATTRIB=PSN                       00000320
         FIELD RELPOS=(17,7),INITIAL='ON ORDER:',ATTRIB=PSN             00000330
STKORD   FIELD RELPOS=(17,17),FORMAT=(7,4,PD),ATTRIB=UAN                00000340
         FIELD RELPOS=(17,25),FORMAT=1,ATTRIB=PSN                       00000350
         FIELD RELPOS=(17,40),INITIAL='AS OF:',ATTRIB=PSN               00000360
ORDDATE  FIELD RELPOS=(17,47),FORMAT=8,ATTRIB=UAN                       00000370
         FIELD RELPOS=(17,56),FORMAT=1,ATTRIB=PSN                       00000380
ERRMAP   MAP   SIZE=(15,80),START=(10,1)                                00000390
         FIELD RELPOS=(1,1),ATTRIB=SUPR,INITIAL=X'125B5F'               00000400
***  ABOVE CLEARS STOCK STATUS INFO. WHEN ERROR MESSAGE APPEARS  ***    00000410
         FIELD RELPOS=(14,33),INITIAL='ERROR MESSAGE:',ATTRIB=PANSEL    00000420
ERRMSG   FIELD RELPOS=(15,10),FORMAT=50,ATTRIB=UANSEL                   00000430
         FIELD RELPOS=(15,61),FORMAT=1,ATTRIB=PSN                       00000440
         ENDGROUP                                                       00000450
         END                                                            00000460
```

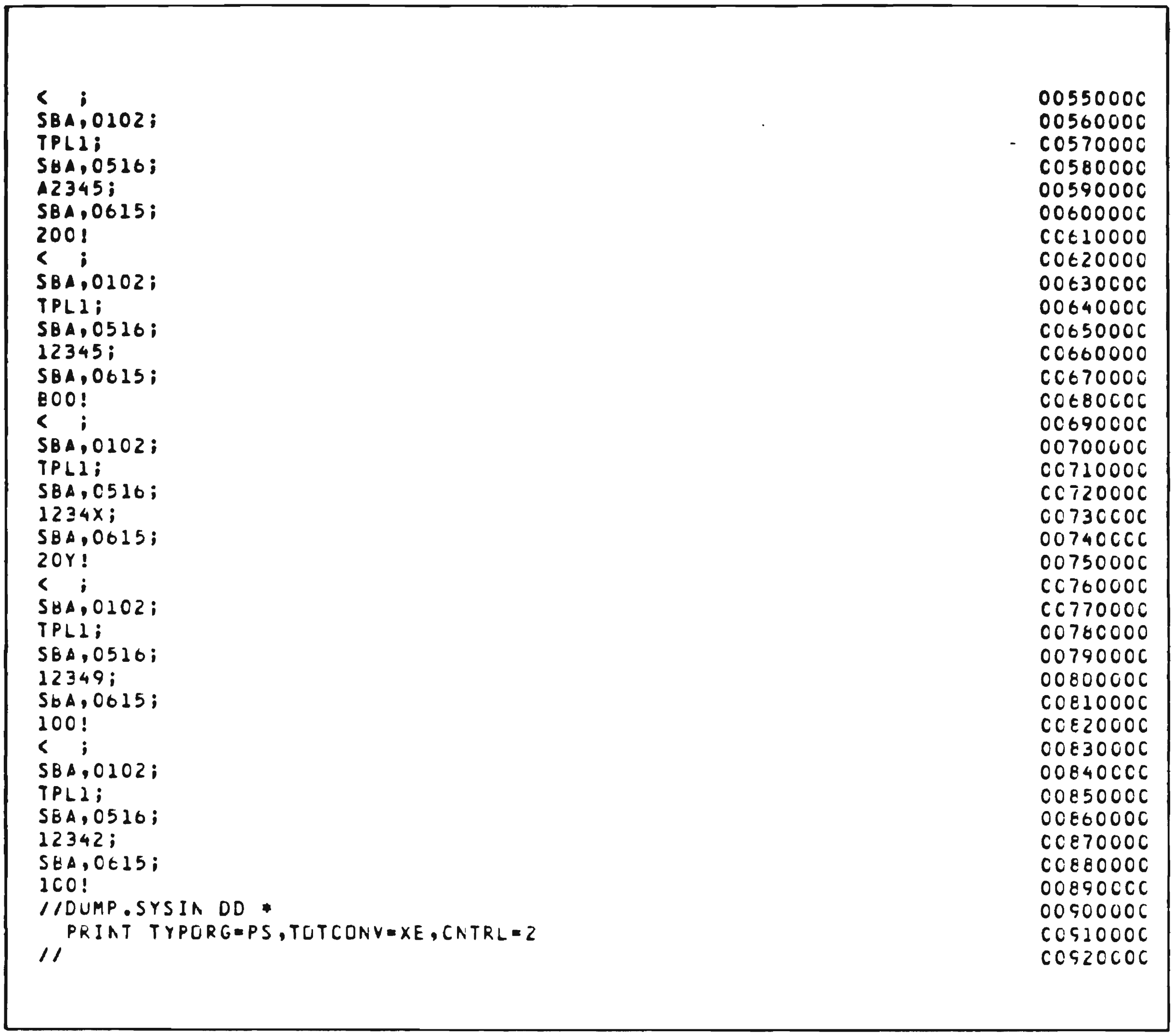
Figure 51. MMU Maps Used by Sample Subsystem

NOTE: the PL/1-oriented parameter BASED is not coded on the MAP macro because the default is YES (map name declared as BASED on PTR_mapname).



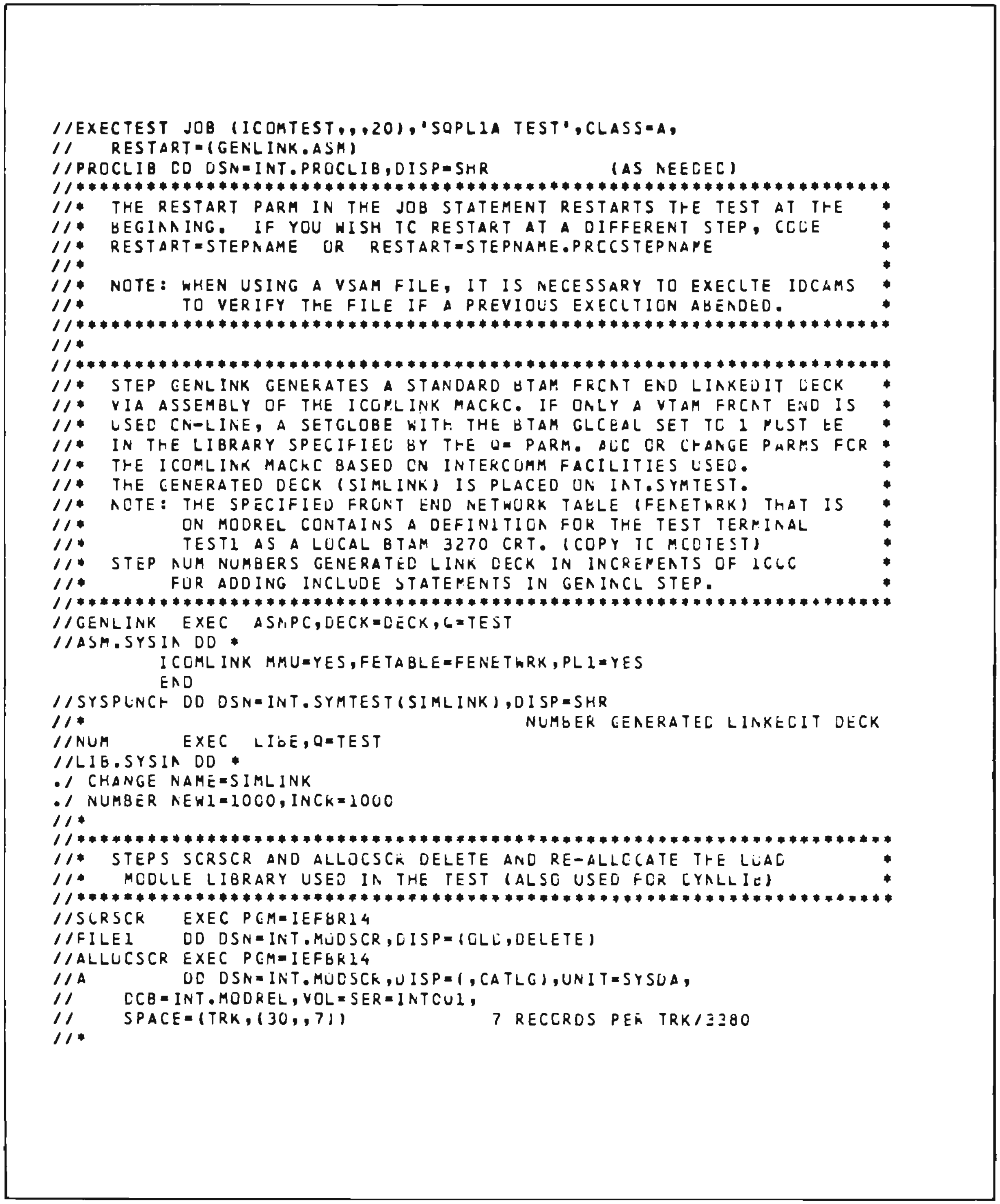
```
//CREATSIM JOB                                                         0001000C
//CRS     PROC  T=                                                     00020C0C
//*   SCRATCH OLD TEST INPUT DATA SET  (IF ANY)                        000300CC
//S       EXEC  PGM=IEFBR14                                            0004000C
//SCR       DD  DSN=INT.TЄT,DISP=(OLD,DELETE)                          00050000
//*   CREATE  NEW TEST INPUT DATA STREAM FOR 327C DEVICE               00C60000
//CRS     EXEC  PGM=CREATSIM                                           0007CC0C
//STEPLIB  DD  DSN=INT.MODREL,DISP=SHR                                 00080000
//SYSPRINT DD SYSCUT=A                                                 0009000C
//SYSUT2    DD  DSN=INT.TЄT,DISP=(,CATLG,CATLG),UNIT=SYSCA,            C010000C
//              VOL=SER=INT001,SPACE=(TRK,(1,1))                       C011000C
//*   PRINT MESSAGES GENERATED ON TEST INPUT CATA SET                  0012000C
//DUMP    EXEC  PGM=IEBPTPCH                                           001300CC
//SYSPRINT DD  SYSOUT=A                                                C014000C
//SYSUT1    DD  DSN=*.CRS.SYSUT2,DISP=OLD                              0015000C
//SYSUT2    DD  SYSOUT=A                                               CC160000
//        PEND                                                         C0170000
//*  FOR THIS EXECUTION OF CREATSIM, THE END-OF-CARD CHARACTER IS A    C018000C
//*   SEMI-CCLON, (USE ALSO AFTER THE VERB-FRONT END SEES THE SBA),    0019000C
//*   THE MESSAGE END CHARACTER IS AN EXCLAMATICN PCINT (EOB).         CC20000C
//EXECCRS EXEC CRS,T=TEST1                                             0021C00C
//CRS.SYSIN  DD  *                                                     C0220CCC
GRAPHIC,ACD,;FF                  CONTINUATICN CODE                     002300CC
GRAPHIC,ADD,<7D                  ENTER KEY                             C024000C
SBA,M2                           USING MODEL 2 SCREEN SIZE             C0250000
<  MMUC,SHCW,(STKSTAT,MAP1)!                                           0026C0CC
<  ;                                                                   0027000C
SBA,0102;                                                              C028000C
TPL1;                                                                  CC29000C
SBA,0516;                                                              C03CCC00
12345;                                                                 0031000C
SBA,C615;                                                              C032000C
20C!                                                                   C033000C
<  ;                                                                   0034C000
SBA,0102;                                                              0035CC0C
TPL1;                                                                  00360C0C
SBA,0516;                                                              C037000C
55555;                                                                 00380000
SBA,0615;                                                              C0390000
200!                                                                   004CCCCC
<  ;                                                                   0041CC0C
SBA,0102;                                                              0042000C
TPL1;                                                                  C043000C
SBA,0516;                                                              C0440000
12348;                                                                 0045CC00
SBA,0615;                                                              00460C0C
30C!                                                                   0047000C
<  ;                                                                   C048000C
SBA,01C2;                                                              C0490000
TPL1;                                                                  C050C00C
SBA,0516;                                                              0051000C
12341;                                                                 005200CC
SBA,C615;                                                              CC53000C
600!                                                                   00540000
```

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



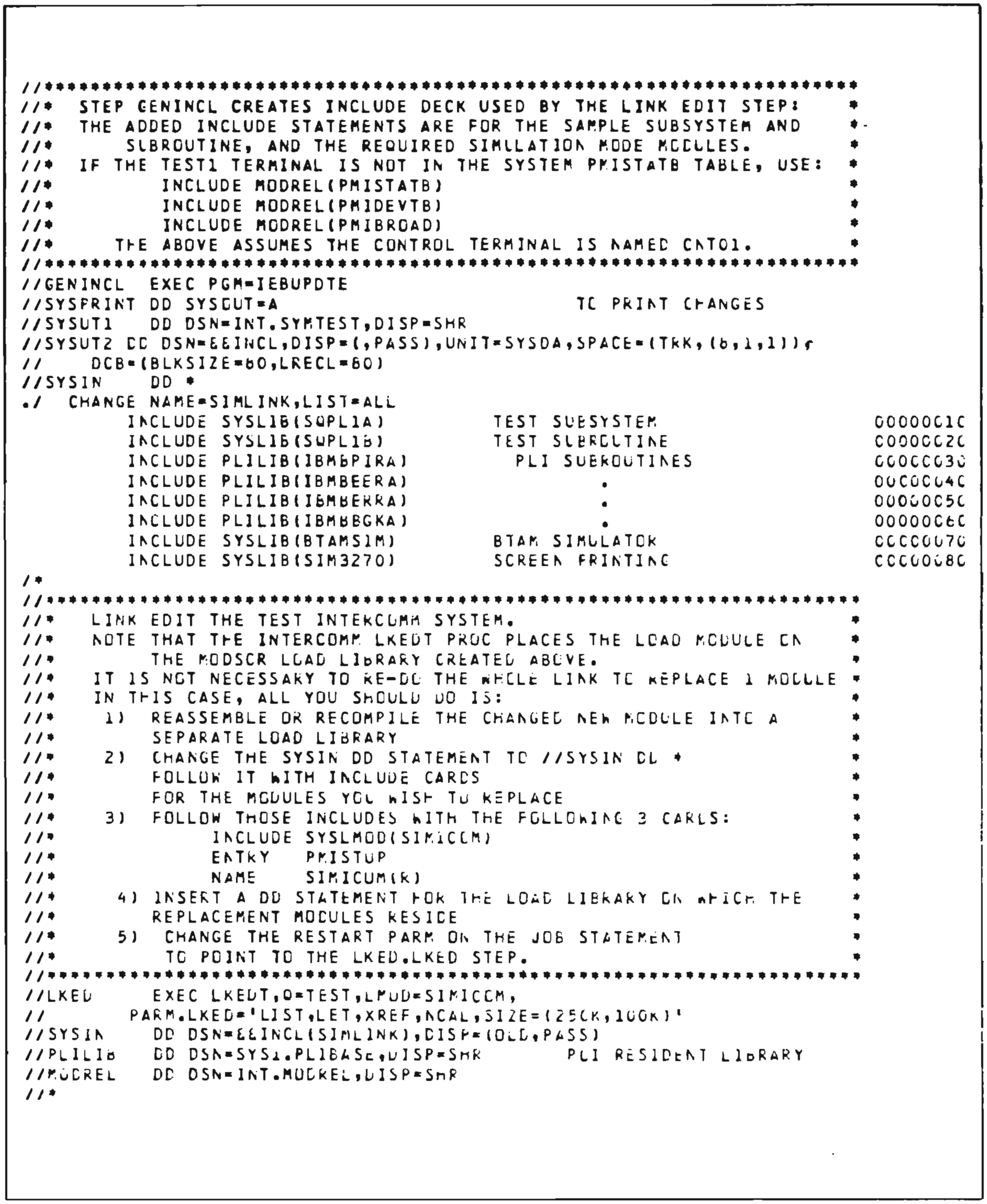
```
<  ;                                                                    00550000
SBA,0102;                                                               00560000
TPL1;                                                                   00570000
SBA,0516;                                                               00580000
A2345;                                                                  00590000
SBA,0615;                                                               00600000
200!                                                                    00610000
<  ;                                                                    00620000
SBA,0102;                                                               00630000
TPL1;                                                                   00640000
SBA,0516;                                                               00650000
12345;                                                                  00660000
SBA,0615;                                                               00670000
800!                                                                    00680000
<  ;                                                                    00690000
SBA,0102;                                                               00700000
TPL1;                                                                   00710000
SBA,0516;                                                               00720000
1234X;                                                                  00730000
SBA,0615;                                                               00740000
20Y!                                                                    00750000
<  ;                                                                    00760000
SBA,0102;                                                               00770000
TPL1;                                                                   00780000
SBA,0516;                                                               00790000
12349;                                                                  00800000
SBA,0615;                                                               00810000
100!                                                                    00820000
<  ;                                                                    00830000
SBA,0102;                                                               00840000
TPL1;                                                                   00850000
SBA,0516;                                                               00860000
12342;                                                                  00870000
SBA,0615;                                                               00880000
100!                                                                    00890000
//DUMP.SYSIN DD *                                                       00900000
  PRINT TYPORG=PS,TOTCONV=XE,CNTRL=2                                    00910000
//                                                                      00920000
```

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



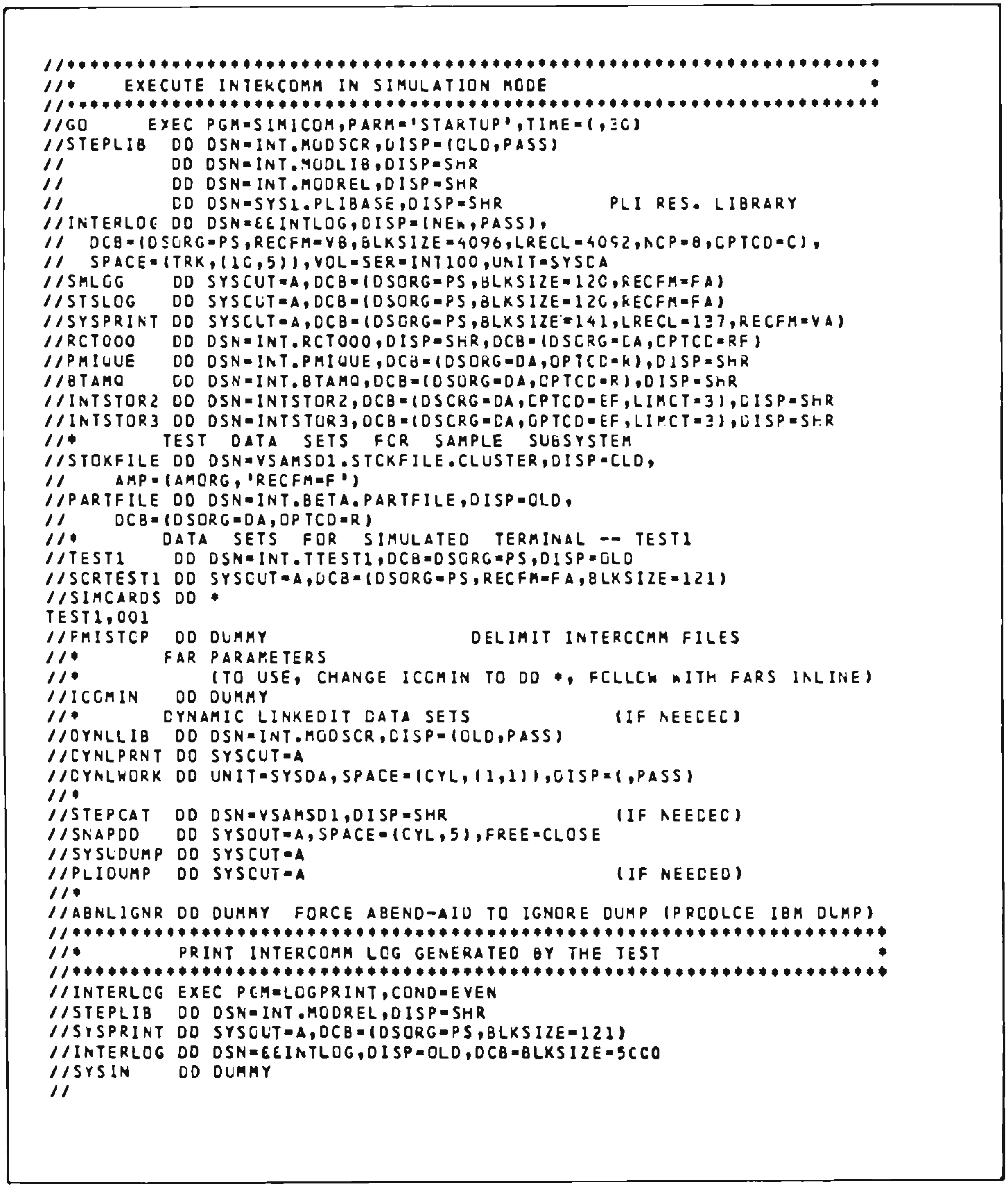
```
//EXECTEST JOB (ICOMTEST,,,20),'SQPL1A TEST',CLASS=A,
//   RESTART=(GENLINK.ASM)
//PROCLIB DD DSN=INT.PROCLIB,DISP=SHR           (AS NEEDED)
//*******************************************************************
//*  THE RESTART PARM IN THE JOB STATEMENT RESTARTS THE TEST AT THE   *
//*  BEGINNING.  IF YOU WISH TO RESTART AT A DIFFERENT STEP, CODE     *
//*  RESTART=STEPNAME  OR  RESTART=STEPNAME.PROCSTEPNAME              *
//*                                                                   *
//*  NOTE: WHEN USING A VSAM FILE, IT IS 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,PL1=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=INTC01,
//    SPACE=(TRK,(30,,7))            7 RECORDS PER TRK/3380
//*
```

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



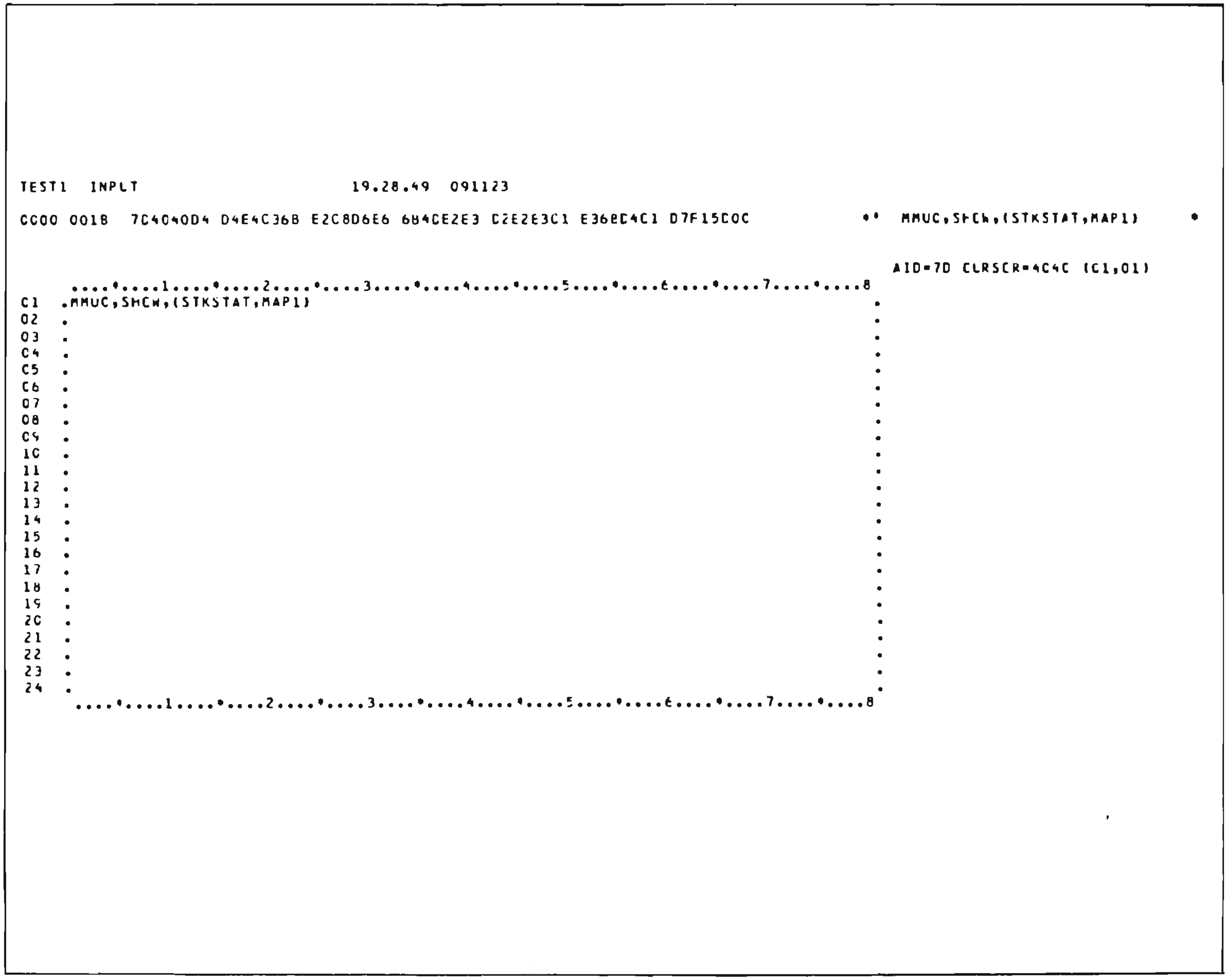
```
//**********************************************************************
//*  STEP GENINCL CREATES INCLUDE DECK USED BY THE LINK EDIT STEP:     *
//*  THE ADDED INCLUDE STATEMENTS ARE FOR THE SAMPLE SUBSYSTEM AND     *
//*      SUBROUTINE, AND THE REQUIRED SIMULATION MODE MODULES.         *
//*  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,(6,1,1)),
//     DCB=(BLKSIZE=80,LRECL=80)
//SYSIN    DD *
./  CHANGE NAME=SIMLINK,LIST=ALL
         INCLUDE SYSLIB(SQPL1A)        TEST SUBSYSTEM                   00000010
         INCLUDE SYSLIB(SQPL1B)        TEST SUBROUTINE                  00000020
         INCLUDE PLILIB(IBMBPIRA)        PLI SUBROUTINES                00000030
         INCLUDE PLILIB(IBMBEERA)                .                      00000040
         INCLUDE PLILIB(IBMBERRA)                .                      00000050
         INCLUDE PLILIB(IBMBBGKA)                .                      00000060
         INCLUDE SYSLIB(BTAMSIM)       BTAM SIMULATOR                   00000070
         INCLUDE SYSLIB(SIM3270)       SCREEN PRINTING                  00000080
/*
//**********************************************************************
//*   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)
//PLILIB    DD DSN=SYS1.PLIBASE,DISP=SHR          PLI RESIDENT LIBRARY
//MODREL    DD DSN=INT.MODREL,DISP=SHR
//*
```

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



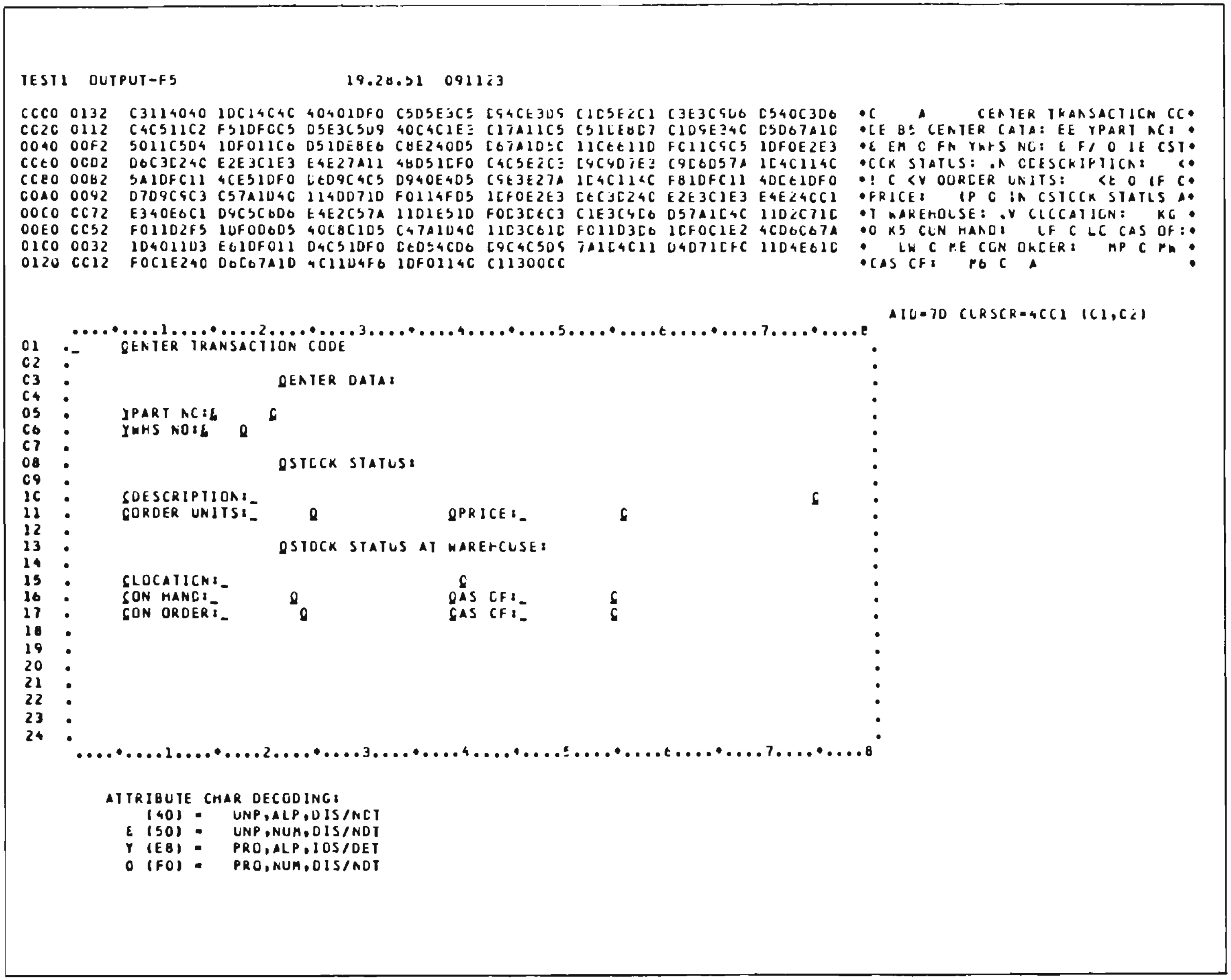
```
//**********************************************************************
//*    EXECUTE INTERCOMM IN SIMULATION MODE                            *
//**********************************************************************
//GO     EXEC PGM=SIMICOM,PARM='STARTUP',TIME=(,30)
//STEPLIB  DD DSN=INT.MODSCR,DISP=(OLD,PASS)
//         DD DSN=INT.MODLIB,DISP=SHR
//         DD DSN=INT.MODREL,DISP=SHR
//         DD DSN=SYS1.PLIBASE,DISP=SHR         PLI RES. 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.STOKFILE.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 *
TEST1,001
//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)
//*
//STEPCAT  DD DSN=VSAMSD1,DISP=SHR               (IF NEEDED)
//SNAPDD   DD SYSOUT=A,SPACE=(CYL,5),FREE=CLOSE
//SYSUDUMP DD SYSOUT=A
//PLIDUMP  DD SYSOUT=A                           (IF NEEDED)
//*
//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 53. Linkedit and Execution JCL for Simulation Mode (Page 3 of 3)



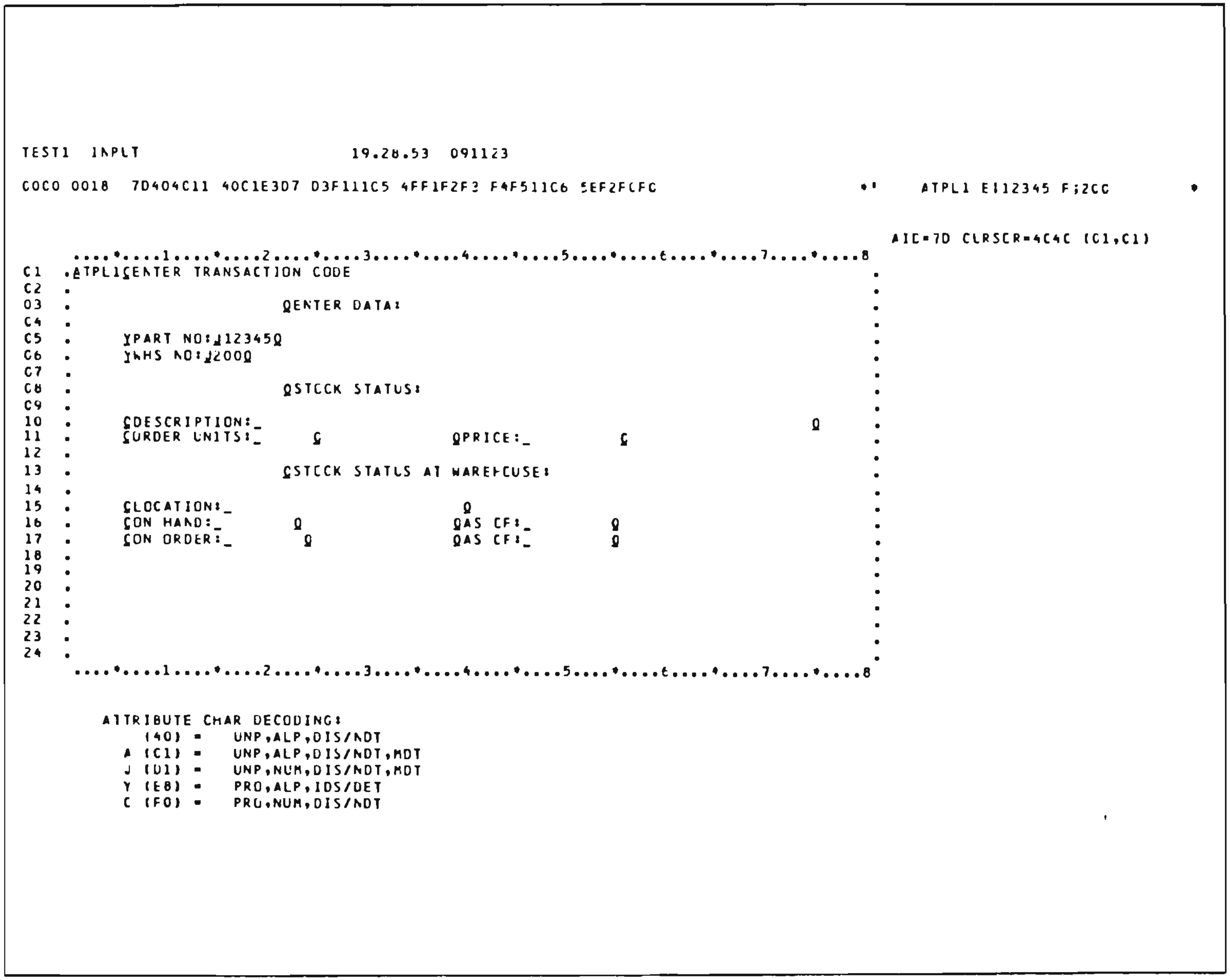
```
TEST1  INPLT                    19.28.49  091123

0000 0018  7C4040D4 D4E4C36B E2C8D6E6 6B4DE2E3 C2E2E3C1 E36BD4C1 D7F15D0C          •'  MMUC,SHCW,(STKSTAT,MAP1)      •

                                                                                     AID=7D CLRSCR=4C4C (01,01)
    ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
01  .MMUC,SHCW,(STKSTAT,MAP1)                                                       .
02  .                                                                               .
03  .                                                                               .
04  .                                                                               .
05  .                                                                               .
06  .                                                                               .
07  .                                                                               .
08  .                                                                               .
09  .                                                                               .
10  .                                                                               .
11  .                                                                               .
12  .                                                                               .
13  .                                                                               .
14  .                                                                               .
15  .                                                                               .
16  .                                                                               .
17  .                                                                               .
18  .                                                                               .
19  .                                                                               .
20  .                                                                               .
21  .                                                                               .
22  .                                                                               .
23  .                                                                               .
24  .                                                                               .
    ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
```

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



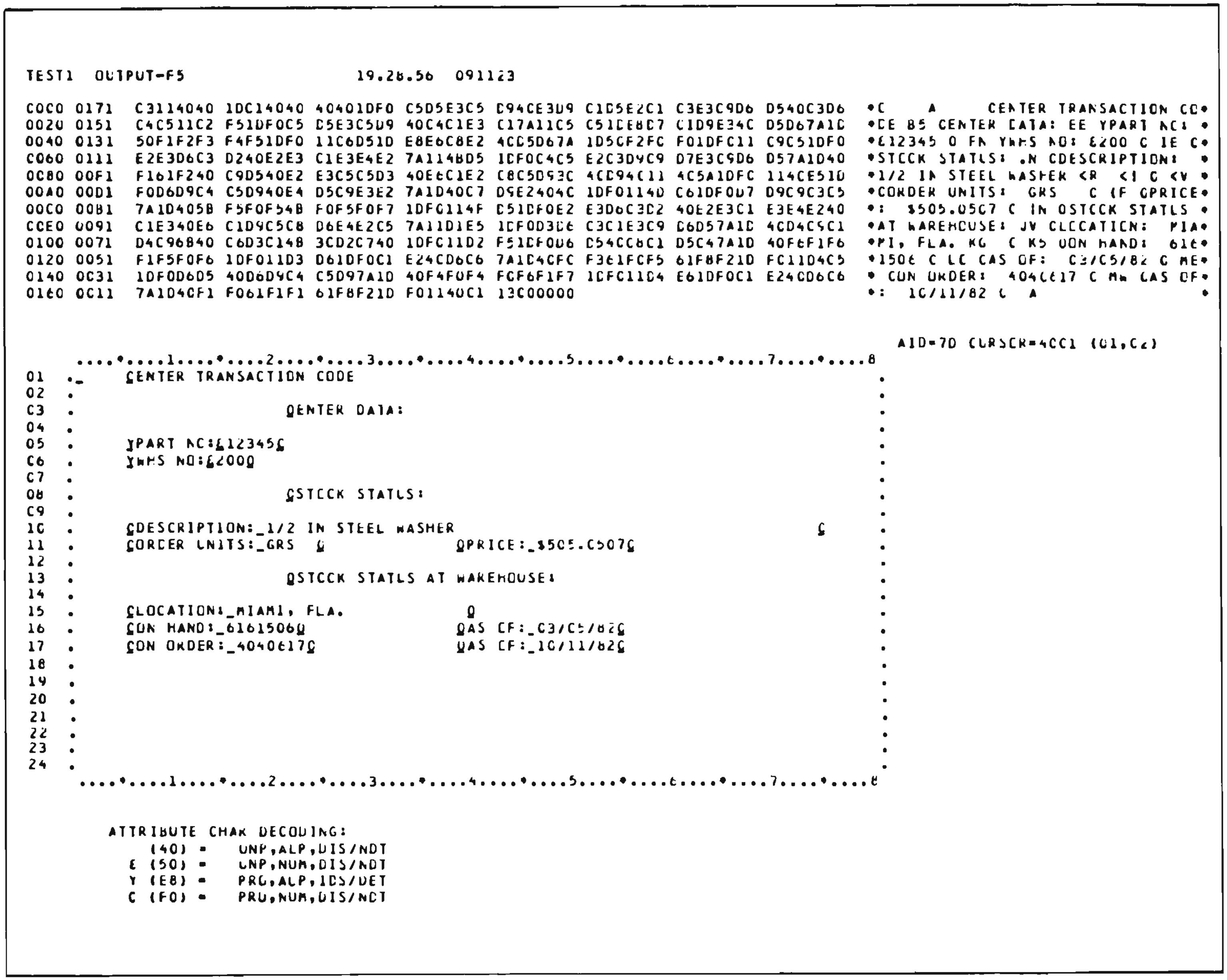
```
TEST1  OUTPUT-F5                    19.28.51  091123

CCCO 0132  C3114040 1DC14C4C 40401DFO C5D5E3C5 C94CE3D9 C1C5E2C1 C3E3C9D6 D540C3D6  •C     A      CENTER TRANSACTICN CC•
CC2C 0112  C4C511C2 F51DFCC5 D5E3C5D9 40C4C1E3 C17A11C5 C51CE8D7 C1D9E34C D5D67A1C  •CE B5 CENTER CATA: EE YPART NC: •
0040 00F2  5011C5D4 1DF011C6 D51DE8E6 C8E240D5 C67A1D5C 11C6611D FC11C9C5 1DF0E2E3  •& EM C FN YWHS NC: E F/ O 1E CST•
CC60 0CD2  D6C3C24C E2E3C1E3 E4E27A11 4BD51CF0 C4C5E2C3 C9C9D7E3 C9C6D57A 1C4C114C  •CCK STATUS: .N CDESCRIPTICN:   <•
CC80 00B2  5A1DFC11 4CE51DF0 C6D9C4C5 D940E4D5 C9E3E27A 1C4C114C F81DFC11 4DC61DF0  •! C <V OORDER UNITS:   <E 0 (F C•
C0A0 0092  D7D9C9C3 C57A1D4C 114DD71D F0114FD5 1CF0E2E3 C6C3C24C E2E3C1E3 E4E24CC1  •PRICE:   (P C IN CSTCCK STATUS A•
00C0 CC72  E340E6C1 D9C5C6D6 E4E2C57A 11D1E51D F0C3C6C3 C1E3C9C6 D57A1C4C 11D2C71C  •T WAREHOUSE: .V CLCCATICN:   KG •
00E0 CC52  F011D2F5 1DF0D6D5 40C8C1D5 C47A1D4C 11C3C61C FC11D3C6 1CF0C1E2 4CD6C67A  •0 K5 CCN HAND:   LF C LC CAS OF:•
01C0 0032  1D4011D3 E61DF011 D4C51DF0 C6D54CD6 C9C4C5D9 7A1C4C11 D4D71CFC 11D4E61C  •   LW C ME CON ORCER:   MP C PW •
0120 CC12  F0C1E240 D6C67A1D 4C11D4F6 1DF0114C C11300CC                             •CAS CF:   P6 C  A               •

                                                                                     AID=7D CURSCR=4CC1 (C1,C2)
       ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
01    ._    CENTER TRANSACTION CODE                                                  .
02    .                                                                              .
03    .                       DENTER DATA:                                           .
04    .                                                                              .
05    .     YPART NC:&      C                                                        .
06    .     YWHS NO:&    Q                                                           .
07    .                                                                              .
08    .                       QSTCCK STATUS:                                         .
09    .                                                                              .
10    .     CDESCRIPTION:_                                                   C       .
11    .     QORDER UNITS:_     Q             QPRICE:_         C                      .
12    .                                                                              .
13    .                       QSTOCK STATUS AT WAREHOUSE:                            .
14    .                                                                              .
15    .     CLOCATICN:_                     C                                        .
16    .     CON HAND:_        Q              QAS CF:_         C                      .
17    .     CON ORDER:_        Q             CAS CF:_         C                      .
18    .                                                                              .
19    .                                                                              .
20    .                                                                              .
21    .                                                                              .
22    .                                                                              .
23    .                                                                              .
24    .                                                                              .
       ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8

         ATTRIBUTE CHAR DECODING:
              (40) =    UNP,ALP,DIS/NCT
            E (50) =    UNP,NUM,DIS/NDT
            Y (E8) =    PRO,ALP,IDS/DET
            0 (F0) =    PRO,NUM,DIS/NDT
```

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



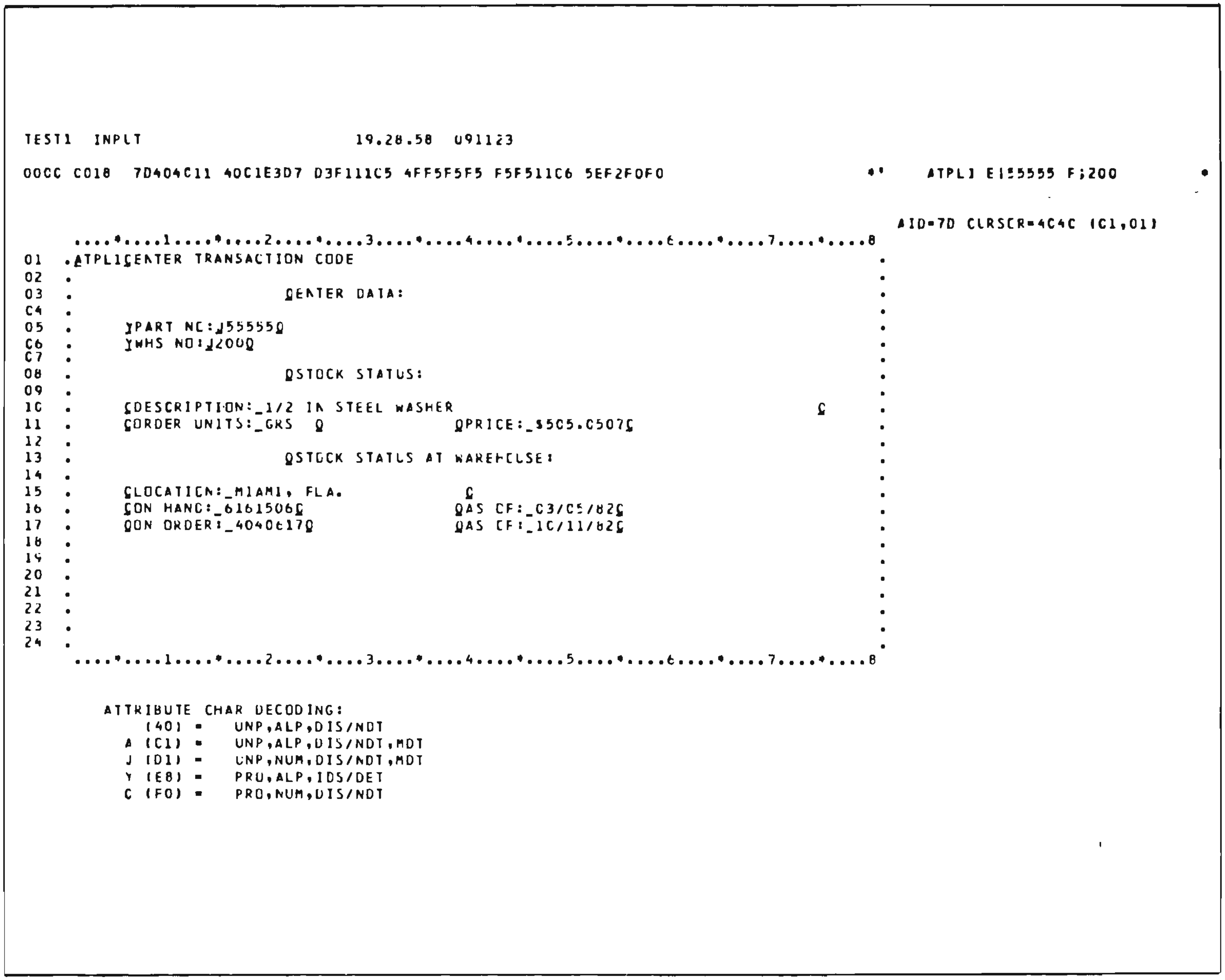
```
TEST1  INPUT                     19.28.53  091123

0000 0018  7D404C11 40C1E3D7 D3F111C5 4FF1F2F3 F4F511C6 5EF2F0F0                     *'    ATPL1 E|12345 F;200         *

                                                                                        AID=7D CURSOR=4C4C (01,01)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
01  .ATPL1CENTER TRANSACTION CODE                                                    .
02  .                                                                                .
03  .                     QENTER DATA:                                               .
04  .                                                                                .
05  .     YPART NO:J12345Q                                                           .
06  .     YWHS NO:J200Q                                                              .
07  .                                                                                .
08  .                     QSTOCK STATUS:                                             .
09  .                                                                                .
10  .     CDESCRIPTION:_                                                        Q    .
11  .     CORDER UNITS:_     C             QPRICE:_         C                        .
12  .                                                                                .
13  .                     CSTOCK STATUS AT WAREHOUSE:                                .
14  .                                                                                .
15  .     CLOCATION:_                       Q                                        .
16  .     CON HAND:_       Q               QAS OF:_        Q                         .
17  .     CON ORDER:_       Q              QAS OF:_        Q                         .
18  .                                                                                .
19  .                                                                                .
20  .                                                                                .
21  .                                                                                .
22  .                                                                                .
23  .                                                                                .
24  .                                                                                .
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8

        ATTRIBUTE CHAR DECODING:
            (40) =    UNP,ALP,DIS/NDT
          A (C1) =    UNP,ALP,DIS/NDT,MDT
          J (D1) =    UNP,NUM,DIS/NDT,MDT
          Y (E8) =    PRO,ALP,IDS/DET
          0 (F0) =    PRO,NUM,DIS/NDT
```

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



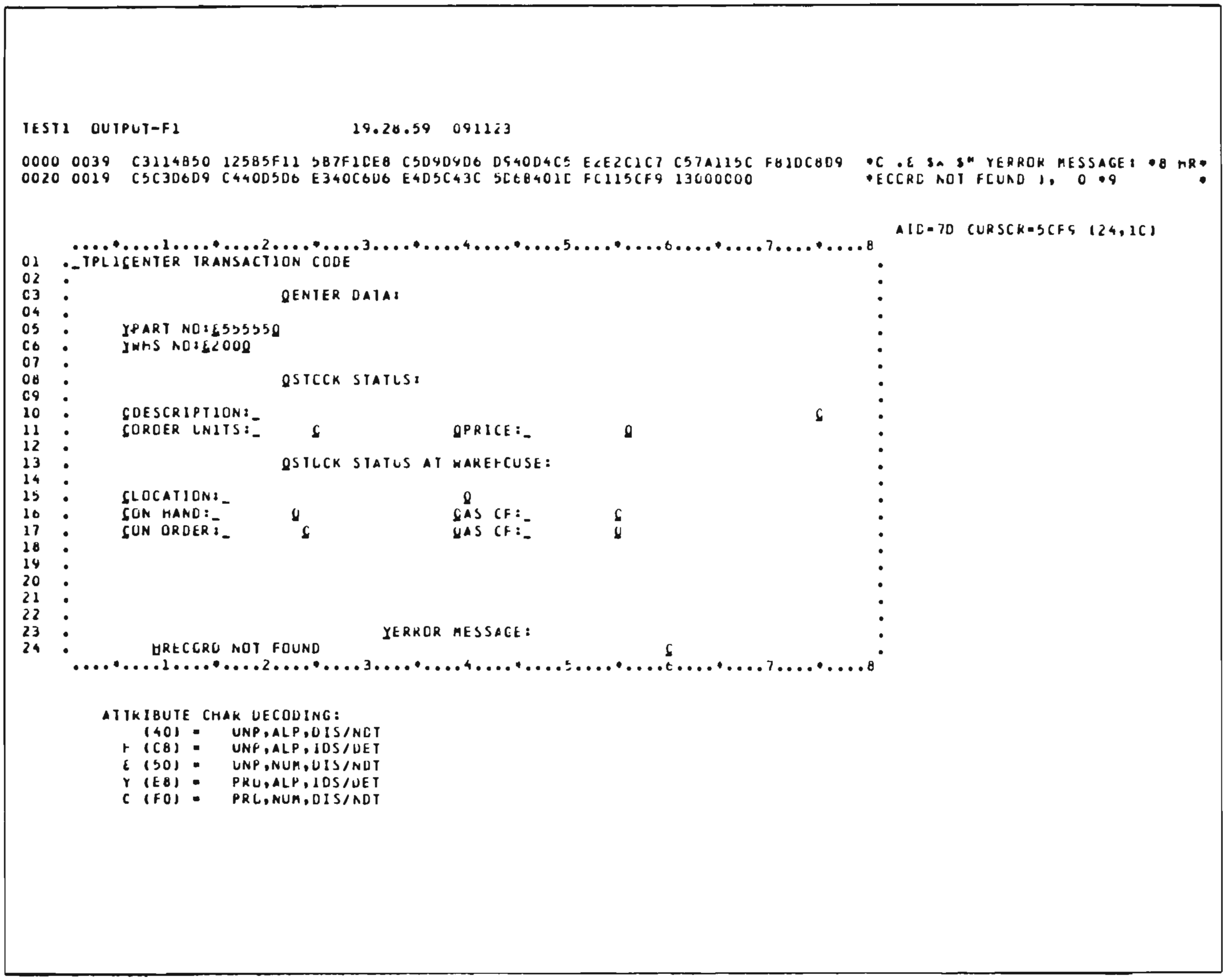
```
TEST1  OUTPUT-F5                    19.26.56  091123

COCO 0171  C3114040 1DC14040 40401DF0 C5D5E3C5 C94CE3D9 C1D5E2C1 C3E3C9D6 D540C3D6  •C     A      CENTER TRANSACTION CO•
0020 0151  C4C511C2 F51DF0C5 D5E3C5D9 40C4C1E3 C17A11C5 C51CE8C7 C1D9E34C D5D67A1C  •CE B5 CENTER CATA: EE YPART NC: •
0040 0131  50F1F2F3 F4F51DF0 11C6D51D E8E6C8E2 4CC5D67A 1D5CF2FC F01DFC11 C9C51DF0  •£12345 0 FN YWHS NO: £200 C IE C•
C060 0111  E2E3D6C3 D240E2E3 C1E3E4E2 7A114BD5 1CF0C4C5 E2C3D9C9 D7E3C9D6 D57A1D40  •STCCK STATLS: .N CDESCRIPTION:  •
0C80 00F1  F161F240 C9D540E2 E3C5C5D3 40E6C1E2 C8C5D93C 4CC94C11 4C5A1DFC 114CE51D  •1/2 IN STEEL WASHER <R  <! C <V •
00A0 00D1  F0D6D9C4 C5D940E4 D5C9E3E2 7A1D40C7 D9E2404C 1DF0114D C61DF0D7 D9C9C3C5  •CORDER UNITS:  GRS   C (F CPRICE•
00C0 00B1  7A1D405B F5F0F54B F0F5F0F7 1DFC114F C51DF0E2 E3D6C3D2 40E2E3C1 E3E4E240  •:  $505.05C7 C IN OSTCCK STATLS •
CCE0 0091  C1E340E6 C1D9C5C8 D6E4E2C5 7A11D1E5 1CF0D3C6 C3C1E3C9 C6D57A1C 4CD4C9C1  •AT WAREHOUSE: JV CLCCATICN:  MIA•
0100 0071  D4C96B40 C6D3C14B 3CD2C740 1DFC11D2 F51CF0D6 C54CC6C1 D5C47A1D 40F6F1F6  •MI, FLA. KG  C K5 OON HAND:  616•
0120 0051  F1F5F0F6 1DF011D3 D61DF0C1 E24CD6C6 7A1C4CFC F3C1FCF5 61F8F21D FC11D4C5  •1506 C LC CAS OF:  C3/C5/82 C ME•
0140 0C31  1DF0D6D5 40D6D9C4 C5D97A1D 40F4F0F4 FCF6F1F7 1CFC11C4 E61DF0C1 E24CD6C6  • CON ORDER:  404C617 C MW CAS OF•
0160 0C11  7A1D4CF1 F061F1F1 61F8F21D F01140C1 13C00000                             •:  1C/11/82 C  A                 •

                                                                               AID=7D CURSCR=4CC1 (01,C2)
      ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
01  •_   CENTER TRANSACTION CODE                                                       •
02  •                                                                                  •
C3  •                     QENTER DATA:                                                 •
04  •                                                                                  •
05  •    YPART NC:£12345C                                                              •
C6  •    YWHS NO:£2000                                                                 •
C7  •                                                                                  •
08  •                     CSTCCK STATLS:                                               •
C9  •                                                                                  •
1C  •    CDESCRIPTION:_1/2 IN STEEL WASHER                                   C         •
11  •    CORDER UNITS:_GRS  Q                     QPRICE:_$505.C507C                   •
12  •                                                                                  •
13  •                     QSTCCK STATLS AT WAREHOUSE:                                  •
14  •                                                                                  •
15  •    CLOCATION:_MIAMI, FLA.                    Q                                   •
16  •    CON HAND:_61615060                        QAS CF:_C3/C5/82C                   •
17  •    CON ORDER:_4040617C                       QAS CF:_1C/11/82C                   •
18  •                                                                                  •
19  •                                                                                  •
20  •                                                                                  •
21  •                                                                                  •
22  •                                                                                  •
23  •                                                                                  •
24  •                                                                                  •
      ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8

     ATTRIBUTE CHAR DECODING:
          (40) =    UNP,ALP,DIS/NDT
        £ (50) =    UNP,NUM,DIS/NDT
        Y (E8) =    PRG,ALP,IDS/DET
        C (F0) =    PRU,NUM,DIS/NCT
```

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



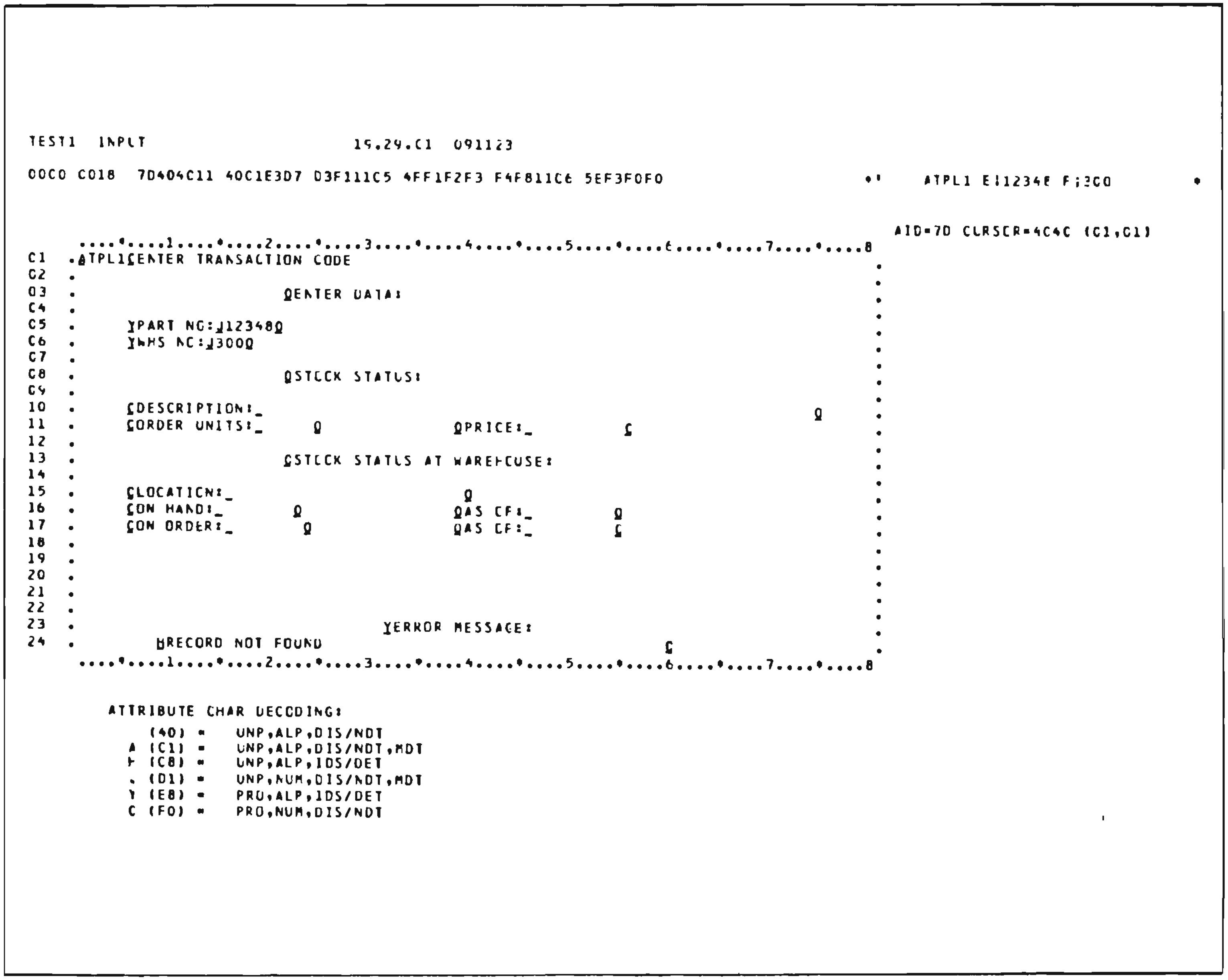
```
TEST1  INPLT                        19.28.58  U91123

000C C018  7D404C11 40C1E3D7 D3F111C5 4FF5F5F5 F5F511C6 5EF2F0F0                     *'    ATPL1 E155555 F;200        *

                                                                                           AID=7D CLRSCR=4C4C (C1,01)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
01  .ATPL1ÇENTER TRANSACTION CODE                                                     .
02  .                                                                                 .
03  .                        QENTER DATA:                                             .
C4  .                                                                                 .
05  .     YPART NC:J55555Q                                                            .
C6  .     YWHS NO:J200Q                                                               .
C7  .                                                                                 .
08  .                        QSTOCK STATUS:                                           .
09  .                                                                                 .
1C  .     ÇDESCRIPTION:_1/2 IN STEEL WASHER                                     Ç     .
11  .     ÇORDER UNITS:_GRS  Q               QPRICE:_$505.C507Ç                       .
12  .                                                                                 .
13  .                        QSTOCK STATUS AT WAREHOUSE:                              .
14  .                                                                                 .
15  .     ÇLOCATION:_MIAMI, FLA.              Ç                                       .
16  .     ÇON HAND:_6161506Ç                 QAS CF:_03/C5/82Ç                        .
17  .     QON ORDER:_4040617Q                QAS CF:_1C/11/82Ç                        .
18  .                                                                                 .
19  .                                                                                 .
20  .                                                                                 .
21  .                                                                                 .
22  .                                                                                 .
23  .                                                                                 .
24  .                                                                                 .
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8


        ATTRIBUTE CHAR DECODING:
            (40) =   UNP,ALP,DIS/NDT
          A (C1) =   UNP,ALP,DIS/NDT,MDT
          J (D1) =   UNP,NUM,DIS/NDT,MDT
          Y (E8) =   PRO,ALP,IDS/DET
          Ç (F0) =   PRO,NUM,DIS/NDT
```

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



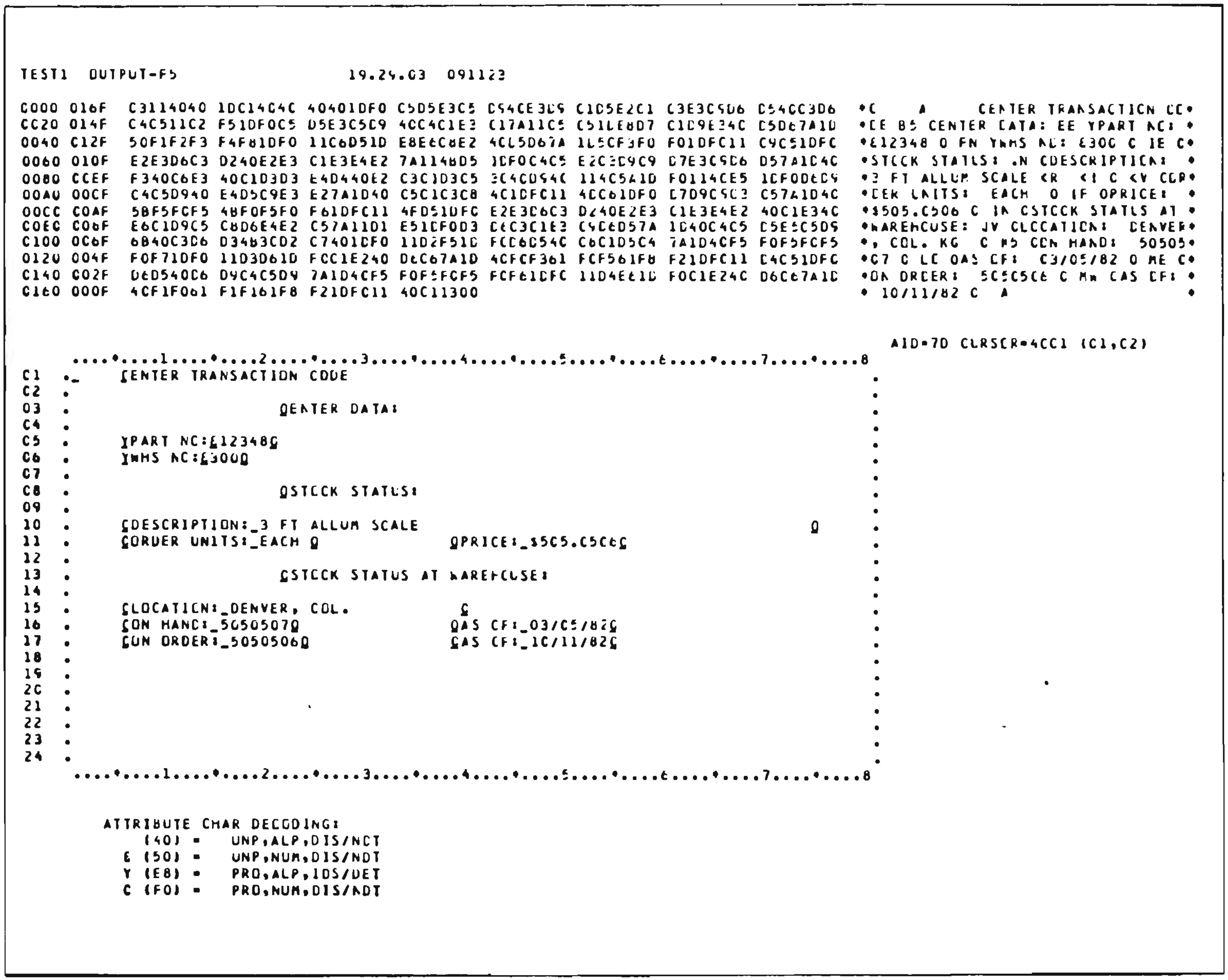
```
TEST1  OUTPUT-F1                 19.28.59  091123

0000 0039  C3114B50 12585F11 5B7F1DE8 C5D9D9D6 D940D4C5 E2E2C1C7 C57A115C F81DC8D9  *C .£ $^ $" YERROR MESSAGE: *8 HR*
0020 0019  C5C3D6D9 C440D5D6 E340C6D6 E4D5C43C 5C68401C FC115CF9 13000000           *ECCRD NOT FOUND ),  0 *9         *

                                                                                        AID=7D CURSCR=5CF5 (24,1C)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
01  ._TPL1£ENTER TRANSACTION CODE                                                     .
02  .                                                                                 .
03  .                     QENTER DATA:                                                .
04  .                                                                                 .
05  .     YPART NO:£55555Q                                                            .
06  .     YWHS NO:£2000Q                                                              .
07  .                                                                                 .
08  .                     QSTOCK STATUS:                                              .
09  .                                                                                 .
10  .     CDESCRIPTION:_                                                        C     .
11  .     CORDER UNITS:_     C             QPRICE:_          Q                        .
12  .                                                                                 .
13  .                     QSTOCK STATUS AT WAREHOUSE:                                 .
14  .                                                                                 .
15  .     CLOCATION:_                        Q                                        .
16  .     CON HAND:_       Q               CAS CF:_         C                         .
17  .     CON ORDER:_       C              QAS CF:_         Q                         .
18  .                                                                                 .
19  .                                                                                 .
20  .                                                                                 .
21  .                                                                                 .
22  .                                                                                 .
23  .                               YERROR MESSAGE:                                   .
24  .        HRECORD NOT FOUND                                   C                    .
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8

        ATTRIBUTE CHAR DECODING:
            (40) =   UNP,ALP,DIS/NDT
          F (C8) =   UNP,ALP,IDS/DET
          £ (50) =   UNP,NUM,DIS/NDT
          Y (E8) =   PRO,ALP,IDS/DET
          C (F0) =   PRO,NUM,DIS/NDT
```

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



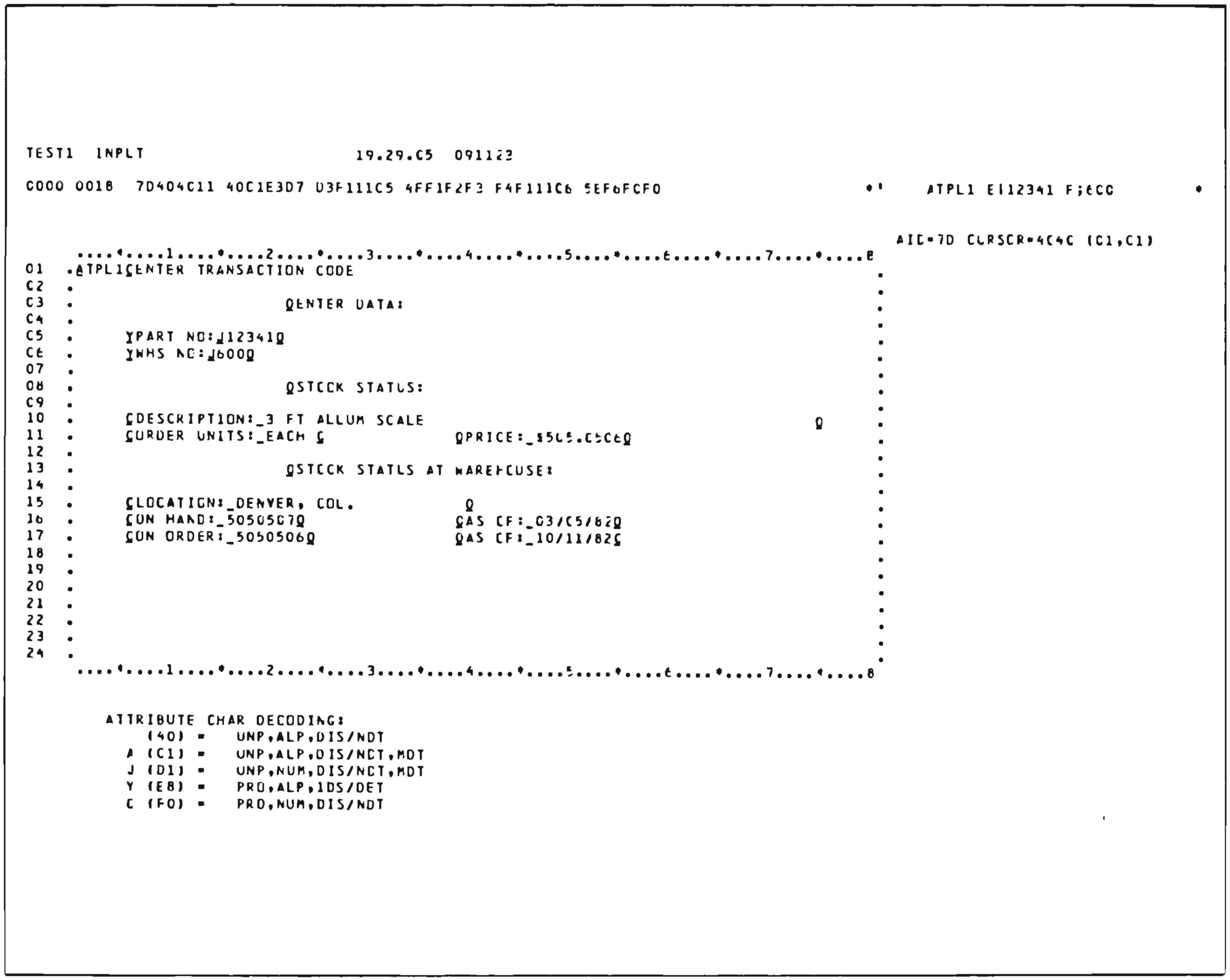
```
TEST1  INPLT                     15.29.01  091123

00C0 C018  7D404C11 40C1E3D7 D3F111C5 4FF1F2F3 F4F811C6 5EF3F0F0                    •'    ATPL1 E:12348 F;300           •

                                                                                       AID=7D CLRSCR=4C4C (01,01)
     ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
C1  .A̲TPL1C̲ENTER TRANSACTION CODE                                                    .
02  .                                                                                .
03  .                    Q̲ENTER UATA:                                                .
C4  .                                                                                .
C5  .     Y̲PART NO:.̲12348Ω                                                           .
C6  .     Y̲WHS NC:.̲300Ω                                                              .
C7  .                                                                                .
C8  .                    Q̲STCCK STATUS:                                              .
C9  .                                                                                .
10  .     C̲DESCRIPTION:_                                                      Ω      .
11  .     C̲ORDER UNITS:_     Ω            Q̲PRICE:_          C̲                        .
12  .                                                                                .
13  .                    C̲STCCK STATLS AT WAREHCUSE:                                 .
14  .                                                                                .
15  .     C̲LOCATICN:_                              Ω                                 .
16  .     C̲ON HAND:_             Ω        Q̲AS OF:_         Ω                         .
17  .     C̲ON ORDER:_             Ω       Q̲AS OF:_         C̲                         .
18  .                                                                                .
19  .                                                                                .
20  .                                                                                .
21  .                                                                                .
22  .                                                                                .
23  .                              Y̲ERROR MESSAGE:                                   .
24  .        F̲RECORD NOT FOUND                                  C̲                    .
     ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8

        ATTRIBUTE CHAR UECCDING:
            (40) =   UNP,ALP,DIS/NDT
          A (C1) =   UNP,ALP,DIS/NDT,MDT
          F (C8) =   UNP,ALP,IDS/DET
          . (D1) =   UNP,NUM,DIS/NDT,MDT
          Y (E8) =   PRO,ALP,IDS/DET
          C (F0) =   PRO,NUM,DIS/NDT
```

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



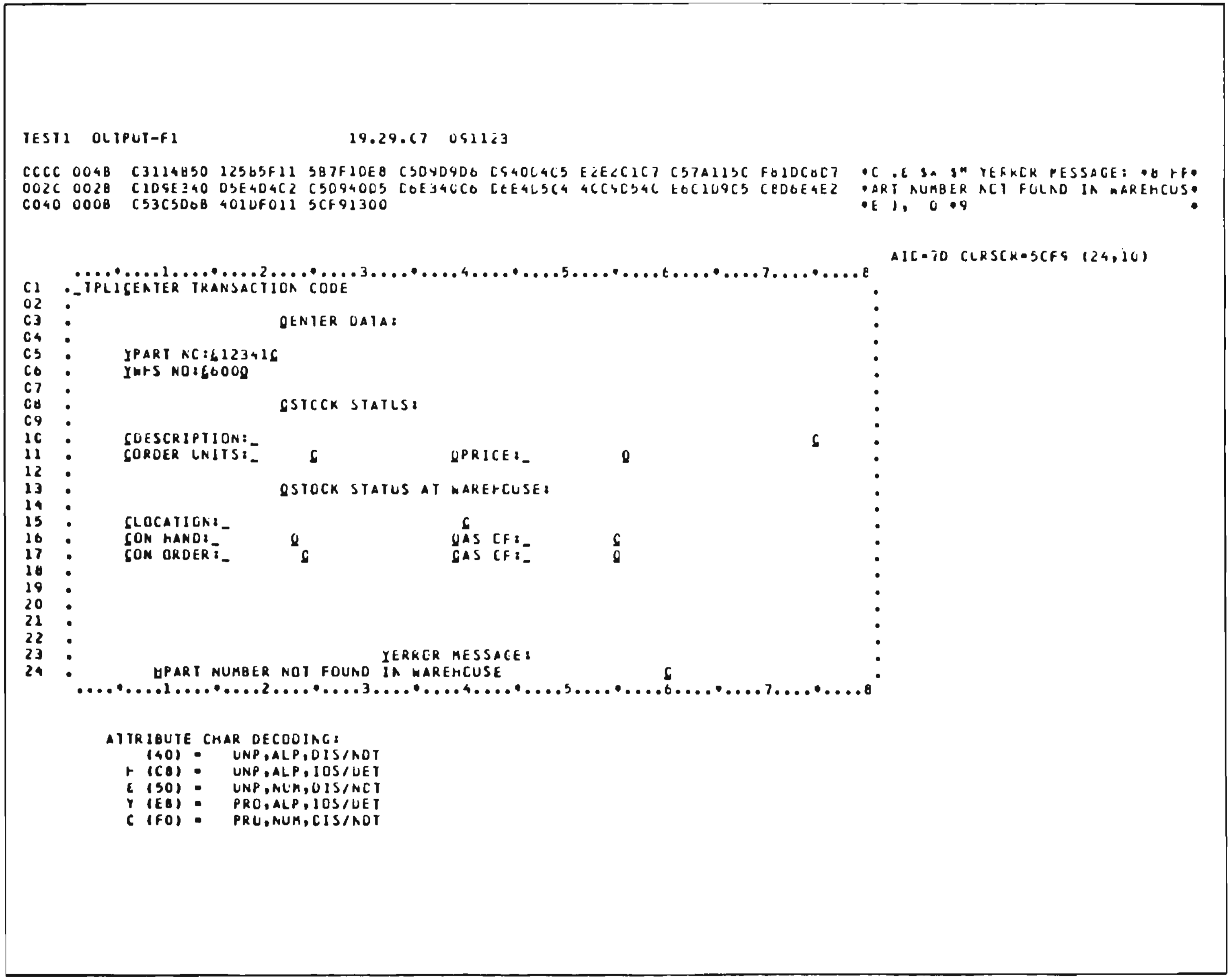
```
TESTI  OUTPUT-F5                    19.24.03  091123

C000 016F  C3114040 1DC14C4C 40401DF0 C5D5E3C5 C54CE3C9 C1D5E2C1 C3E3C9D6 C54CC3D6  •C     A      CENTER TRANSACTICN CC•
CC20 014F  C4C511C2 F51DF0C5 D5E3C5C9 4CC4C1E3 C17A11C5 C51LE8D7 C1C9E34C C5D67A1D  •CE B5 CENTER CATA: EE YPART NC: •
0040 C12F  50F1F2F3 F4F81DF0 11C6D51D E8E6C8E2 4CL5D67A 1L5CF3F0 F01DFC11 C9C51DFC  •£12348 0 FN YWHS NC: £30C C IE C•
0060 010F  E2E3D6C3 D240E2E3 C1E3E4E2 7A114BD5 1DF0C4C5 E2C3C9C9 D7E3C9D6 D57A1C4C  •STCCK STATLS: .N CDESCRIPTICN:  •
0080 CCEF  F340C6E3 40C1D3D3 E4D440E2 C3C1D3C5 3C4CD94C 114C5A1D F0114CE5 1CF0D6C9  •3 FT ALLUM SCALE <R  <I C <V CGR•
00A0 00CF  C4C5D940 E4D5C9E3 E27A1D40 C5C1C3C8 4C1DFC11 4CC61DF0 D7D9C9C3 C57A1D4C  •CER UNITS:  EACH  0 (F OPRICE:  •
00CC C0AF  5BF5FCF5 4BF0F5F0 F61DFC11 4FD51DFC E2E3C6C3 D240E2E3 C1E3E4E2 40C1E34C  •$505.C506 C IN CSTCCK STATLS AT •
C0EC C06F  E6C1D9C5 C8D6E4E2 C57A11D1 E51CF0D3 C6C3C1E3 C9C6D57A 1D40C4C5 D5E5C5D9  •WAREHCUSE: JV CLCCATICN:  DENVER•
C100 0C6F  6B40C3D6 D34B3CD2 C7401CF0 11D2F51C FCC6C54C C6C1D5C4 7A1D4CF5 F0F5FCF5  •, COL. KG  C W5 CCN HAND:  50505•
0120 004F  F0F71DF0 11D3D61D FCC1E240 D6C67A1D 4CFCF361 FCF561F8 F21DFC11 C4C51DFC  •C7 C LC OAS CF:  C3/05/82 0 ME C•
C140 C02F  D6D540C6 D9C4C5D9 7A1D4CF5 F0F5FCF5 FCF61CFC 11D4E61D F0C1E24C D6C67A1C  •ON ORCER:  5C5C5C6 C MW CAS CF: •
C160 000F  4CF1F061 F1F161F8 F21DFC11 40C11300                                       • 10/11/82 C  A                   •

                                                                                          AID=7D CURSCR=4CC1 (C1,C2)
     ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
C1   •_   CENTER TRANSACTION CODE                                                      •
C2   •                                                                                 •
03   •                        QENTER DATA:                                             •
C4   •                                                                                 •
C5   •    YPART NC:£12348C                                                             •
C6   •    YWHS NC:£300Q                                                                •
C7   •                                                                                 •
C8   •                        QSTCCK STATUS:                                           •
09   •                                                                                 •
10   •    CDESCRIPTION:_3 FT ALLUM SCALE                                          Q        •
11   •    CORDER UNITS:_EACH Q                  QPRICE:_$5C5.C5C6C                         •
12   •                                                                                 •
13   •                        CSTCCK STATUS AT WAREHCUSE:                              •
14   •                                                                                 •
15   •    CLOCATICN:_DENVER, COL.                C                                     •
16   •    CON HANC:_5C505070                    QAS CF:_03/C5/82C                          •
17   •    CON ORDER:_5050506Q                   CAS CF:_1C/11/82C                          •
18   •                                                                                 •
19   •                                                                                 •
2C   •                                                                                 •
21   •                                                                                 •
22   •                                                                                 •
23   •                                                                                 •
24   •                                                                                 •
     ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8

      ATTRIBUTE CHAR DECODING:
          (40) =   UNP,ALP,DIS/NCT
        £ (50) =   UNP,NUM,DIS/NDT
        Y (E8) =   PRO,ALP,IDS/DET
        C (F0) =   PRO,NUM,DIS/NDT
```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 8 of 22)



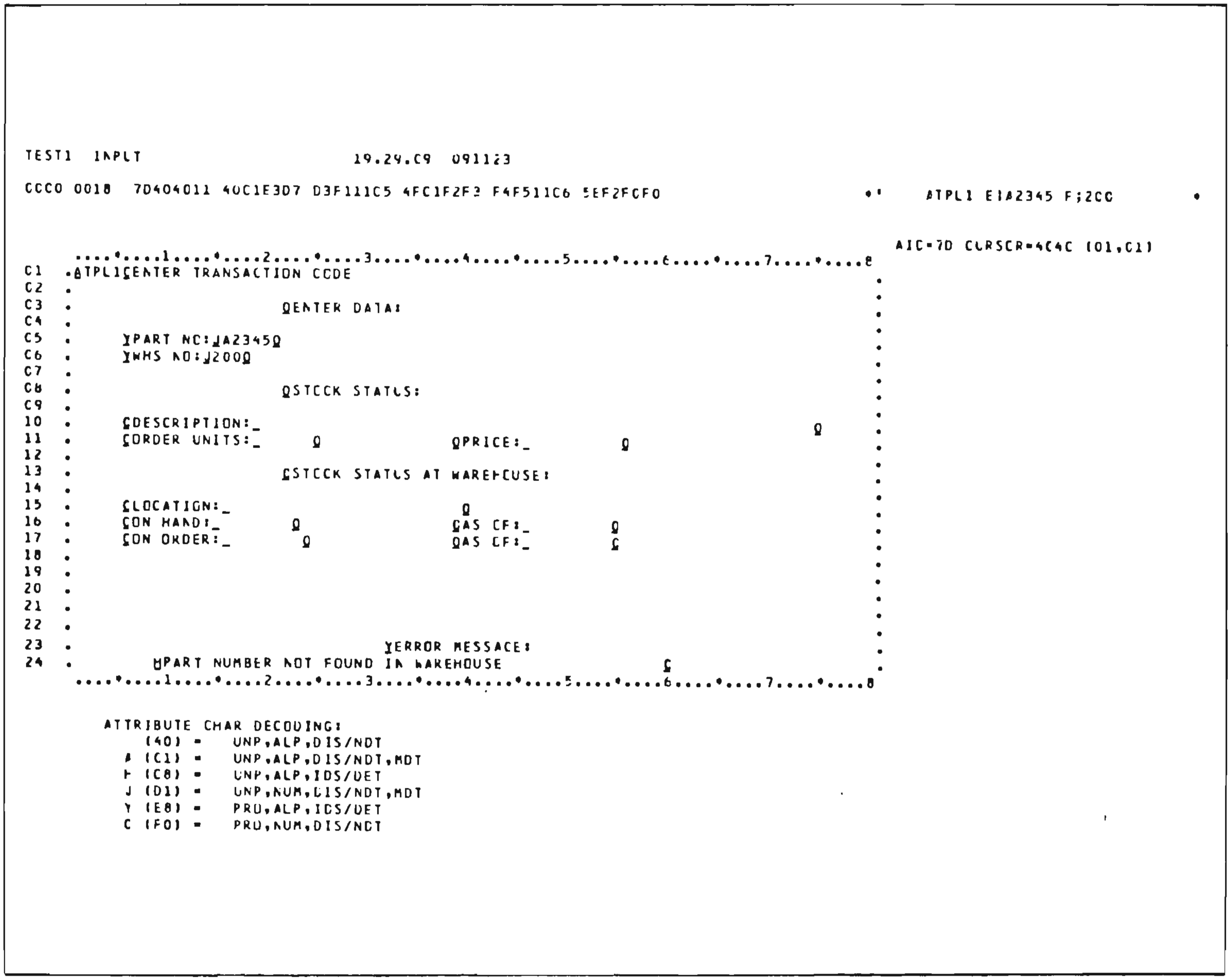
```
TEST1  INPLT                     19.29.C5  091123

C000 0018  7D404C11 40C1E3D7 D3F111C5 4FF1F2F3 F4F111C6 5EF6FCF0                     *'     ATPL1 E112341 F;6CC        *

                                                                                       AID=7D CURSCR=4C4C (C1,C1)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
01  .ATPL1CENTER TRANSACTION CODE                                                    .
C2  .                                                                                .
C3  .                    QENTER DATA:                                                .
C4  .                                                                                .
C5  .    YPART NO:J12341Q                                                            .
C6  .    YWHS NC:J600Q                                                               .
07  .                                                                                .
08  .                    QSTCCK STATUS:                                              .
C9  .                                                                                .
10  .    CDESCRIPTION:_3 FT ALLUM SCALE                                       Q      .
11  .    CURDER UNITS:_EACH C             QPRICE:_$505.C5C6Q                         .
12  .                                                                                .
13  .                    QSTCCK STATLS AT WAREFCUSE:                                 .
14  .                                                                                .
15  .    CLOCATION:_DENVER, COL.           Q                                         .
16  .    CON HAND:_5050507Q               CAS CF:_03/C5/82Q                          .
17  .    CON ORDER:_5050506Q              QAS CF:_10/11/82C                          .
18  .                                                                                .
19  .                                                                                .
20  .                                                                                .
21  .                                                                                .
22  .                                                                                .
23  .                                                                                .
24  .                                                                                .
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8


        ATTRIBUTE CHAR DECODING:
            (40) =     UNP,ALP,DIS/NDT
          A (C1) =     UNP,ALP,DIS/NCT,MDT
          J (D1) =     UNP,NUM,DIS/NCT,MDT
          Y (E8) =     PRO,ALP,IDS/DET
          C (F0) =     PRO,NUM,DIS/NDT
```

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



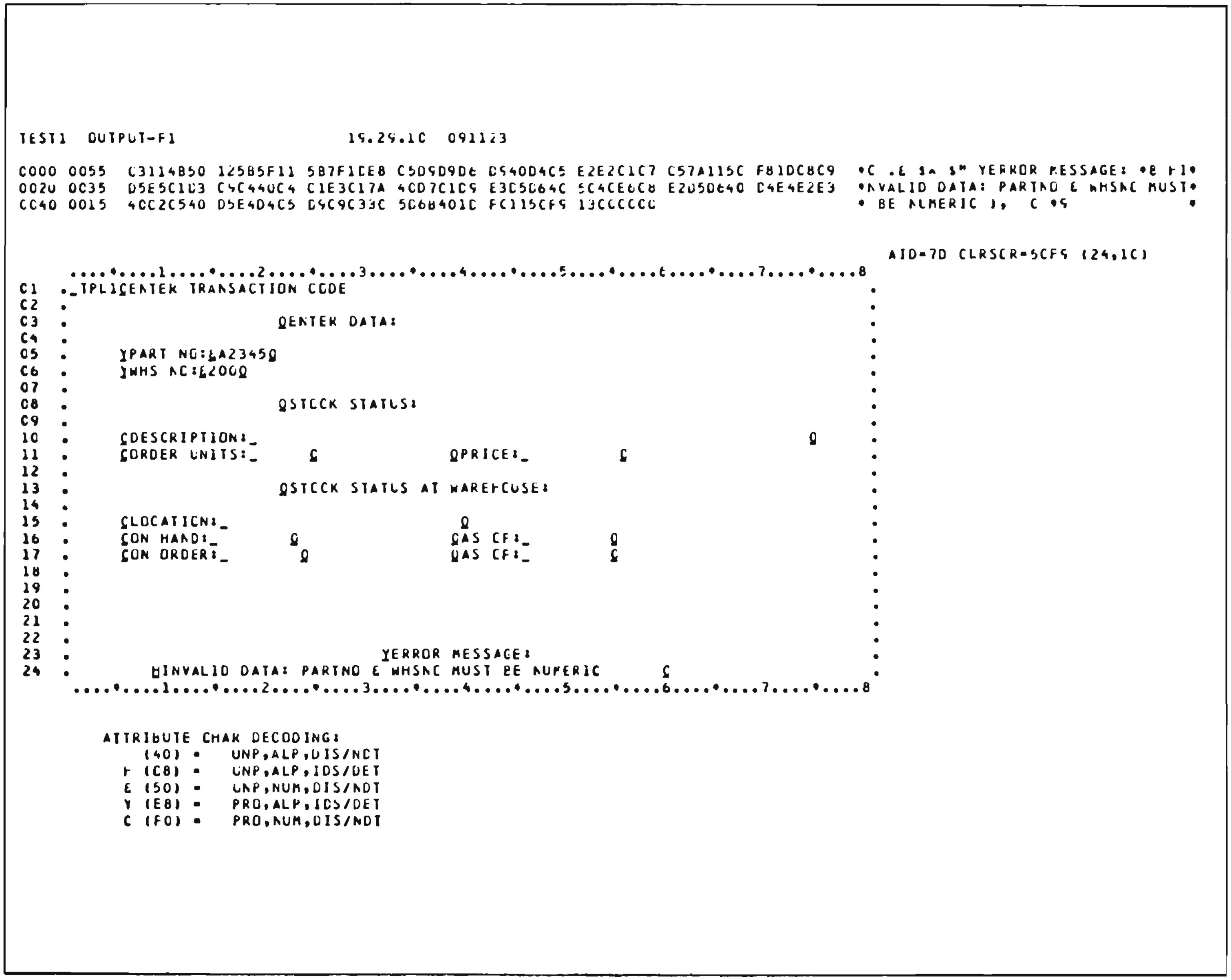
```
TEST1  OUTPUT-F1                  19.29.C7  0S1123

CCCC 004B  C3114B50 125B5F11 5B7F10E8 C5D9D9D6 C540C4C5 E2E2C1C7 C57A115C F61DC8C7  •C .Ɛ $^ $" YERRCR MESSAGE: •b ⊦F•
002C 0028  C1D9E340 D5E4D4C2 C5094005 C6E340C6 C6E4D5C4 4CC5C54C E6C1D9C5 C8D6E4E2  •ART NUMBER NCT FOUND IN WAREHCUS•
C040 000B  C53C5D6B 401DF011 5CF91300                                               •E ), 0 •9                        •

                                                                                          AIC=7D CURSCR=5CF9 (24,10)
       ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
C1    •_TPL1C̲ENTER TRANSACTION CODE                                                     •
02    •                                                                                 •
C3    •                    Q̲ENTER DATA:                                                  •
C4    •                                                                                 •
C5    •     Y̲PART NC:Ɛ̲12341C̲                                                            •
C6    •     Y̲WFS NO:Ɛ̲600Q̲                                                               •
C7    •                                                                                 •
C8    •                    C̲STCCK STATUS:                                               •
C9    •                                                                                 •
1C    •     C̲DESCRIPTION:_                                                   C̲          •
11    •     C̲ORDER UNITS:_     C̲              Q̲PRICE:_         Q̲                       •
12    •                                                                                 •
13    •                    Q̲STOCK STATUS AT WAREHCUSE:                                  •
14    •                                                                                 •
15    •     C̲LOCATION:_                       C̲                                        •
16    •     C̲ON HAND:_        Q̲               Q̲AS CF:_         C̲                       •
17    •     C̲ON ORDER:_        C̲              C̲AS CF:_         Q̲                       •
18    •                                                                                 •
19    •                                                                                 •
20    •                                                                                 •
21    •                                                                                 •
22    •                                                                                 •
23    •                                  Y̲ERRCR MESSAGE:                                •
24    •       ⊦PART NUMBER NOT FOUND IN WAREHCUSE                C̲                      •
       ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8


          ATTRIBUTE CHAR DECODING:
               (40) =   UNP,ALP,DIS/NDT
             ⊦ (C8) =   UNP,ALP,IDS/DET
             Ɛ (50) =   UNP,NUM,DIS/NCT
             Y (E8) =   PRO,ALP,IDS/DET
             C (F0) =   PRU,NUM,CIS/NDT
```

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



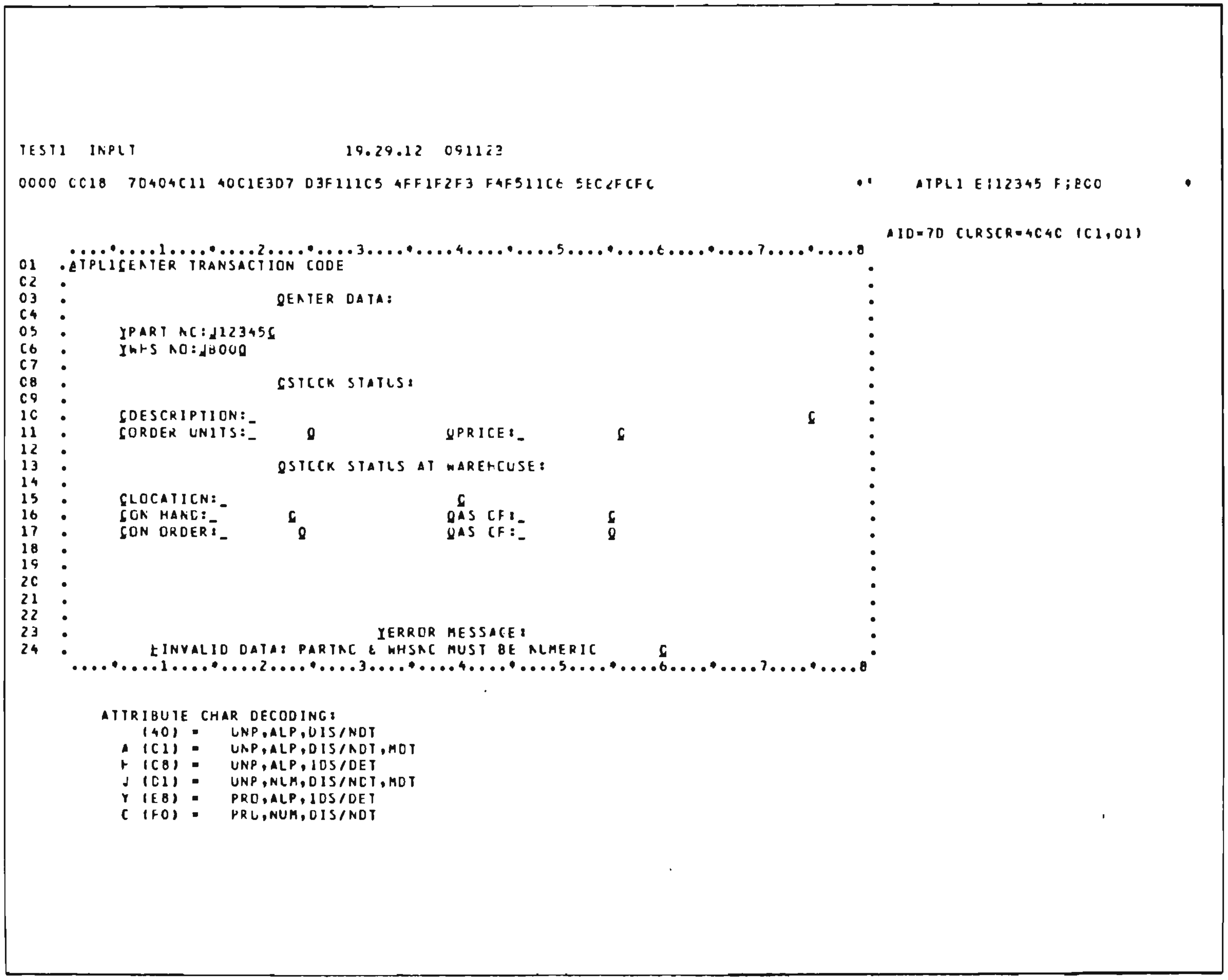
```
TEST1  INPLT                     19.29.C9  091123

CCCO 0018  7D404011 40C1E3D7 D3F111C5 4FC1F2F3 F4F511C6 5EF2FCFO                    .'     ATPL1 EIA2345 F;2CC        .

                                                                                       AIC=7D CURSCR=4C4C (01,C1)
      ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
C1   .ATPL1CENTER TRANSACTION CCDE                                                    .
C2   .                                                                                .
C3   .                     QENTER DATA:                                               .
C4   .                                                                                .
C5   .     YPART NC:JA2345Q                                                           .
C6   .     YWHS NO:J200Q                                                              .
C7   .                                                                                .
C8   .                     QSTCCK STATLS:                                             .
C9   .                                                                                .
10   .     CDESCRIPTION:_                                                          Q    .
11   .     CORDER UNITS:_     Q             QPRICE:_         Q                        .
12   .                                                                                .
13   .                     CSTCCK STATLS AT WAREHCUSE:                                .
14   .                                                                                .
15   .     CLOCATIGN:_                       Q                                        .
16   .     CON HAND:_       Q               CAS CF:_        Q                         .
17   .     CON ORDER:_       Q              QAS CF:_        C                         .
18   .                                                                                .
19   .                                                                                .
20   .                                                                                .
21   .                                                                                .
22   .                                                                                .
23   .                               YERROR MESSACE:                                  .
24   .        HPART NUMBER NOT FOUND IN WAREHOUSE                 C                   .
      ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8

        ATTRIBUTE CHAR DECODING:
            (40) =    UNP,ALP,DIS/NDT
          A (C1) =    UNP,ALP,DIS/NDT,MDT
          H (C8) =    UNP,ALP,IDS/DET
          J (D1) =    UNP,NUM,DIS/NDT,MDT
          Y (E8) =    PRU,ALP,ICS/DET
          C (F0) =    PRU,NUM,DIS/NCT
```

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



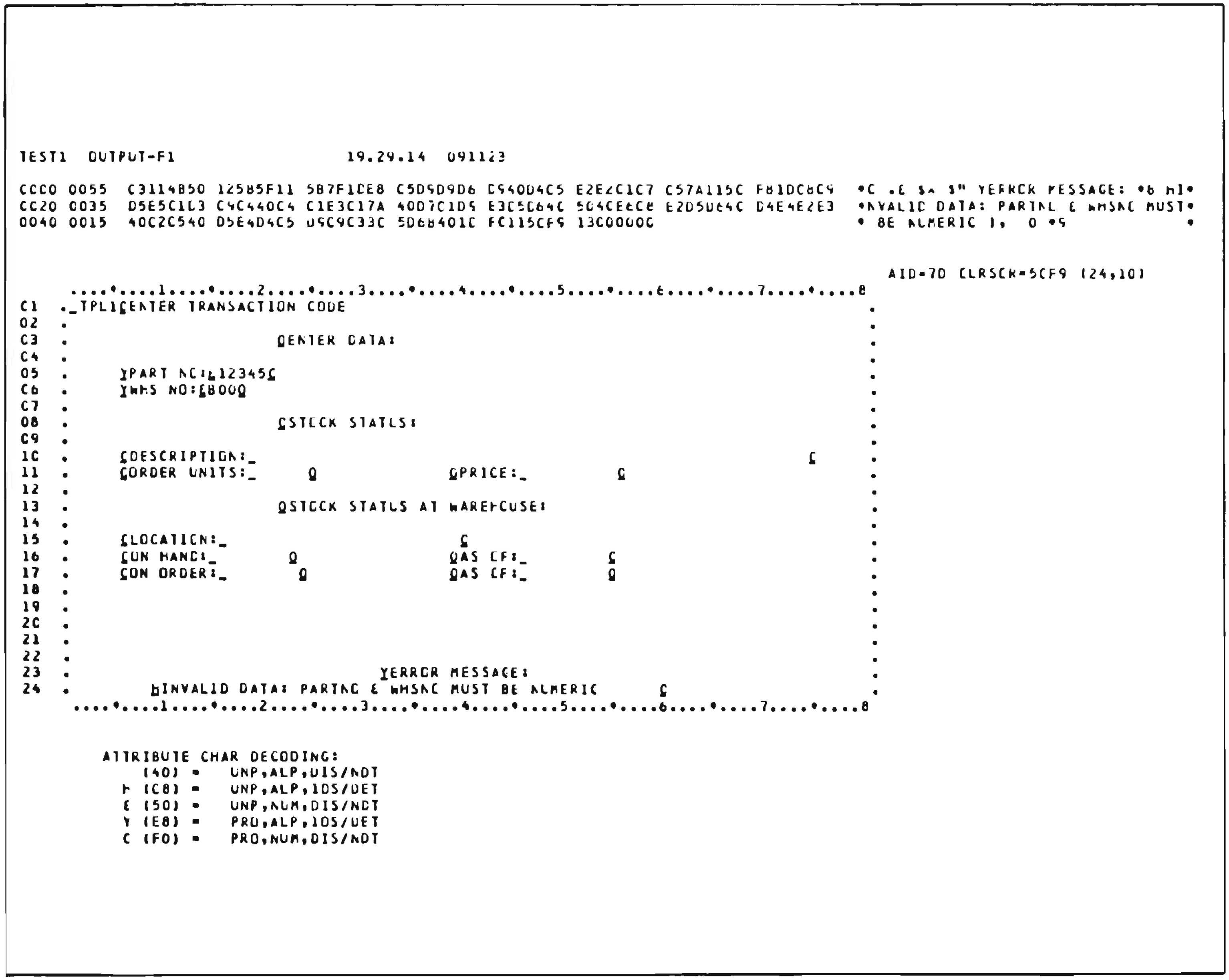
```
TEST1  OUTPUT-F1                  19.29.1C  091123

C000 0055  C3114B50 125B5F11 5B7F1CE8 C5D9D9D6 D940D4C5 E2E2C1C7 C57A115C F81DC8C9  *C .& $^ $" YERROR MESSAGE: *8 H1*
0020 0C35  D5E5C1D3 C9C440C4 C1E3C17A 40D7C1C9 E3D5D64C 5C4CE6C8 E2D5D640 D4E4E2E3  *NVALID DATA: PARTNO & WHSNC MUST*
CC40 0015  40C2C540 D5E4D4C5 D9C9C33C 5D6B401C FC115CF9 13CCCCCC                    * BE NUMERIC ),  C *9               *


                                                                                       AID=7D CLRSCR=5CF9 (24,1C)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
C1   *_TPL1CENTER TRANSACTION CODE                                                   *
C2   *                                                                                *
C3   *                    QENTER DATA:                                                *
C4   *                                                                                *
05   *     YPART NO:&A2345Q                                                           *
C6   *     YWHS NC:&200Q                                                              *
07   *                                                                                *
C8   *                    QSTOCK STATUS:                                              *
C9   *                                                                                *
10   *     CDESCRIPTION:_                                                      Q      *
11   *     CORDER UNITS:_     C              QPRICE:_         C                       *
12   *                                                                                *
13   *                    QSTOCK STATUS AT WAREHOUSE:                                 *
14   *                                                                                *
15   *     CLOCATION:_                        Q                                       *
16   *     CON HAND:_       Q                CAS CF:_        Q                        *
17   *     CON ORDER:_       Q               QAS CF:_        C                        *
18   *                                                                                *
19   *                                                                                *
20   *                                                                                *
21   *                                                                                *
22   *                                                                                *
23   *                                YERROR MESSAGE:                                 *
24   *        HINVALID DATA: PARTNO & WHSNC MUST BE NUMERIC      C                    *
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8


        ATTRIBUTE CHAR DECODING:
            (40) =   UNP,ALP,DIS/NDT
          H (C8) =   UNP,ALP,IDS/DET
          & (50) =   UNP,NUM,DIS/NDT
          Y (E8) =   PRO,ALP,IDS/DET
          C (F0) =   PRO,NUM,DIS/NDT
```

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



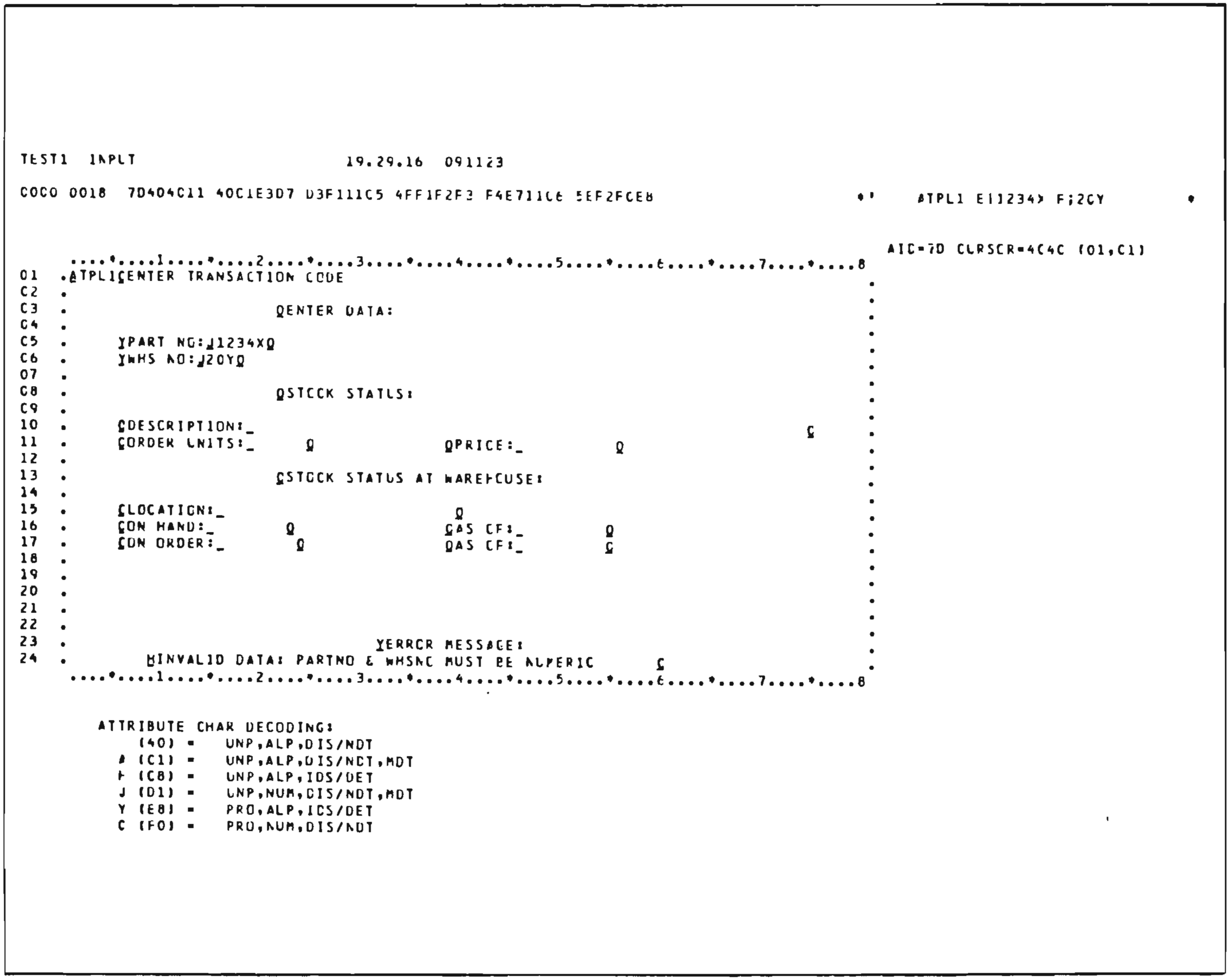
```
TEST1  INPUT                        19.29.12  091123

0000 CC18  70404C11 40C1E3D7 D3F111C5 4FF1F2F3 F4F511C6 5EC2F0F0                  •'     ATPL1 E:12345 F;BCO        •

                                                                                       AID=7D CURSCR=4C4C (C1,01)
      ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
01   .ATPL1CENTER TRANSACTION CODE                                                    .
C2   .                                                                                .
03   .                      QENTER DATA:                                              .
C4   .                                                                                .
05   .     YPART NO:J12345C                                                           .
C6   .     YWHS NO:JB00Q                                                              .
C7   .                                                                                .
C8   .                      CSTOCK STATUS:                                            .
C9   .                                                                                .
1C   .     CDESCRIPTION:_                                                      C      .
11   .     CORDER UNITS:_    Q               QPRICE:_        C                        .
12   .                                                                                .
13   .                      QSTOCK STATUS AT WAREHOUSE:                               .
14   .                                                                                .
15   .     CLOCATION:_                        C                                       .
16   .     CON HAND:_       C                QAS OF:_        C                        .
17   .     CON ORDER:_       Q               QAS OF:_        Q                        .
18   .                                                                                .
19   .                                                                                .
2C   .                                                                                .
21   .                                                                                .
22   .                                                                                .
23   .                                      YERROR MESSAGE:                           .
24   .        EINVALID DATA: PARTNC & WHSNC MUST BE NUMERIC       C                   .
      ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8

        ATTRIBUTE CHAR DECODING:
            (40) =    UNP,ALP,DIS/NDT
          A (C1) =    UNP,ALP,DIS/NDT,MDT
          F (C8) =    UNP,ALP,IDS/DET
          J (D1) =    UNP,NUM,DIS/NDT,MDT
          Y (E8) =    PRO,ALP,IDS/DET
          C (F0) =    PRO,NUM,DIS/NDT
```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 13 of 22)



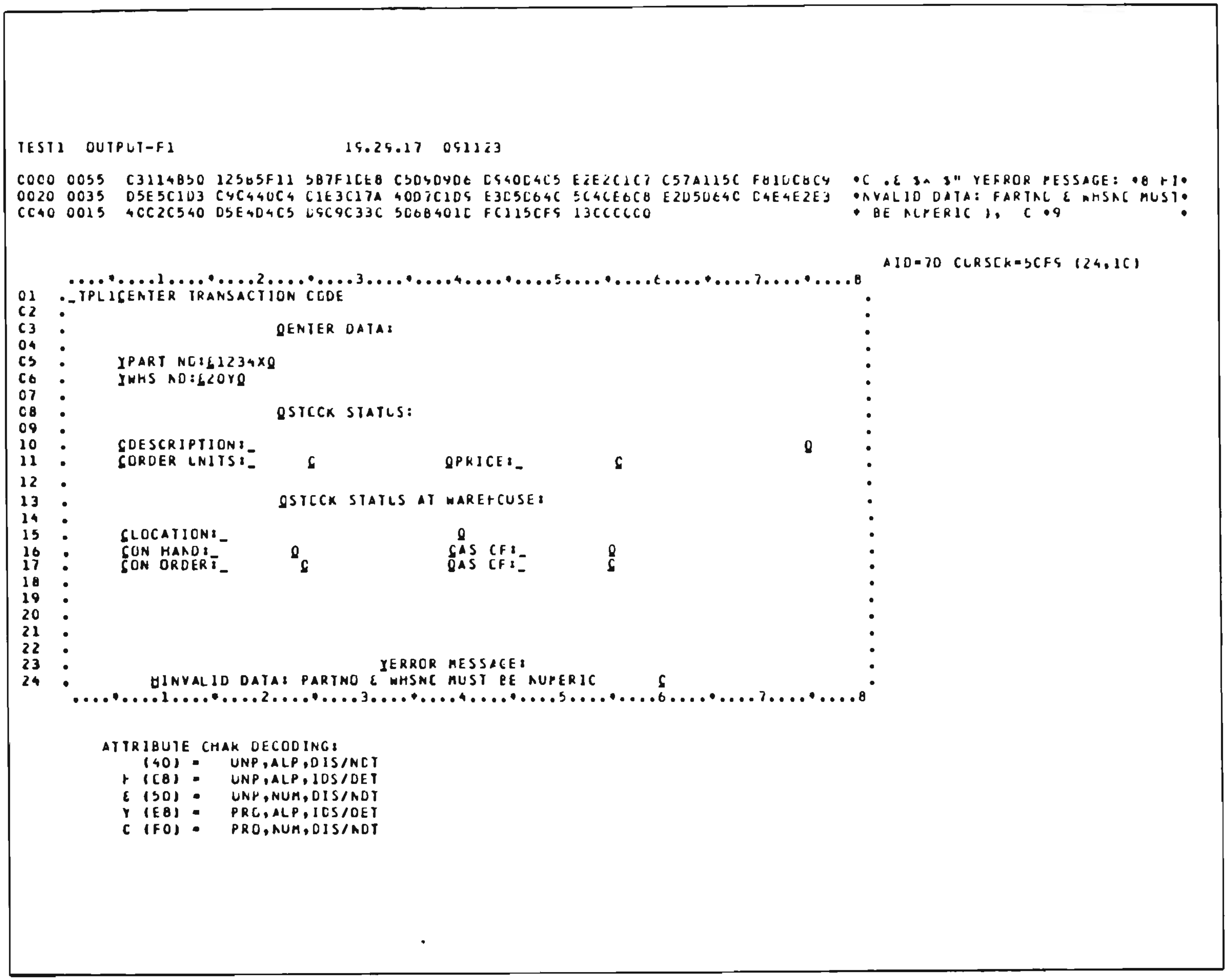
```
TEST1  OUTPUT-F1                    19.29.14  091123

CCCO 0055  C3114B50 125B5F11 5B7F1CE8 C5D9D9D6 D940D4C5 E2E2C1C7 C57A115C F81DC8C9  *C .E $A $" YERROR MESSAGE: *6 HI*
CC20 0035  D5E5C1C3 C9C440C4 C1E3C17A 40D7C1D9 E3C5C64C 5C4CE6C8 E2D5U64C D4E4E2E3  *NVALID DATA: PARTNC & WHSNC MUST*
0040 0015  40C2C540 D5E4D4C5 D9C9C33C 5D6B401C FC115CF9 13C0000C                    * BE NUMERIC ),  0 *9             *

                                                                                           AID=7D CLRSCR=5CF9 (24,10)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
C1   ._TPL1ÇENTER TRANSACTION CODE                                                    .
02   .                                                                                .
C3   .                    QENTER DATA:                                                .
C4   .                                                                                .
05   .     YPART NC:Ŀ12345Ç                                                           .
C6   .     YWHS NO:Ŀ800Q                                                              .
C7   .                                                                                .
08   .                    ÇSTCCK STATLS:                                              .
C9   .                                                                                .
1C   .     ÇDESCRIPTION:_                                                      Ç      .
11   .     ÇORDER UNITS:_      Q                QPRICE:_          Ç                   .
12   .                                                                                .
13   .                    QSTCCK STATLS AT WAREHOUSE:                                 .
14   .                                                                                .
15   .     ÇLOCATION:_                          Ç                                     .
16   .     ÇON HAND:_        Q                  QAS CF:_          Ç                   .
17   .     ÇON ORDER:_        Q                 QAS CF:_          Q                   .
18   .                                                                                .
19   .                                                                                .
2C   .                                                                                .
21   .                                                                                .
22   .                                                                                .
23   .                              YERROR MESSAGE:                                   .
24   .       ḃINVALID DATA: PARTNC & WHSNC MUST BE NUMERIC       Ç                    .
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8


        ATTRIBUTE CHAR DECODING:
            (40) =    UNP,ALP,DIS/NDT
          H (C8) =    UNP,ALP,IDS/DET
          E (50) =    UNP,NUM,DIS/NDT
          Y (E8) =    PRO,ALP,IDS/DET
          C (F0) =    PRO,NUM,DIS/NDT
```

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



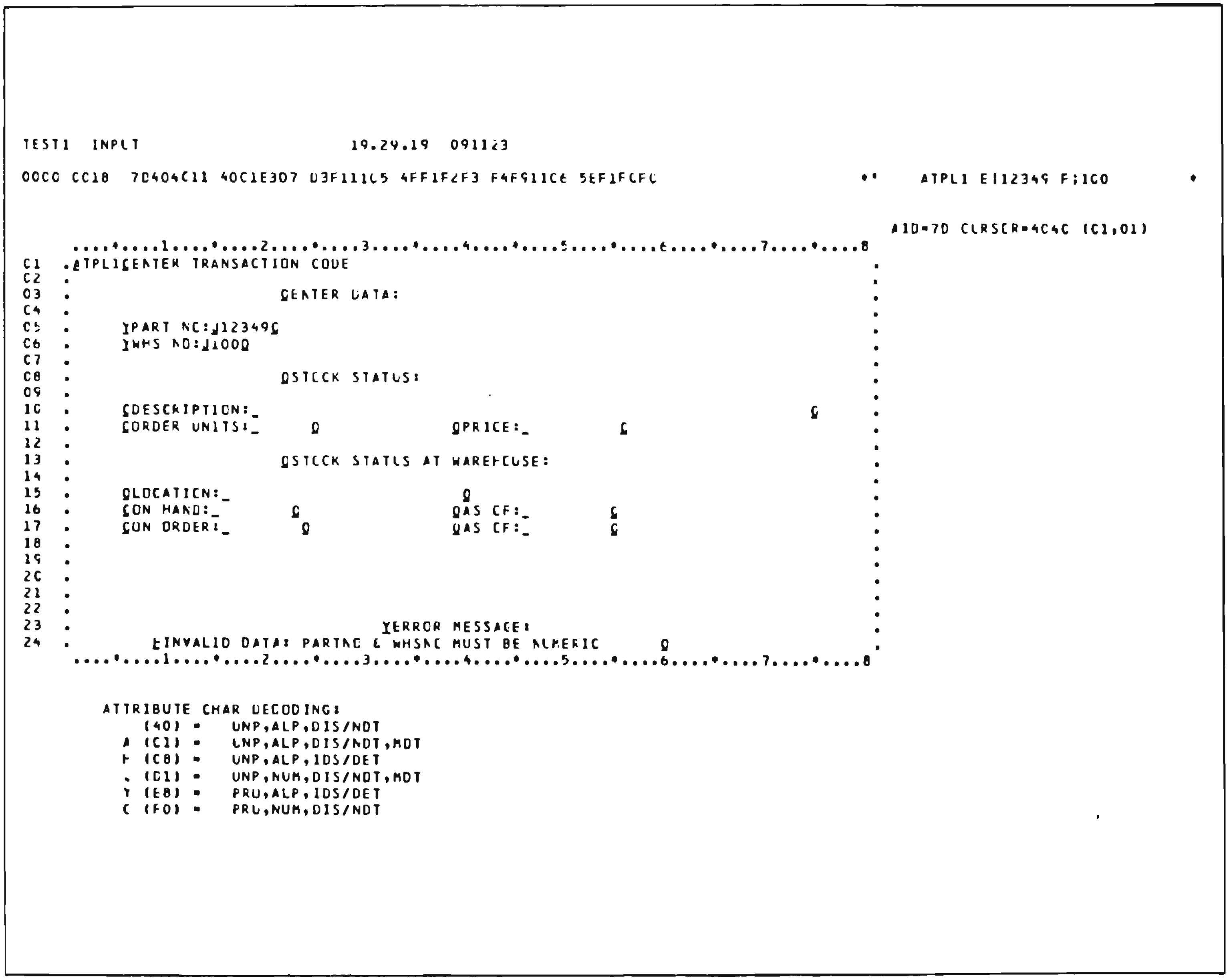
```
TEST1  INPLT                     19.29.16  091123

COCO 0018  7D404C11 40C1E3D7 D3F111C5 4FF1F2F3 F4E711C6 5EF2FCE8                     •'    ATPL1 E11234> F;2CY        •

                                                                                        AIC=7D CLRSCR=4C4C (01,C1)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
01  •ATPL1CENTER TRANSACTION CCDE                                                     •
C2  •                                                                                 •
C3  •                     QENTER DATA:                                                •
C4  •                                                                                 •
C5  •     YPART NG:J1234XQ                                                            •
C6  •     YWHS NO:J20YQ                                                               •
07  •                                                                                 •
C8  •                     QSTCCK STATLS:                                              •
C9  •                                                                                 •
10  •     CDESCRIPTION:_                                                        C     •
11  •     CORDER UNITS:_     Q             QPRICE:_         Q                         •
12  •                                                                                 •
13  •                     CSTOCK STATUS AT WAREHCUSE:                                 •
14  •                                                                                 •
15  •     CLOCATION:_                       Q                                         •
16  •     CON HAND:_       Q               CAS CF:_        Q                          •
17  •     CON ORDER:_       Q              QAS CF:_        C                          •
18  •                                                                                 •
19  •                                                                                 •
20  •                                                                                 •
21  •                                                                                 •
22  •                                                                                 •
23  •                               YERRCR MESSAGE:                                   •
24  •        HINVALID DATA: PARTNO & WHSNC MUST BE ALPERIC       C                    •
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8


        ATTRIBUTE CHAR DECODING:
            (40) =   UNP,ALP,DIS/NDT
          # (C1) =   UNP,ALP,DIS/NCT,MDT
          ⊦ (C8) =   UNP,ALP,IDS/DET
          J (D1) =   UNP,NUM,CIS/NDT,MDT
          Y (E8) =   PRO,ALP,ICS/DET
          C (F0) =   PRO,NUM,DIS/NDT
```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 15 of 22)



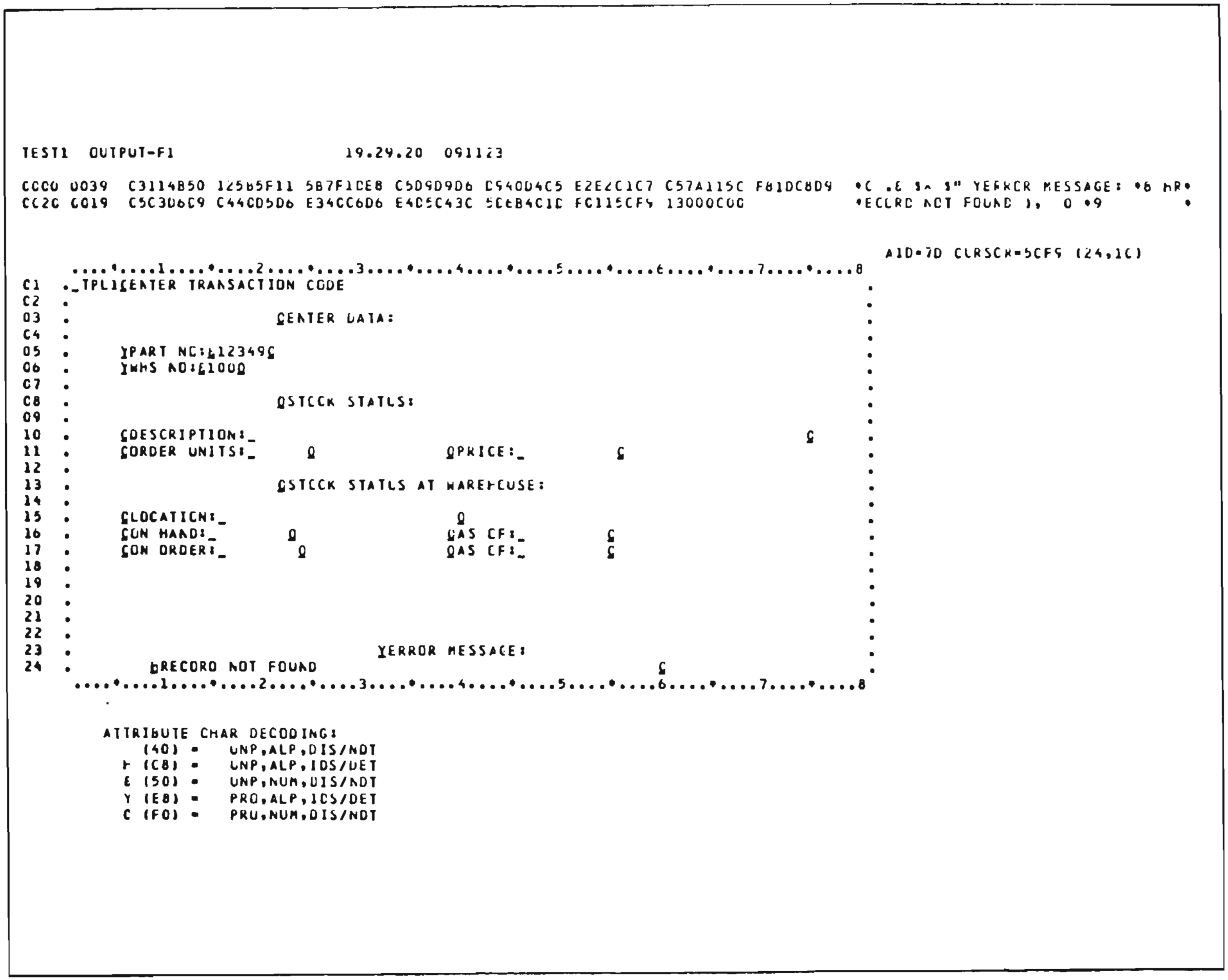
```
TEST1  OUTPUT-F1                     19.29.17  091123

C000 0055  C3114B50 12565F11 5B7F1CE8 C5D9D9D6 D940D4C5 E2E2C1C7 C57A115C F81DC8C9  •C .£ $^ $" YERROR MESSAGE: •8 FI•
0020 0035  D5E5C1D3 C9C440C4 C1E3C17A 40D7C1D9 E3D5D64C 5C4CE6C8 E2D5D64C C4E4E2E3  •NVALID DATA: PARTNC & WHSNC MUST•
CC40 0015  4CC2C540 D5E4D4C5 D9C9C33C 5D6B401C FC115CF9 13CCCCCO                    • BE NUMERIC ),  C •9            •

                                                                                      AID=7D CURSCR=5CF9 (24,1C)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
01   •_TPL1CENTER TRANSACTION CODE                                                   •
C2   •                                                                               •
C3   •                     QENTER DATA:                                              •
04   •                                                                               •
C5   •     YPART NO:£1234XQ                                                          •
C6   •     YWHS NO:£20YQ                                                             •
07   •                                                                               •
C8   •                     QSTOCK STATUS:                                            •
09   •                                                                               •
10   •     CDESCRIPTION:_                                                    Q       •
11   •     CORDER UNITS:_      C               QPRICE:_         C                    •
12   •                                                                               •
13   •                     QSTOCK STATUS AT WAREHOUSE:                               •
14   •                                                                               •
15   •     CLOCATION:_                         Q                                     •
16   •     CON HAND:_         Q                CAS OF:_         Q                    •
17   •     CON ORDER:_         C               QAS OF:_         C                    •
18   •                                                                               •
19   •                                                                               •
20   •                                                                               •
21   •                                                                               •
22   •                                                                               •
23   •                                    YERROR MESSAGE:                            •
24   •        UINVALID DATA: PARTNO & WHSNC MUST BE NUMERIC      C                   •
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8


        ATTRIBUTE CHAR DECODING:
            (40) =    UNP,ALP,DIS/NCT
          ⊦ (C8) =    UNP,ALP,IDS/DET
          £ (50) =    UNP,NUM,DIS/NDT
          Y (E8) =    PRG,ALP,IDS/DET
          C (F0) =    PRO,NUM,DIS/NDT
```

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



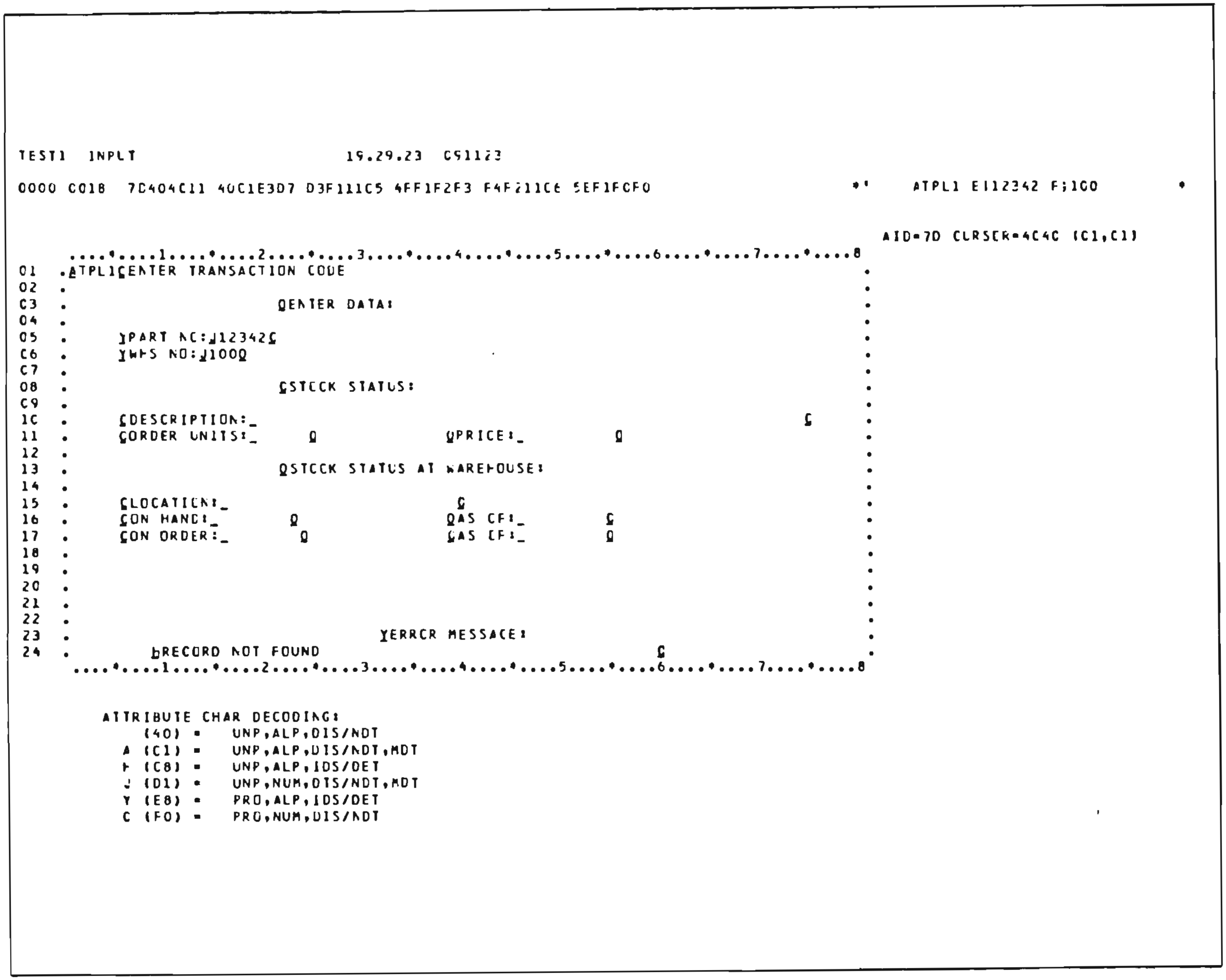
```
TEST1  INPUT                     19.29.19  091123

0000 0018  7D404C11 40C1E3D7 D3F111C5 4FF1F2F3 F4F911C6 5EF1F0F0                     *'    ATPL1 E|12349 F;100        *

                                                                                        AID=7D CURSOR=4C4C (01,01)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
01  .ATPL1CENTER TRANSACTION CODE                                                    .
02  .                                                                                .
03  .                    CENTER DATA:                                                .
04  .                                                                                .
05  .    YPART NO:J12349C                                                            .
06  .    YWHS NO:J100Q                                                               .
07  .                                                                                .
08  .                    QSTOCK STATUS:                                              .
09  .                                                                                .
10  .    CDESCRIPTION:_                                                        C     .
11  .    CORDER UNITS:_     Q             QPRICE:_         C                         .
12  .                                                                                .
13  .                    QSTOCK STATUS AT WAREHOUSE:                                 .
14  .                                                                                .
15  .    QLOCATION:_                        Q                                        .
16  .    CON HAND:_       C               QAS OF:_        C                          .
17  .    CON ORDER:_       Q              QAS OF:_        C                          .
18  .                                                                                .
19  .                                                                                .
20  .                                                                                .
21  .                                                                                .
22  .                                                                                .
23  .                              YERROR MESSAGE:                                   .
24  .       EINVALID DATA: PARTNO & WHSNO MUST BE NUMERIC       Q                    .
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8

        ATTRIBUTE CHAR DECODING:
            (40) =   UNP,ALP,DIS/NDT
          A (C1) =   UNP,ALP,DIS/NDT,MDT
          F (C8) =   UNP,ALP,IDS/DET
          J (D1) =   UNP,NUM,DIS/NDT,MDT
          Y (E8) =   PRU,ALP,IDS/DET
          C (F0) =   PRU,NUM,DIS/NDT
```

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



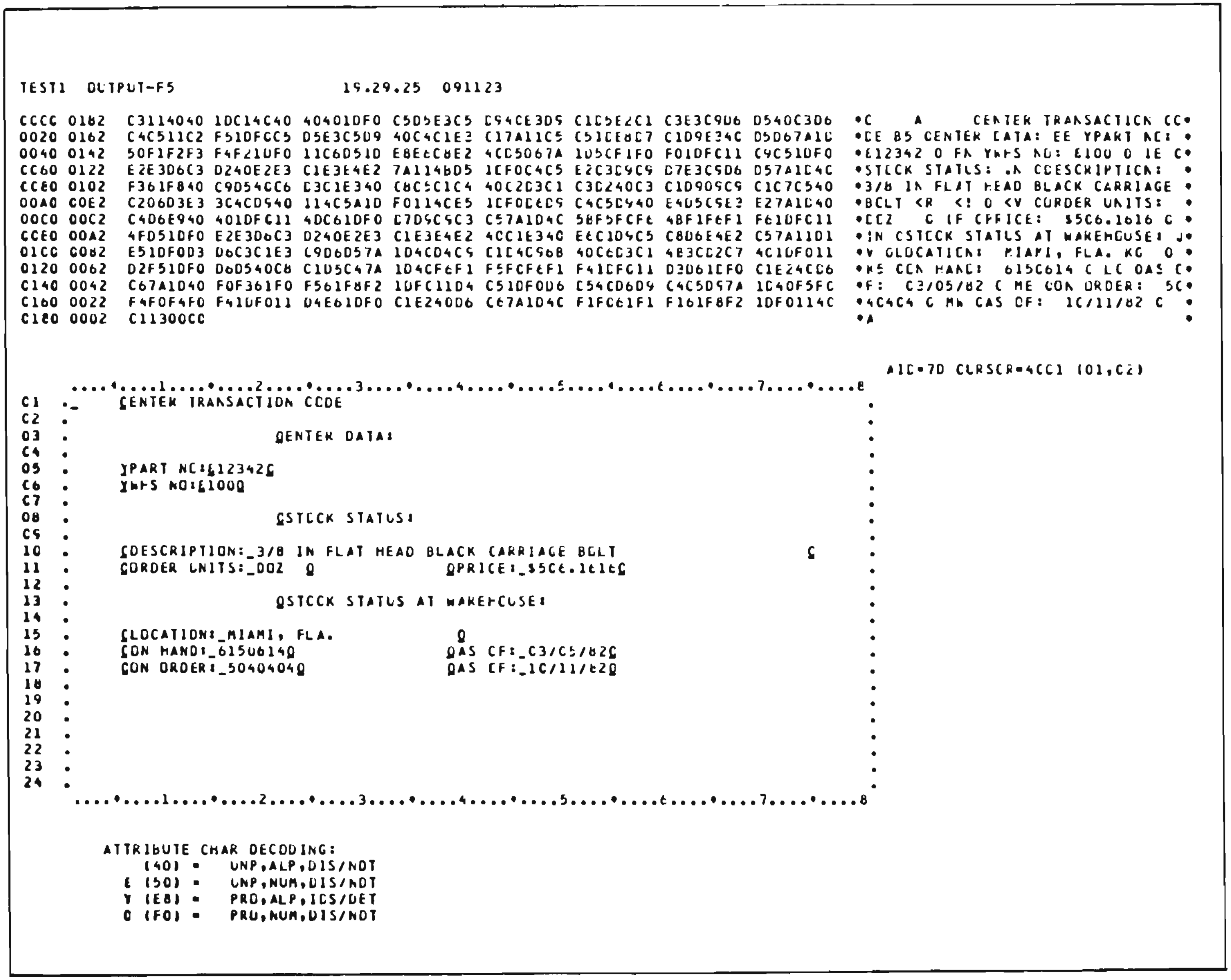
```
TEST1  OUTPUT-F1                   19.29.20  091123

CCCO 0039  C3114B50 125b5F11 5B7F1CE8 C5D9D9D6 D940D4C5 E2E2C1C7 C57A115C F81DC8D9  *C .£ $^ $" YERRCR MESSAGE: *6 HR*
CC2C C019  C5C3D6C9 C44CD5D6 E34CC6D6 E4D5C43C 5C£B4C1C FC115CF9 13000C0C          *ECCRC NCT FOUND ), 0 *9        *


                                                                                             AID=7D CLRSCR=5CF9 (24,1C)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
C1   ._TPL1ÇENTER TRANSACTION CODE                                                    .
C2   .                                                                                .
03   .                      ÇENTER DATA:                                              .
C4   .                                                                                .
05   .       Y̲PART NC:£12349Ç                                                         .
06   .       Y̲WHS NO:£100Ǫ                                                            .
C7   .                                                                                .
C8   .                      ǪSTCCK STATLS:                                            .
09   .                                                                                .
10   .       ÇDESCRIPTION:_                                                    Ç      .
11   .       ÇORDER UNITS:_      Ǫ                ǪPRICE:_          Ç                 .
12   .                                                                                .
13   .                      ÇSTCCK STATLS AT WAREHCUSE:                               .
14   .                                                                                .
15   .       ÇLOCATICN:_                       Ǫ                                      .
16   .       ÇON HAND:_        Ǫ              ÇAS CF:_          Ç                     .
17   .       ÇON ORDER:_        Ǫ             ǪAS CF:_          Ç                     .
18   .                                                                                .
19   .                                                                                .
20   .                                                                                .
21   .                                                                                .
22   .                                                                                .
23   .                              Y̲ERROR MESSAGE:                                   .
24   .          ḇRECORD NOT FOUND                                   Ç                 .
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8


      ATTRIBUTE CHAR DECODING:
          (40) =    UNP,ALP,DIS/NDT
        ⊦ (C8) =    UNP,ALP,IDS/DET
        £ (50) =    UNP,NUM,DIS/NDT
        Y (E8) =    PRO,ALP,ICS/DET
        C (F0) =    PRU,NUM,DIS/NDT
```

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



```
TEST1  INPLT                        19.29.23  C91123

0000 C018  7C404C11 40C1E3D7 D3F111C5 4FF1F2F3 F4F211C6 5EF1FCFO                 *'    ATPL1 E112342 F;1CO        *

                                                                                      AID=7D CLRSCR=4C4C (C1,C1)
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
01  .ATPL1CENTER TRANSACTION CODE                                                    .
02  .                                                                                .
C3  .                     QENTER DATA:                                               .
04  .                                                                                .
05  .    YPART NC:J12342C                                                            .
C6  .    YWFS NO:J100Q                                                               .
C7  .                                                                                .
08  .                     CSTCCK STATUS:                                             .
C9  .                                                                                .
1C  .    CDESCRIPTION:_                                                         C    .
11  .    CORDER UNITS:_      Q             QPRICE:_         Q                        .
12  .                                                                                .
13  .                     QSTCCK STATUS AT WAREHOUSE:                                .
14  .                                                                                .
15  .    CLOCATICN:_                        C                                        .
16  .    CON HAND:_        Q               QAS CF:_        C                         .
17  .    CON ORDER:_        Q              CAS CF:_        Q                         .
18  .                                                                                .
19  .                                                                                .
20  .                                                                                .
21  .                                                                                .
22  .                                                                                .
23  .                               YERRCR MESSACE:                                  .
24  .        bRECORD NOT FOUND                                  C                    .
     ....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8


        ATTRIBUTE CHAR DECODING:
            (40) =   UNP,ALP,DIS/NDT
          A (C1) =   UNP,ALP,DIS/NDT,MDT
          F (C8) =   UNP,ALP,IDS/DET
          J (D1) =   UNP,NUM,DIS/NDT,MDT
          Y (E8) =   PRO,ALP,IDS/DET
          C (F0) =   PRO,NUM,DIS/NDT
```

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 19 of 22)



```
TESTl  OUTPUT-F5                   19.29.25  091123

CCCC 0182  C3114040 1DC14C40 40401DF0 C5D5E3C5 C94CE3D9 C1C5E2C1 C3E3C9U6 D540C3D6   •C    A      CENTER TRANSACTICN CC•
0020 0162  C4C511C2 F51DFCC5 D5E3C5D9 40C4C1E3 C17A11C5 C51CE8C7 C1D9E34C D5D67A1C   •CE 85 CENTER CATA: EE YPART NC: •
0040 0142  50F1F2F3 F4F21UF0 11C6D51D E8E6C8E2 4CD5067A 1U5CF1F0 F01DFC11 C9C51DF0   •£12342 0 FN YHFS NU: £100 0 1E C•
CC60 0122  E2E3D6C3 D240E2E3 C1E3E4E2 7A114BD5 1CF0C4C5 E2C3C9C9 D7E3C9D6 D57A1C4C   •STCCK STATUS: .N CDESCRIPTICN:  •
CC80 0102  F361F840 C9D54CC6 C3C1E340 C8C5C1C4 40C2C3C1 C3C240C3 C1D909C9 C1C7C540   •3/8 IN FLAT HEAD BLACK CARRIAGE •
00A0 C0E2  C206D3E3 3C4CD940 114C5A1D F0114CE5 1CF0C6C9 C4C5C940 E4D5C9E2 E27A1C40   •BCLT <R  <! 0 <V CORDER UNITS:  •
00C0 00C2  C4D6E940 401DFC11 4DC61DF0 C7D9C9C3 C57A1D4C 58F5FCF6 4BF1F6F1 F61DFC11   •CCZ   C (F CFFICE:  $5C6.1616 C •
CCE0 00A2  4FD51DF0 E2E3D6C3 D240E2E3 C1E3E4E2 4CC1E340 E6C1D9C5 C8D6E4E2 C57A11D1   •¦N CSTCCK STATLS AT WAREHOUSE: J•
01CC C082  E51DF0D3 D6C3C1E3 C9D6D57A 1D4CD4C9 C1C4C96B 40C6C3C1 4B3CC2C7 4C1DF011   •V GLOCATICN:  MIAMI, FLA. KG  0 •
0120 0062  D2F51DF0 D6D540C8 C1D5C47A 1D4CF6F1 F5FCF6F1 F41CFC11 D3D61CF0 C1E24CC6   •K5 CCN HANC:  615C614 C LC OAS C•
C140 0042  C67A1D40 F0F361F0 F561F8F2 1DFC11D4 C51DF0D6 C54CD6D9 C4C5D97A 1C40F5FC   •F:  C3/05/82 C ME CON ORDER:  5C•
C160 0022  F4F0F4F0 F41DF011 D4E61DF0 C1E240D6 C67A1D4C F1FC61F1 F161F8F2 1DF0114C   •4C4C4 C MW CAS CF:  1C/11/82 C  •
C180 0002  C11300CC                                                                   •A                              •

                                                                                     AIC=7D CURSCR=4CC1 (01,C2)
     ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8
C1  ._     CENTER TRANSACTION CCDE                                                    .
C2  .                                                                                 .
03  .                         QENTER DATA:                                            .
C4  .                                                                                 .
05  .      YPART NC:£12342C                                                           .
C6  .      YHFS NO:£100Q                                                              .
C7  .                                                                                 .
08  .                         CSTCCK STATUS:                                          .
C9  .                                                                                 .
10  .      CDESCRIPTION:_3/8 IN FLAT HEAD BLACK CARRIAGE BCLT                    C    .
11  .      CORDER UNITS:_002  Q                 QPRICE:_$5C6.1616C                    .
12  .                                                                                 .
13  .                         QSTCCK STATUS AT WAREHCUSE:                             .
14  .                                                                                 .
15  .      CLOCATION:_MIAMI, FLA.                Q                                    .
16  .      CON HAND:_6150614Q                   QAS CF:_C3/C5/82C                     .
17  .      CON ORDER:_5040404Q                  QAS CF:_1C/11/82Q                     .
18  .                                                                                 .
19  .                                                                                 .
20  .                                                                                 .
21  .                                                                                 .
22  .                                                                                 .
23  .                                                                                 .
24  .                                                                                 .
     ....•....1....•....2....•....3....•....4....•....5....•....6....•....7....•....8


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

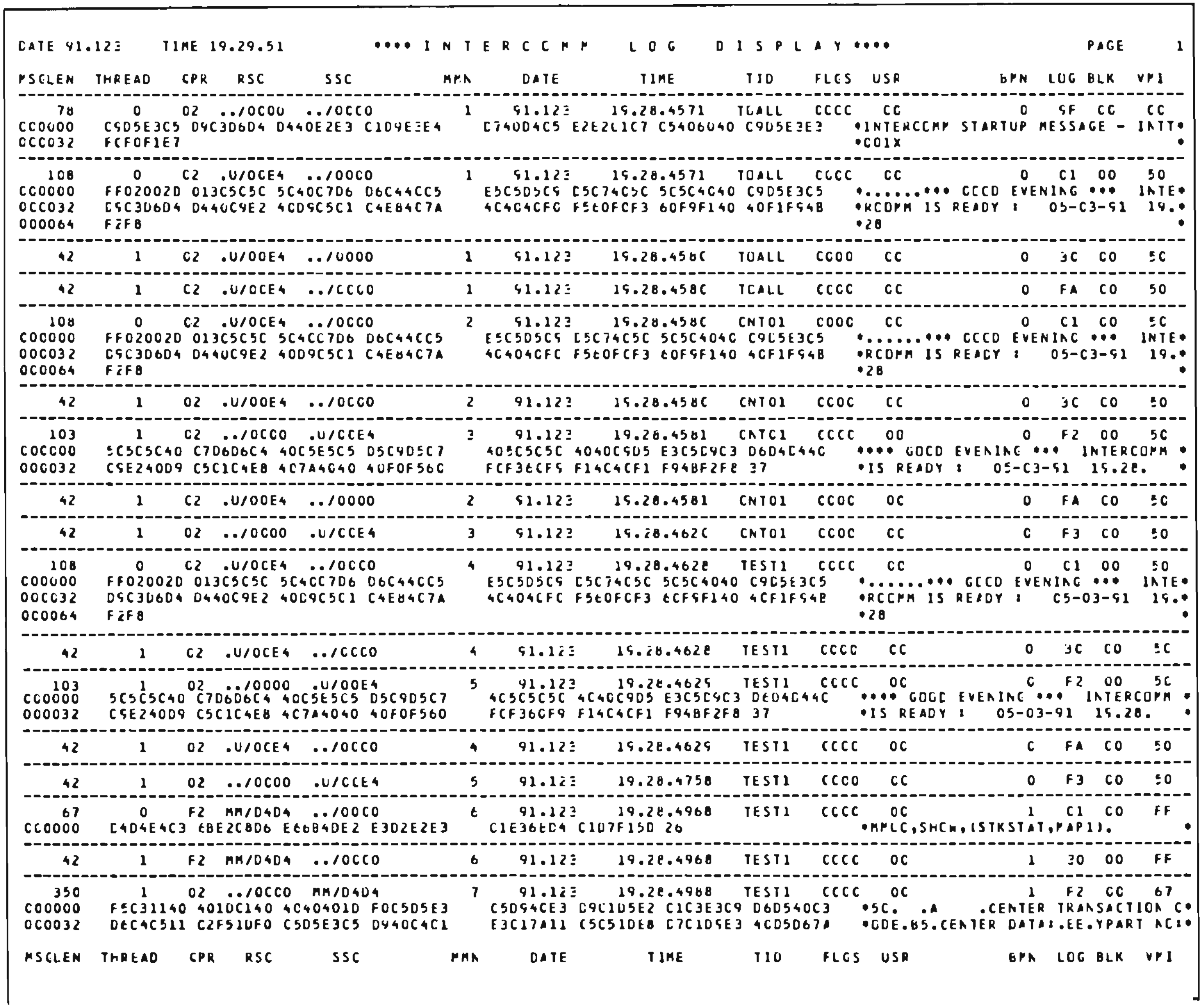
Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 20 of 22)

THIS PAGE INTENTIONALLY UNUSED

Figure 54. SIM3270 Printout from Simulation Mode Execution (Page 21 of 22)

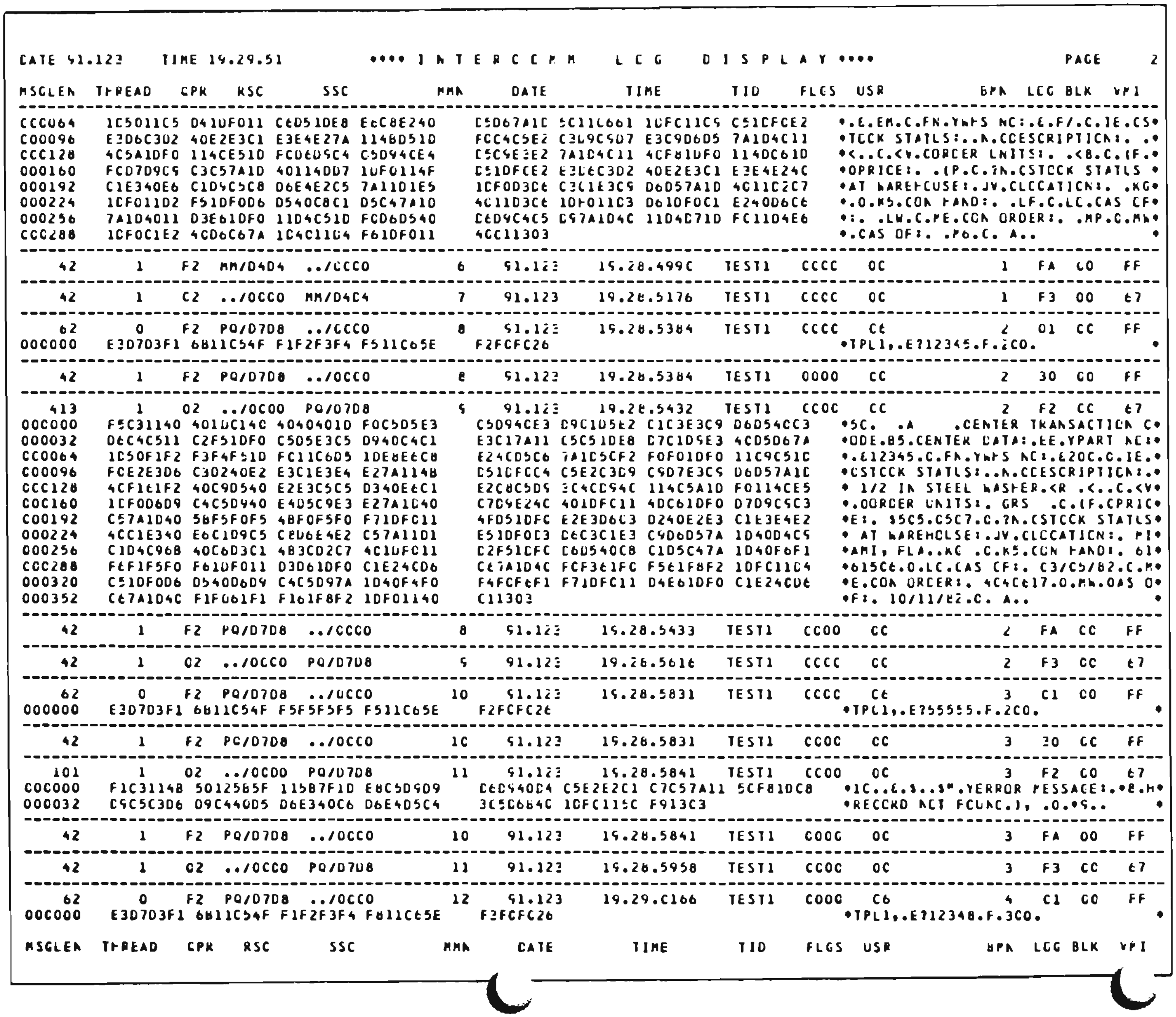
THIS PAGE INTENTIONALLY UNUSED

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



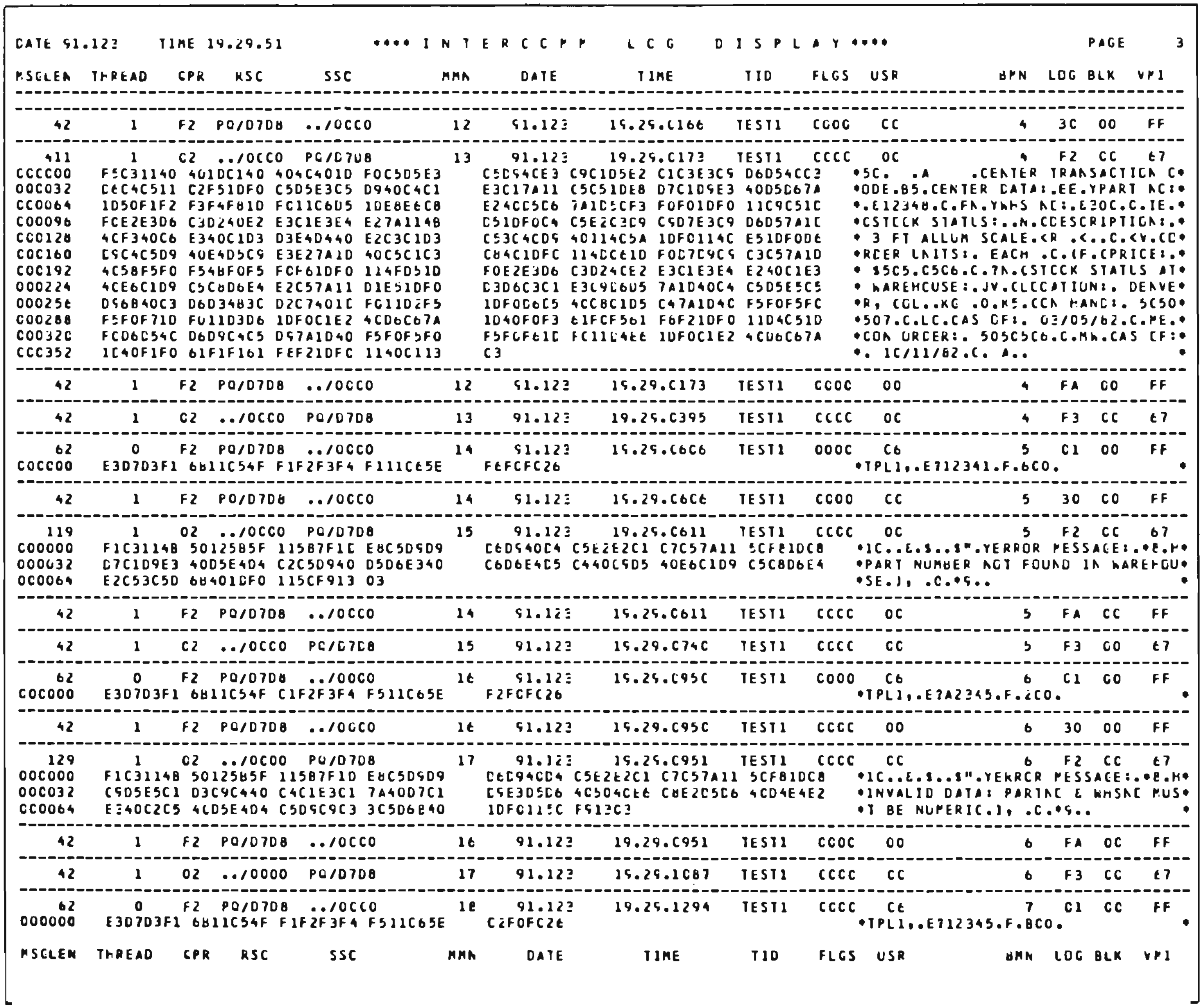
```
DATE 91.123    TIME 19.29.51         **** I N T E R C O M M   L O G   D I S P L A Y ****                PAGE     1

MSGLEN  THREAD   CPR   RSC        SSC           MMN      DATE          TIME         TID      FLGS  USR             BPN  LOG BLK   VPI
-----------------------------------------------------------------------------------------------------------------------------------
   78       0     02  ../0C00   ../0CC0           1    91.123     19.28.4571    TOALL    CCCC    CC                  0   9F  CC    CC
CC0000   C9D5E3C5 D9C3D6D4 D440E2E3 C1D9E3E4    C740D4C5 E2E2C1C7 C5406040 C9D5E3E3   *INTERCOMM STARTUP MESSAGE - INTT*
0CC032   FCF0F1E7                                                                     *C01X                             *
-----------------------------------------------------------------------------------------------------------------------------------
  108       0     C2  .U/0CE4   ../00C0           1    91.123     19.28.4571    TOALL    CCCC    CC                  0   C1  00    50
CC0000   FF02002D 013C5C5C 5C40C7D6 D6C44CC5    E5C5D5C9 C5C74C5C 5C5C4C40 C9D5E3C5   *.......*** GOOD EVENING ***  INTE*
0CC032   D9C3D6D4 D440C9E2 40D9C5C1 C4E84C7A    4C4C4CFC F5C0FCF3 60F9F140 40F1F94B   *RCOMM IS READY :   05-03-91  19.*
000064   F2F8                                                                         *28                               *
-----------------------------------------------------------------------------------------------------------------------------------
   42       1     C2  .U/00E4   ../0000           1    91.123     19.28.458C    TOALL    C000    CC                  0   3C  C0    5C
-----------------------------------------------------------------------------------------------------------------------------------
   42       1     C2  .U/0CE4   ../CCC0           1    91.123     19.28.458C    TCALL    CCCC    CC                  0   FA  C0    50
-----------------------------------------------------------------------------------------------------------------------------------
  108       0     C2  .U/0CE4   ../0CC0           2    91.123     19.28.458C    CNT01    C00C    CC                  0   C1  C0    5C
C00000   FF02002D 013C5C5C 5C4CC7D6 D6C44CC5    E5C5D5C9 C5C74C5C 5C5C404C C9C5E3C5   *.......*** GOOD EVENING ***  INTE*
00C032   D9C3D6D4 D440C9E2 40D9C5C1 C4E84C7A    4C404CFC F5C0FCF3 C0F9F140 4CF1F94B   *RCOMM IS READY :   05-03-91  19.*
0C0064   F2F8                                                                         *28                               *
-----------------------------------------------------------------------------------------------------------------------------------
   42       1     02  .U/00E4   ../0CC0           2    91.123     19.28.458C    CNT01    CC0C    CC                  0   3C  C0    50
-----------------------------------------------------------------------------------------------------------------------------------
  103       1     C2  ../0CC0   .U/CCE4           3    91.123     19.28.4581    CNTC1    CCCC    00                  0   F2  00    5C
C0CC00   5C5C5C40 C7D6D6C4 40C5E5C5 D5C9D5C7    405C5C5C 4040C9D5 E3C5C9C3 D6D4C44C   **** GOOD EVENING ***  INTERCOMM *
000032   C9E240D9 C5C1C4E8 4C7A4040 40F0F56C    FCF36CF9 F14C4CF1 F94BF2F8 37          *IS READY :   05-03-91  19.28.   *
-----------------------------------------------------------------------------------------------------------------------------------
   42       1     C2  .U/00E4   ../0000           2    91.123     19.28.4581    CNT01    CC0C    0C                  0   FA  C0    5C
-----------------------------------------------------------------------------------------------------------------------------------
   42       1     02  ../0C00   .U/CCE4           3    91.123     19.28.462C    CNT01    CC0C    CC                  C   F3  C0    50
-----------------------------------------------------------------------------------------------------------------------------------
  108       0     C2  .U/0CE4   ../0CC0           4    91.123     19.28.4628    TEST1    CCCC    CC                  0   C1  00    50
C00000   FF02002D 013C5C5C 5C4CC7D6 D6C44CC5    E5C5D5C9 C5C74C5C 5C5C4040 C9C5E3C5   *.......*** GOOD EVENING ***  INTE*
00C032   D9C3D6D4 D440C9E2 40C9C5C1 C4E84C7A    4C404CFC F5C0FCF3 6CF9F140 4CF1F94B   *RCOMM IS READY :   C5-03-91  19.*
0C0064   F2F8                                                                         *28                               *
-----------------------------------------------------------------------------------------------------------------------------------
   42       1     C2  .U/0CE4   ../CCC0           4    91.123     19.28.4628    TEST1    CCCC    CC                  0   3C  C0    5C
-----------------------------------------------------------------------------------------------------------------------------------
  103       1     02  ../0000   .U/00E4           5    91.123     19.28.4629    TEST1    CCCC    0C                  C   F2  00    5C
CC0000   5C5C5C40 C7D6D6C4 40C5E5C5 D5C9D5C7    4C5C5C5C 4C40C9D5 E3C5C9C3 D6D4C44C   **** GOOD EVENING ***  INTERCOMM *
000032   C9E240D9 C5C1C4E8 4C7A4040 40F0F560    FCF36CF9 F14C4CF1 F948F2F8 37          *IS READY :   05-03-91  19.28.   *
-----------------------------------------------------------------------------------------------------------------------------------
   42       1     02  .U/0CE4   ../0CC0           4    91.123     19.28.4629    TEST1    CCCC    0C                  C   FA  C0    50
-----------------------------------------------------------------------------------------------------------------------------------
   42       1     02  ../0C00   .U/CCE4           5    91.123     19.28.4758    TEST1    CCC0    CC                  0   F3  C0    50
-----------------------------------------------------------------------------------------------------------------------------------
   67       0     F2  MM/D4D4   ../00C0           6    91.123     19.28.4968    TEST1    CCCC    0C                  1   C1  C0    FF
CC0000   C4D4E4C3 6BE2C8D6 E66B4DE2 E3D2E2E3    C1E36BD4 C1D7F15D 26                   *MPLC,SHOW,(STKSTAT,MAP1).       *
-----------------------------------------------------------------------------------------------------------------------------------
   42       1     F2  MM/D4D4   ../0CC0           6    91.123     19.28.4968    TEST1    CCCC    0C                  1   30  00    FF
-----------------------------------------------------------------------------------------------------------------------------------
  350       1     02  ../0CC0   MM/D4D4           7    91.123     19.28.4988    TEST1    CCCC    0C                  1   F2  CC    67
C00000   F5C31140 401DC140 4C40401D F0C5D5E3    C5D94CE3 C9C1D5E2 C1C3E3C9 D6D540C3   *5C.  .A    .CENTER TRANSACTION C*
0C0032   D6C4C511 C2F51DF0 C5D5E3C5 D940C4C1    E3C17A11 C5C51DE8 C7C1D9E3 4CD5D67A   *ODE.B5.CENTER DATA:.EE.YPART NC:*

MSGLEN  THREAD   CPR   RSC        SSC           MMN      DATE          TIME         TID      FLGS  USR             BPN  LOG BLK   VPI
```

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



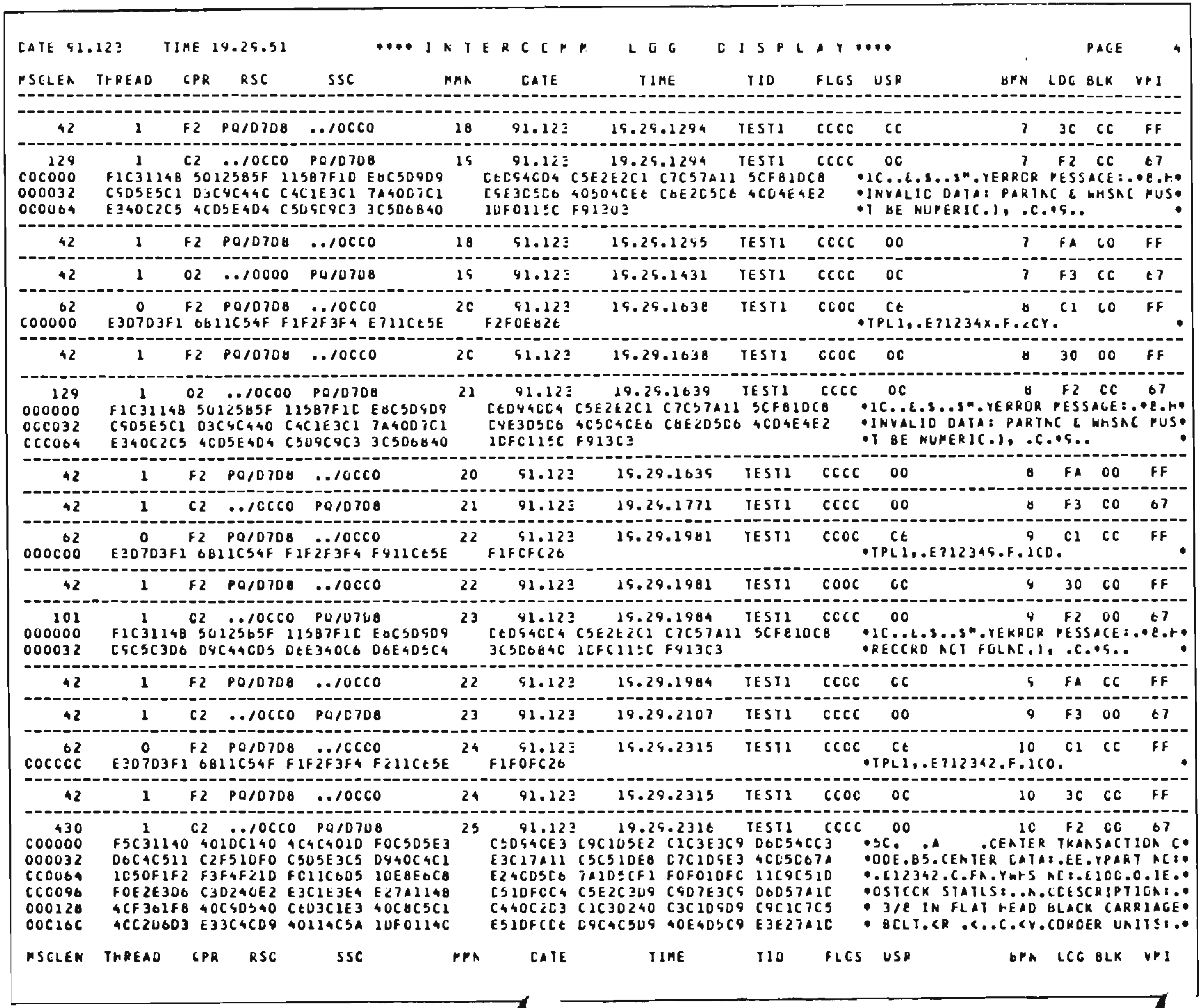
```
DATE 91.123    TIME 19.29.51         **** I N T E R C O M M    L O G    D I S P L A Y ****                PAGE     2

MSGLEN  THREAD   CPR   RSC        SSC          MMN      DATE          TIME        TID     FLGS  USR            BPN  LOG BLK  VPI
-------------------------------------------------------------------------------------------------------------------------------
CCC064   1C5011C5 D41DF011 C6D51DE8 E6C8E240    C5D67A1C 5C11C661 1DFC11C9 C51CFCE2   *.&.EM.C.FN.YNFS NO:.&.F/.C.IE.CS*
C00096   E3D6C3D2 40E2E3C1 E3E4E27A 114BD51D    FCC4C5E2 C3L9C9D7 E3C9D6D5 7A1D4C11   *TOCK STATUS:..N.CDESCRIPTION:. .*
CCC128   4C5A1DF0 114CE51D FCD6D9C4 C5D94CE4    C5C9E3E2 7A1D4C11 4CF81DF0 114DC61D   *<..C.<V.CORDER UNITS:. .<8.C.(F.*
000160   FCD7D9C9 C3C57A1D 40114DD7 1DF0114F    C51DFCE2 E3D6C3D2 40E2E3C1 E3E4E24C   *OPRICE:. .(P.C.?N.CSTOCK STATUS *
000192   C1E340E6 C1D9C5C8 D6E4E2C5 7A11D1E5    1CF0D3C6 C3C1E3C9 D6D57A1D 4C11C2C7   *AT WAREHOUSE:.JV.CLOCATION:. .KG*
000224   1CF011D2 F51DF0D6 D540C8C1 D5C47A1D    4C11D3C6 1DF011C3 D61DF0C1 E240D6C6   *.O.K5.CON HAND:. .LF.C.LC.CAS CF*
000256   7A1D4011 D3E61DF0 11D4C51D FCD6D540    C6D9C4C5 C97A1D4C 11D4C71D FC11D4E6   *:. .LW.C.ME.CON ORDER:. .MP.C.MW*
CCC288   1CF0C1E2 4CD6C67A 1C4C11C4 F61DF011    4CC11303                              *.CAS OF:. .M6.C. A..            *
-------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  MM/D4D4  ../CCCO          6    91.123     19.28.499C   TEST1   CCCC   OC              1   FA  C0   FF
-------------------------------------------------------------------------------------------------------------------------------
    42       1    C2  ../OCCO  MM/D4C4          7    91.123     19.28.5176   TEST1   CCCC   OC              1   F3  00   67
-------------------------------------------------------------------------------------------------------------------------------
    62       0    F2  PQ/D7D8  ../CCCO          8    91.123     19.28.5384   TEST1   CCCC   C6              2   01  CC   FF
000000   E3D7D3F1 6B11C54F F1F2F3F4 F511C65E    F2FCFC26                              *TPL1,.E712345.F.2CO.           *
-------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  PQ/D7D8  ../OCCO          8    91.123     19.28.5384   TEST1   0000   CC              2   30  C0   FF
-------------------------------------------------------------------------------------------------------------------------------
   413       1    02  ../0C00  PQ/07D8          9    91.123     19.28.5432   TEST1   CCOC   CC              2   F2  CC   67
00C000   F5C31140 401DC14C 4040401D F0C5D5E3    C5D94CE3 D9C1D5E2 C1C3E3C9 D6D54CC3   *5C.  .A     .CENTER TRANSACTION C*
000032   D6C4C511 C2F51DF0 C5D5E3C5 D940C4C1    E3C17A11 C5C51DE8 D7C1D9E3 4CD5D67A   *ODE.B5.CENTER DATA:.EE.YPART NO:*
CC0064   1D50F1F2 F3F4F51D FC11C6D5 1DE8E6C8    E24CD5C6 7A1C5CF2 F0F01DF0 11C9C51C   *.&12345.C.FN.YNFS NO:.&20C.C.IE.*
C00096   FCE2E3D6 C3D240E2 E3C1E3E4 E27A114B    C51CFCC4 C5E2C3D9 C9D7E3C9 D6D57A1C   *CSTOCK STATUS:..N.CDESCRIPTION:.*
CCC128   4CF161F2 40C9D540 E2E3C5C5 D340E6C1    E2C8C5D9 3C4CD94C 114C5A1D F0114CE5   * 1/2 IN STEEL WASHER.<R .<..C.<V*
C0C160   1CF006D9 C4C5D940 E4D5C9E3 E27A1C40    C7D9E24C 401DFC11 4DC61DF0 D7D9C9C3   *.OORDER UNITS:. GRS  .C.(F.CPRIC*
C00192   C57A1D40 5BF5F0F5 4BF0F5F0 F71DFC11    4FD51DFC E2E3D6C3 D240E2E3 C1E3E4E2   *E:. $5C5.C5C7.C.?N.CSTOCK STATUS*
000224   4CC1E340 E6C1D9C5 C8D6E4E2 C57A11D1    E51DF0C3 D6C3C1E3 C9D6D57A 1D40D4C9   * AT WAREHOUSE:.JV.CLOCATION:. MI*
000256   C1D4C96B 40C6D3C1 4B3CD2C7 4C1DFC11    C2F51CFC C6D540C8 C1D5C47A 1D40F6F1   *AMI, FLA..KG .C.K5.CON HAND:. 61*
CCC288   F6F1F5F0 F61DF011 03D61DF0 C1E24CD6    C67A1D4C FCF361FC F561F8F2 1DFC11C4   *615C6.O.LC.CAS CF:. C3/C5/82.C.M*
000320   C51DF0D6 D540D6D9 C4C5D97A 1D40F4F0    F4FCF6F1 F71DFC11 D4E61DF0 C1E24CD6   *E.CON ORDER:. 4C4C617.O.MW.OAS O*
000352   C67A1D4C F1F061F1 F161F8F2 1DF01140    C11303                                *F:. 10/11/82.C. A..             *
-------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  PQ/D7D8  ../CCCO          8    91.123     19.28.5433   TEST1   CC00   CC              2   FA  CC   FF
-------------------------------------------------------------------------------------------------------------------------------
    42       1    02  ../OCCO  PQ/D7D8          9    91.123     19.28.5616   TEST1   CCCC   CC              2   F3  CC   67
-------------------------------------------------------------------------------------------------------------------------------
    62       0    F2  PQ/D7D8  ../UCCO         10    91.123     19.28.5831   TEST1   CCCC   C6              3   C1  C0   FF
000000   E3D7D3F1 6B11C54F F5F5F5F5 F511C65E    F2FCFC26                              *TPL1,.E755555.F.2CO.           *
-------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  PQ/D7D8  ../OCCO         1C    91.123     19.28.5831   TEST1   CCOC   CC              3   30  CC   FF
-------------------------------------------------------------------------------------------------------------------------------
   101       1    02  ../0C00  PQ/D7D8         11    91.123     19.28.5841   TEST1   CC00   OC              3   F2  C0   67
C0C000   F1C3114B 501265F 115B7F1D E8C5D909     C6C940C4 C5E2E2C1 C7C57A11 5CF81DC8   *1C..&.$..$".YERROR MESSAGE:.*8.H*
000032   C9C5C3D6 09C440D5 D6E340C6 D6E4D5C4    3C5C6B4C 1DFC115C F913C3              *RECORD NOT FOUND.), .O.*9..     *
-------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  PQ/D7D8  ../0CCO         10    91.123     19.28.5841   TEST1   C00C   OC              3   FA  00   FF
-------------------------------------------------------------------------------------------------------------------------------
    42       1    02  ../0CCO  PQ/D7D8         11    91.123     19.28.5958   TEST1   CCOC   OC              3   F3  CC   67
-------------------------------------------------------------------------------------------------------------------------------
    62       0    F2  PQ/D7D8  ../0CCO         12    91.123     19.29.C166   TEST1   C00C   C6              4   C1  C0   FF
00C000   E3D7D3F1 6B11C54F F1F2F3F4 F811C65E    F2FCFC26                              *TPL1,.E712348.F.3CO.           *

MSGLEN  THREAD   CPR   RSC        SSC          MMN      DATE          TIME        TID     FLGS  USR            BPN  LOG BLK  VPI
```

Figure 55. Simulation Mode Execution Log Printout (Page 2 of 6)



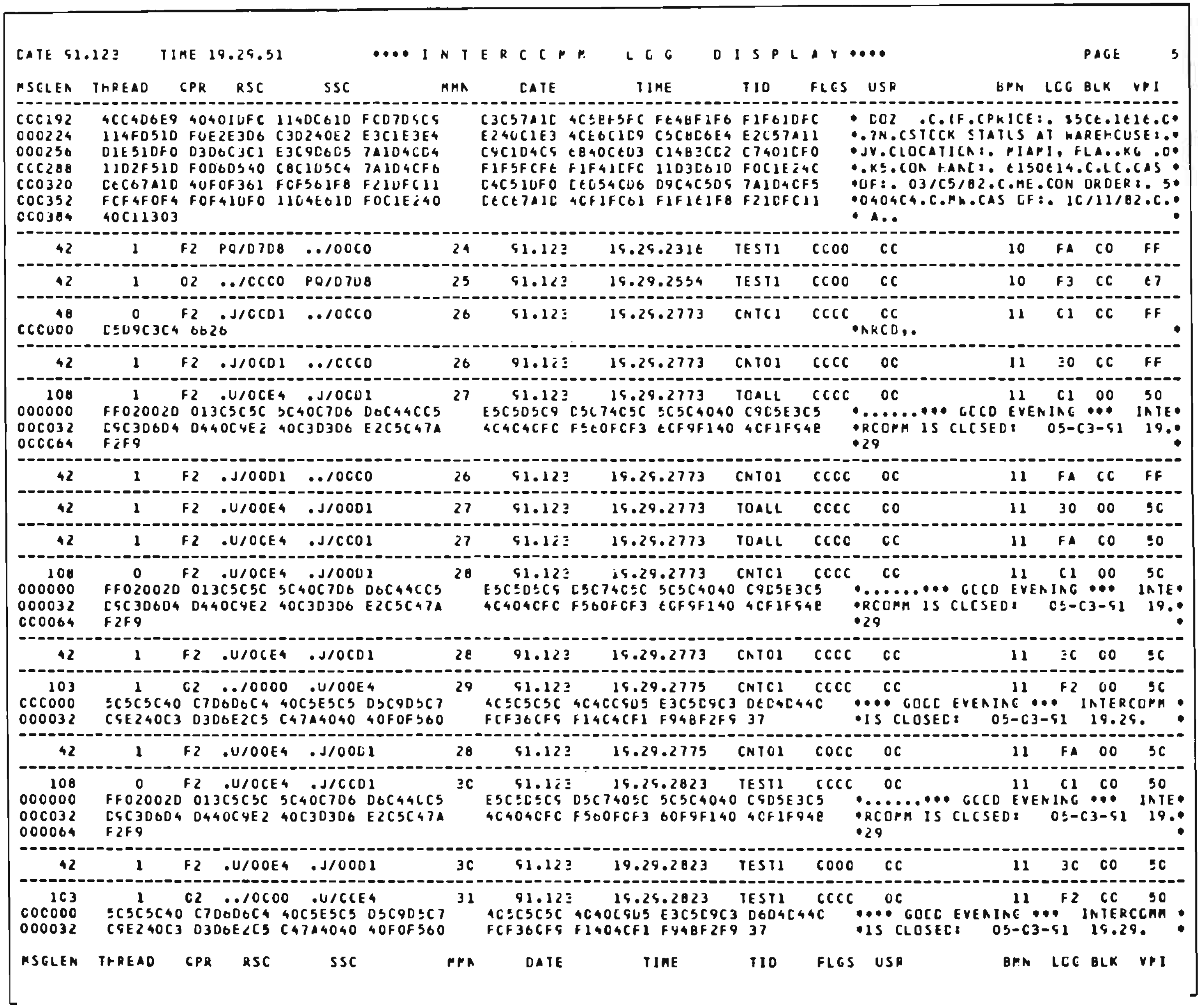
```
DATE 91.123    TIME 19.29.51         **** I N T E R C C P P   L O G   D I S P L A Y ****              PAGE     3
MSGLEN  THREAD   CPR   RSC        SSC            MMN      DATE          TIME         TID      FLGS  USR              BMN  LOG BLK  VPI
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8   ../0CC0           12     91.123      19.29.C166     TEST1    C00C   CC                 4   3C  00    FF
-----------------------------------------------------------------------------------------------------------------------------------
   411      1     C2  ../0CC0   PQ/D7D8           13     91.123      19.29.C173     TEST1    CCCC   0C                 4   F2  CC    67
CCCC00   F5C31140 401DC140 404C401D F0C5D5E3    C5D94CE3 C9C1D5E2 C1C3E3C9 D6D54CC3   *5C.   .A    .CENTER TRANSACTION C*
00C032   C6C4C511 C2F51DF0 C5D5E3C5 D940C4C1    E3C17A11 C5C51DE8 D7C1D9E3 40D5C67A   *ODE.B5.CENTER DATA:.EE.YPART NC:*
CC0064   1D50F1F2 F3F4F81D FC11C6D5 1DE8E6C8    E24CC5C6 7A1D5CF3 F0F01DF0 11C9C51C   *.E12348.C.FN.YWHS NC:.E30C.C.IE.*
C00096   FCE2E3D6 C3D240E2 E3C1E3E4 E27A114B    C51DF0C4 C5E2C3C9 C9D7E3C9 D6D57A1C   *CSTCCK STATUS:..N.CDESCRIPTION:.*
CC0128   4CF340C6 E340C1D3 D3E4D440 E2C3C1D3    C53C4CD9 40114C5A 1DF0114C E51DF0D6   * 3 FT ALLUM SCALE.<R .<..C.<V.CO*
C0C160   C9C4C5D9 40E4D5C9 E3E27A1D 40C5C1C3    C84C1DFC 114DC61D F0D7D9C9 C3C57A1D   *RDER UNITS:. EACH .C.(F.CPRICE:.*
C0C192   4C58F5F0 F54BF0F5 FCF61DF0 114FD51D    F0E2E3D6 C3D24CE2 E3C1E3E4 E240C1E3   * $5C5.C5C6.C.?N.CSTCCK STATUS AT*
000224   4CE6C1D9 C5C8D6E4 E2C57A11 D1E51DF0    C3D6C3C1 E3C9D6D5 7A1D40C4 C5D5E5C5   * WAREHOUSE:.JV.CLOCATION:. DENVE*
00025E   D96B40C3 D6D34B3C D2C7401C FC11D2F5    1DF0D6C5 4CC8C1D5 C47A1D4C F5F0F5FC   *R, COL..KG .0.K5.CON HAND:. 5C50*
C00288   F5F0F71D F011D3D6 1DF0C1E2 4CD6C67A    1D40F0F3 61FCF561 F6F21DF0 11D4C51D   *507.C.LC.CAS OF:. 03/05/62.C.ME.*
C0032C   FCD6C54C D6D9C4C5 D97A1D40 F5F0F5F0    F5FCF61C FC11D4E6 1DF0C1E2 4CD6C67A   *CON ORDER:. 505C5C6.C.MW.CAS OF:*
CCC352   1C40F1F0 61F1F161 F6F21DFC 1140C113    C3                                    *. 1C/11/82.C. A..               *
-----------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8   ../0CC0           12     91.122      19.29.C173     TEST1    C00C   00                 4   FA  00    FF
-----------------------------------------------------------------------------------------------------------------------------------
    42      1     C2  ../0CC0   PQ/D7D8           13     91.123      19.29.C395     TEST1    CCCC   0C                 4   F3  CC    67
-----------------------------------------------------------------------------------------------------------------------------------
    62      0     F2  PQ/D7D8   ../0CC0           14     91.123      19.29.C6C6     TEST1    000C   C6                 5   C1  00    FF
C0CC00   E3D7D3F1 6B11C54F F1F2F3F4 F111C65E    F6FCFC26                              *TPL1,.E712341.F.6C0.            *
-----------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8   ../0CC0           14     91.123      19.29.C6C6     TEST1    C000   CC                 5   30  C0    FF
-----------------------------------------------------------------------------------------------------------------------------------
   119      1     02  ../0CC0   PQ/D7D8           15     91.123      19.29.C611     TEST1    CCCC   0C                 5   F2  CC    67
C00000   F1C3114B 5012585F 115B7F1C E8C5D9D9    C6D940C4 C5E2E2C1 C7C57A11 5CF81DC8   *1C..E.$..$".YERROR MESSAGE:.*8.H*
000032   D7C1D9E3 40D5E4D4 C2C5D940 D5D6E340    C6D6E4C5 C440C9D5 40E6C1D9 C5C8D6E4   *PART NUMBER NOT FOUND IN WAREHOU*
0C0064   E2C53C5D 6B401DF0 115CF913 03                                                *SE.), .C.*9..                   *
-----------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8   ../0CC0           14     91.123      19.29.C611     TEST1    CCCC   0C                 5   FA  CC    FF
-----------------------------------------------------------------------------------------------------------------------------------
    42      1     C2  ../0CC0   PQ/D7D8           15     91.123      19.29.C74C     TEST1    CCCC   CC                 5   F3  C0    67
-----------------------------------------------------------------------------------------------------------------------------------
    62      0     F2  PQ/D7D8   ../00C0           16     91.123      19.29.C95C     TEST1    C000   C6                 6   C1  C0    FF
C0C000   E3D7D3F1 6B11C54F C1F2F3F4 F511C65E    F2FCFC26                              *TPL1,.E7A2345.F.2C0.            *
-----------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8   ../0CC0           16     91.123      19.29.C95C     TEST1    CCCC   00                 6   30  00    FF
-----------------------------------------------------------------------------------------------------------------------------------
   129      1     02  ../0C00   PQ/D7D8           17     91.123      19.29.C951     TEST1    CCCC   CC                 6   F2  CC    67
00C000   F1C3114B 50125B5F 115B7F1D E8C5D9D9    C6C94CD4 C5E2E2C1 C7C57A11 5CF81DC8   *1C..E.$..$".YERRCR MESSAGE:.*8.H*
00C032   C9D5E5C1 D3C9C440 C4C1E3C1 7A40D7C1    C9E3D5C6 4C504CE6 C8E2C5C6 4CD4E4E2   *INVALID DATA: PARTNC & WHSNC MUS*
CC0064   E340C2C5 4CD5E4D4 C5D9C9C3 3C5D6B40    1DF0115C F913C3                       *T BE NUMERIC.), .C.*9..         *
-----------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8   ../0CC0           16     91.123      19.29.C951     TEST1    C00C   00                 6   FA  0C    FF
-----------------------------------------------------------------------------------------------------------------------------------
    42      1     02  ../0000   PQ/D7D8           17     91.123      19.29.1C87     TEST1    CCCC   CC                 6   F3  CC    67
-----------------------------------------------------------------------------------------------------------------------------------
    62      0     F2  PQ/D7D8   ../0CC0           18     91.123      19.29.1294     TEST1    CCCC   C6                 7   C1  CC    FF
000000   E3D7D3F1 6B11C54F F1F2F3F4 F511C65E    C2F0FC26                              *TPL1,.E712345.F.BC0.            *

MSGLEN  THREAD   CPR   RSC        SSC            MMN      DATE          TIME         TID      FLGS  USR              BMN  LOG BLK  VPI
```

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



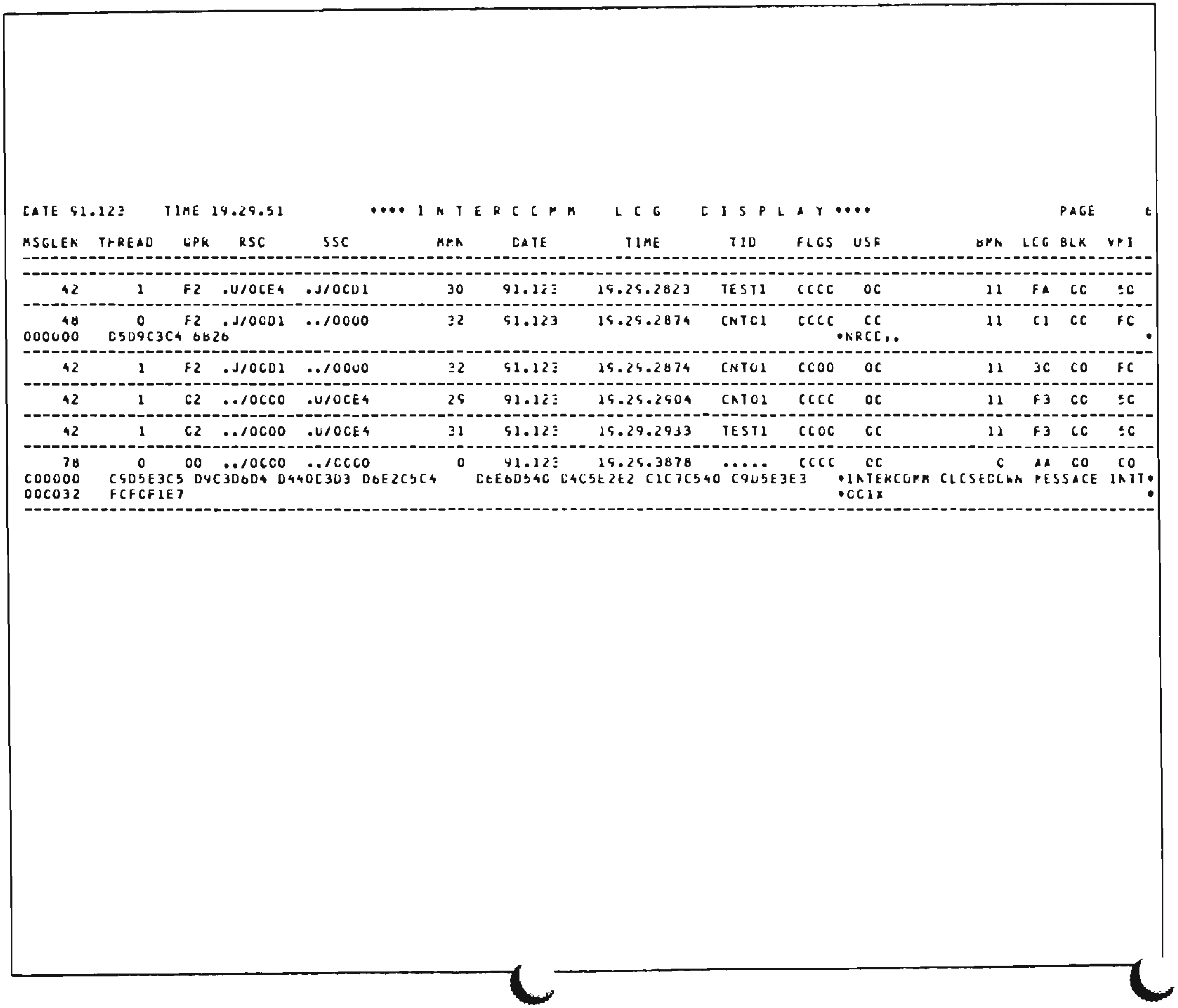
```
CATE 91.123     TIME 19.29.51          **** I N T E R C C M M    L O G    C I S P L A Y ****              PAGE     4

MSGLEN  THREAD   CPR   RSC       SSC            MMA     CATE          TIME          TID     FLGS  USR                BPN  LDG BLK  VPI
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8  ../0CC0          18     91.123      19.29.1294    TEST1    CCCC   CC                  7   3C  CC    FF
-------------------------------------------------------------------------------------------------------------------------------------
   129      1     C2  ../0CC0  PQ/D7D8          19     91.123      19.29.1294    TEST1    CCCC   0C                  7   F2  CC    67
C0C000    F1C3114B 5012585F 115B7F1D E8C5D9D9    C6C94CD4 C5E2E2C1 C7C57A11 5CF81DC8   *1C..&.$..$".YERROR MESSACE:.*8.H*
000032    C9D5E5C1 D3C9C44C C4C1E3C1 7A40D7C1    C9E3C5C6 40504CE6 C6E2C5C6 4CD4E4E2   *INVALIC DATA: PARTNC & WHSNC MUS*
0C0064    E340C2C5 4CD5E4D4 C5D9C9C3 3C5D6840    1DF0115C F91303                       *T BE NUMERIC.), .C.*9..          *
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8  ../0CC0          18     91.123      19.29.1295    TEST1    CCCC   00                  7   FA  C0    FF
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     02  ../0000  PQ/D7D8          19     91.123      19.29.1431    TEST1    CCCC   0C                  7   F3  CC    67
-------------------------------------------------------------------------------------------------------------------------------------
    62      0     F2  PQ/D7D8  ../0CC0          2C     91.123      19.29.1638    TEST1    CCOC   C6                  8   C1  C0    FF
C00000    E3D7D3F1 6B11C54F F1F2F3F4 E711C65E    F2F0E826                              *TPL1,.E71234X.F.<CY.             *
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8  ../0CC0          2C     91.123      19.29.1638    TEST1    CCOC   0C                  8   30  00    FF
-------------------------------------------------------------------------------------------------------------------------------------
   129      1     02  ../0C00  PQ/D7D8          21     91.123      19.29.1639    TEST1    CCCC   0C                  8   F2  CC    67
000000    F1C3114B 5012585F 115B7F1C E8C5D9D9    C6D94CD4 C5E2E2C1 C7C57A11 5CF81DC8   *1C..&.$..$".YERROR MESSAGE:.*8.H*
0CC032    C9D5E5C1 D3C9C440 C4C1E3C1 7A40D7C1    C9E3D5C6 4C5C4CE6 C8E2D5C6 4CD4E4E2   *INVALID DATA: PARTNC & WHSNC MUS*
CCC064    E340C2C5 4CD5E4D4 C5D9C9C3 3C5D6840    1CFC115C F913C3                       *T BE NUMERIC.), .C.*9..          *
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8  ../0CC0          20     91.123      19.29.1639    TEST1    CCCC   00                  8   FA  00    FF
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     C2  ../CCC0  PQ/D7D8          21     91.123      19.29.1771    TEST1    CCCC   00                  8   F3  C0    67
-------------------------------------------------------------------------------------------------------------------------------------
    62      0     F2  PQ/D7D8  ../0CC0          22     91.123      19.29.1981    TEST1    CCOC   C6                  9   C1  CC    FF
000C00    E3D7D3F1 6B11C54F F1F2F3F4 F911C65E    F1FCFC26                              *TPL1,.E712349.F.1CD.             *
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8  ../0CC0          22     91.123      19.29.1981    TEST1    COOC   GC                  9   30  C0    FF
-------------------------------------------------------------------------------------------------------------------------------------
   101      1     C2  ../0CC0  PQ/D7D8          23     91.123      19.29.1984    TEST1    CCCC   00                  9   F2  00    67
000000    F1C3114B 5012565F 115B7F1C E6C5D9D9    C6D94CC4 C5E2E2C1 C7C57A11 5CF81DC8   *1C..&.$..$".YERROR MESSACE:.*8.H*
000032    C9C5C3D6 D9C44CD5 D6E340C6 D6E4D5C4    3C5D684C 1CFC115C F913C3              *RECCRD NCT FOLNC.), .C.*9..      *
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8  ../0CC0          22     91.123      19.29.1984    TEST1    CCCC   GC                  9   FA  CC    FF
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     C2  ../0CC0  PQ/D7D8          23     91.123      19.29.2107    TEST1    CCCC   00                  9   F3  00    67
-------------------------------------------------------------------------------------------------------------------------------------
    62      0     F2  PQ/D7D8  ../CCC0          24     91.123      19.29.2315    TEST1    CCCC   C6                 10   C1  CC    FF
COCCCC    E3D7D3F1 6811C54F F1F2F3F4 F211C65E    F1F0FC26                              *TPL1,.E712342.F.1C0.             *
-------------------------------------------------------------------------------------------------------------------------------------
    42      1     F2  PQ/D7D8  ../0CC0          24     91.123      19.29.2315    TEST1    CCOC   0C                 10   3C  CC    FF
-------------------------------------------------------------------------------------------------------------------------------------
   430      1     C2  ../0CC0  PQ/D7D8          25     91.123      19.29.2316    TEST1    CCCC   00                 1C   F2  CC    67
C00000    F5C31140 401DC140 4C4C401D F0C5D5E3    C5D94CE3 C9C1D5E2 C1C3E3C9 D6D54CC3   *5C.  .A    .CENTER TRANSACTION C*
000032    D6C4C511 C2F51DF0 C5D5E3C5 D940C4C1    E3C17A11 C5C51DE8 C7C1D9E3 4CD5C67A   *ODE.B5.CENTER CATA:.EE.YPART NC:*
CC0064    1D50F1F2 F3F4F21D FC11C6D5 1DE8E6C8    E24CD5C6 7A1D5CF1 F0F01DFC 11C9C51D   *.E12342.C.FN.YWHS NC:.E10C.0.IE.*
CC0096    F0E2E3D6 C3D240E2 E3C1E3E4 E27A1148    C51DFCC4 C5E2C3D9 C9D7E3C9 D6D57A1C   *OSTCCK STATUS:..N.CDESCRIPTION:.*
000128    4CF361F8 40C9D540 C6D3C1E3 40C8C5C1    C440C2C3 C1C3D240 C3C1D9D9 C9C1C7C5   * 3/8 IN FLAT HEAD BLACK CARRIAGE*
00C16C    4CC2D6D3 E33C4CD9 40114C5A 1DF0114C    E51DFCC6 D9C4C5D9 40E4D5C9 E3E27A1C   * BCLT.<R .<..C.<V.CORDER UNITS:.*

MSGLEN  THREAD   CPR   RSC       SSC            MMA     CATE          TIME          TID     FLGS  USR                BPN  LCG BLK  VPI
```

Figure 55. Simulation Mode Execution Log Printout (Page 4 of 6)



```
CATE 51.123    TIME 19.29.51         **** I N T E R C C M M    L G G    D I S P L A Y ****                    PAGE     5

MSCLEN  THREAD   CPR   RSC       SSC          MMN      CATE          TIME         TID     FLGS  USR             BPN  LCG BLK  VPI
--------------------------------------------------------------------------------------------------------------------------------
CCC192   4CC4D6E9 40401DFC 114DC61D FCD7DSCS    C3C57A1C 4C5BF5FC F64BF1F6 F1F61DFC   * DOZ  .C.(F.CPRICE:. $5C6.1616.C*
000224   114FD51D FUE2E3D6 C3D240E2 E3C1E3E4    E240C1E3 4CE6C1C9 C5C8D6E4 E2C57A11   *.7N.CSTCCK STATLS AT WAREHCUSE:.*
000256   D1E51DFO D3D6C3C1 E3C9D6D5 7A1D4CD4    C9C1D4CS 6840C6D3 C14B3CD2 C7401CFO   *JV.CLOCATICN:. MIAMI, FLA..KG .O*
CCC288   11D2F51D FOD6D540 C8C1D5C4 7A1D4CF6    F1F5FCF6 F1F41CFC 11D3C61D FOC1E24C   *.K5.CON HANC:. 6150614.C.LC.CAS *
CC0320   C6C67A1D 40F0F361 FCF561F8 F21DFC11    C4C51DFO C6D54CD6 D9C4C5DS 7A1D4CF5   *DF:. 03/C5/82.C.ME.CON ORDER:. 5*
COC352   FCF4FOF4 FOF41DFO 11D4E61D FOC1E240    C6C67A1C 4CF1FC61 F1F161F8 F21DFC11   *0404C4.C.MW.CAS CF:. 1C/11/82.C.*
CC0384   40C11303                                                                      * A..                            *
--------------------------------------------------------------------------------------------------------------------------------
    42       1     F2   PQ/D7D8   ../00CO         24     51.123      19.29.2316    TEST1    CC00    CC             10   FA   CO    FF
--------------------------------------------------------------------------------------------------------------------------------
    42       1     02   ../CCCO   PQ/D7D8         25     51.123      19.29.2554    TEST1    CC00    CC             10   F3   CC    67
--------------------------------------------------------------------------------------------------------------------------------
    48       0     F2   .J/CCD1   ../0CCO         26     51.123      19.29.2773    CNTC1    CCCC    CC             11   C1   CC    FF
CCC000   C5D9C3C4 6626                                                                 *NRCD,.                          *
--------------------------------------------------------------------------------------------------------------------------------
    42       1     F2   .J/0CD1   ../CCCD         26     91.123      19.29.2773    CNT01    CCCC    0C             11   30   CC    FF
--------------------------------------------------------------------------------------------------------------------------------
   108       1     F2   .U/0CE4   .J/0CD1         27     51.123      19.29.2773    TOALL    CCCC    0C             11   C1   00    50
000000   FF02002D 013C5C5C 5C40C7D6 D6C44CC5    E5C5D5C9 D5C74C5C 5C5C4040 C9D5E3C5   *......*** GCCD EVENING ***   INTE*
00C032   C9C3D6D4 D440C9E2 40C3D3D6 E2C5C47A    4C4C4CFC F560FCF3 6CF9F140 4CF1F54B   *RCOMM IS CLCSED:    05-C3-51  19.*
0CCC64   F2F9                                                                          *29                              *
--------------------------------------------------------------------------------------------------------------------------------
    42       1     F2   .J/00D1   ../0CCO         26     51.123      19.29.2773    CNT01    CCCC    0C             11   FA   CC    FF
--------------------------------------------------------------------------------------------------------------------------------
    42       1     F2   .U/00E4   .J/00D1         27     51.123      19.29.2773    TOALL    CCCC    C0             11   30   00    5C
--------------------------------------------------------------------------------------------------------------------------------
    42       1     F2   .U/0CE4   .J/CC01         27     51.123      19.29.2773    TOALL    CCCC    CC             11   FA   CO    50
--------------------------------------------------------------------------------------------------------------------------------
   108       0     F2   .U/0CE4   .J/00D1         28     51.123      19.29.2773    CNTC1    CCCC    CC             11   C1   00    5C
000000   FF02002D 013C5C5C 5C40C7D6 D6C44CC5    E5C5D5C9 C5C74C5C 5C5C4040 C9D5E3C5   *......*** GCCD EVENING ***   INTE*
000032   C9C3D6D4 D440C9E2 40C3D3D6 E2C5C47A    4C404CFC F560FCF3 6CF9F140 4CF1F54B   *RCOMM IS CLCSED:    05-C3-51  19.*
CC0064   F2F9                                                                          *29                              *
--------------------------------------------------------------------------------------------------------------------------------
    42       1     F2   .U/0CE4   .J/0CD1         28     91.123      19.29.2773    CNT01    CCCC    CC             11   3C   C0    5C
--------------------------------------------------------------------------------------------------------------------------------
   103       1     C2   ../0000   .U/00E4         29     51.123      19.29.2775    CNTC1    CCCC    CC             11   F2   00    5C
CCC000   5C5C5C40 C7D6D6C4 40C5E5C5 D5C9D5C7    4C5C5C5C 4C4CC9D5 E3C5C9C3 D6D4C44C   **** GOCC EVENING ***   INTERCOMM *
000032   C9E240C3 D3D6E2C5 C47A4040 40F0F560    FCF36CF9 F14C4CF1 F94BF2F9 37         *IS CLOSEC:   05-C3-51   19.29.   *
--------------------------------------------------------------------------------------------------------------------------------
    42       1     F2   .U/00E4   .J/00D1         28     51.123      19.29.2775    CNT01    COCC    0C             11   FA   00    5C
--------------------------------------------------------------------------------------------------------------------------------
   108       0     F2   .U/0CE4   .J/CCD1         3C     51.123      19.29.2823    TEST1    CCCC    0C             11   C1   CO    50
000000   FF02002D 013C5C5C 5C40C7D6 D6C44CC5    E5C5D5C9 D5C7405C 5C5C4040 C9D5E3C5   *......*** GCCD EVENING ***   INTE*
00C032   C9C3D6D4 D440C9E2 40C3D3D6 E2C5C47A    4C404CFC F560FCF3 60F9F140 4CF1F94B   *RCOMM IS CLCSED:    05-C3-51  19.*
000064   F2F9                                                                          *29                              *
--------------------------------------------------------------------------------------------------------------------------------
    42       1     F2   .U/00E4   .J/00D1         3C     51.123      19.29.2823    TEST1    C000    CC             11   3C   CO    5C
--------------------------------------------------------------------------------------------------------------------------------
   1C3       1     C2   ../0C00   .U/CCE4         31     91.123      19.29.2823    TEST1    CCCC    0C             11   F2   CC    50
COC000   5C5C5C40 C7D6D6C4 40C5E5C5 D5C9D5C7    4C5C5C5C 4C40C9D5 E3C5C9C3 D6D4C44C   **** GOCC EVENING ***   INTERCOMM *
000032   C9E240C3 D3D6E2C5 C47A4040 40F0F560    FCF36CF9 F1404CF1 F94BF2F9 37         *IS CLOSEC:   05-C3-51   19.29.   *

MSGLEN  THREAD   CPR   RSC       SSC          MMN      DATE          TIME         TID     FLGS  USR             BMN  LCG BLK  VPI
```

Figure 55. Simulation Mode Execution Log Printout (Page 5 of 6)



```
DATE 91.123    TIME 19.29.51         **** I N T E R C O M M   L O G   D I S P L A Y ****                 PAGE     6
MSGLEN  THREAD   GPK   RSC      SSC          MMN     DATE        TIME        TID     FLGS  USR            BPN  LOG BLK  VPI
---------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------
    42       1     F2  .U/0CE4  .J/0CD1        30    91.123    19.29.2823   TEST1   CCCC   0C              11   FA  CC   5C
---------------------------------------------------------------------------------------------------------------------------
    48       0     F2  .J/0CD1  ../0000        32    91.123    19.29.2874   CNT01   CCCC   CC              11   C1  CC   FC
000000   D5D9C3C4 6B26                                                                 *NRCD,.                            *
---------------------------------------------------------------------------------------------------------------------------
    42       1     F2  .J/0CD1  ../0000        32    91.123    19.29.2874   CNT01   CC00   0C              11   3C  C0   FC
---------------------------------------------------------------------------------------------------------------------------
    42       1     C2  ../0CC0  .U/0CE4        29    91.123    19.29.2904   CNT01   CCCC   0C              11   F3  CC   5C
---------------------------------------------------------------------------------------------------------------------------
    42       1     C2  ../0C00  .U/0CE4        31    91.123    19.29.2933   TEST1   CC0C   CC              11   F3  CC   5C
---------------------------------------------------------------------------------------------------------------------------
    78       0     00  ../0CC0  ../CCC0         0    91.123    19.29.3878   .....   CCCC   CC               C   AA  C0   C0
C00000   C9D5E3C5 D9C3D6D4 D440C3D3 D6E2C5C4    C6E6D540 D4C5E2E2 C1C7C540 C9D5E3E3    *INTERCOMM CLOSEDOWN MESSAGE INIT*
00C032   FCFCF1E7                                                              *CC1X                               *
---------------------------------------------------------------------------------------------------------------------------
```

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 PL/1 subsystem (SQPL1) designed for the same purpose as SQPL1A, but using the Edit, Output and Change/Display Utilities.

12.2 TESTING A SUBSYSTEM IN TEST MODE

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

<u>NOTE</u>: 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 PL/1 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 <u>Utilities Users Guide</u> and the <u>Operating Reference Manual</u>).

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 accomodate 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 on 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.

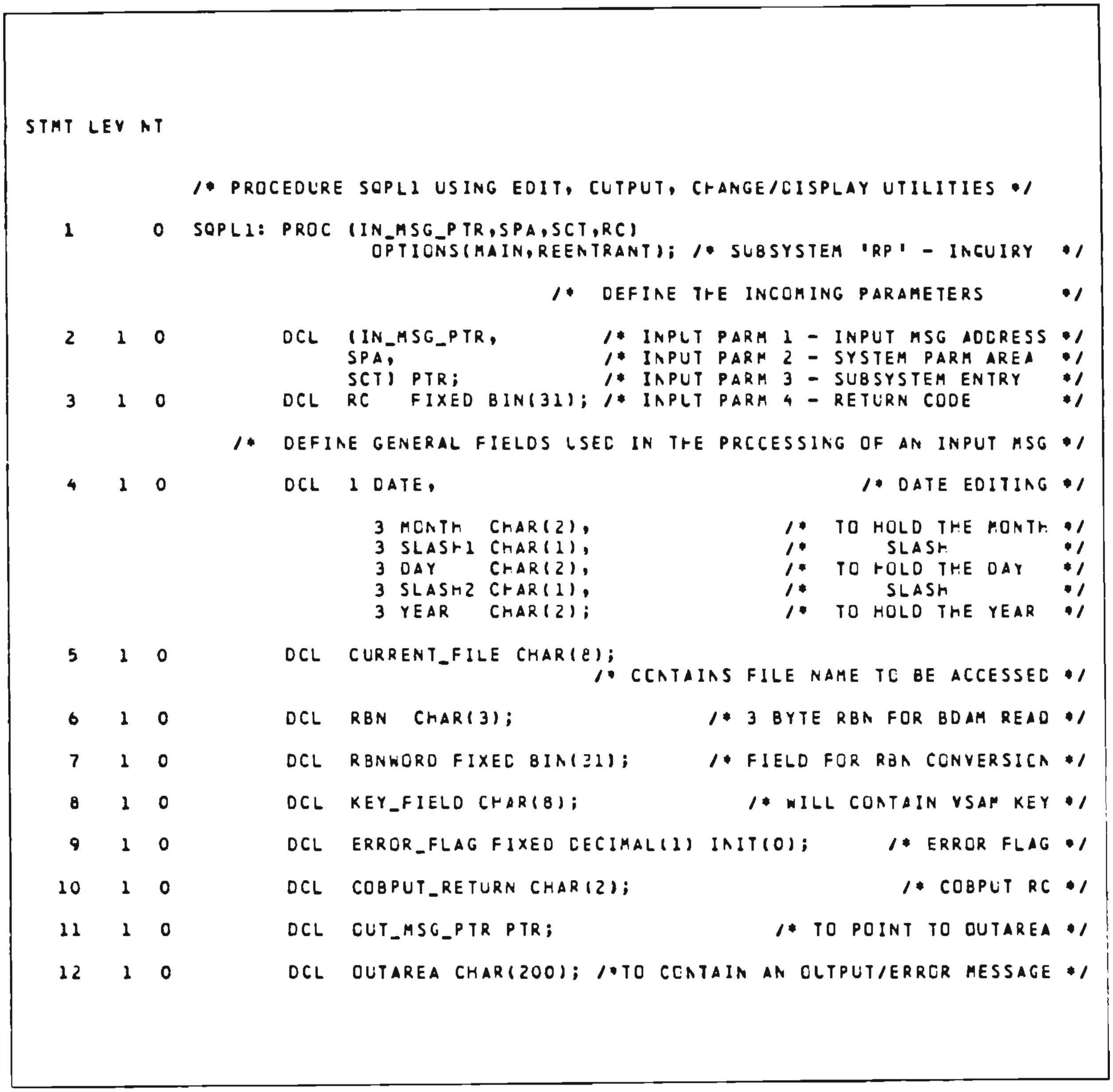
    The resulting snaps for the test mode execution of the sample inquiry subsystem are illustrated in Figure 62.

    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.

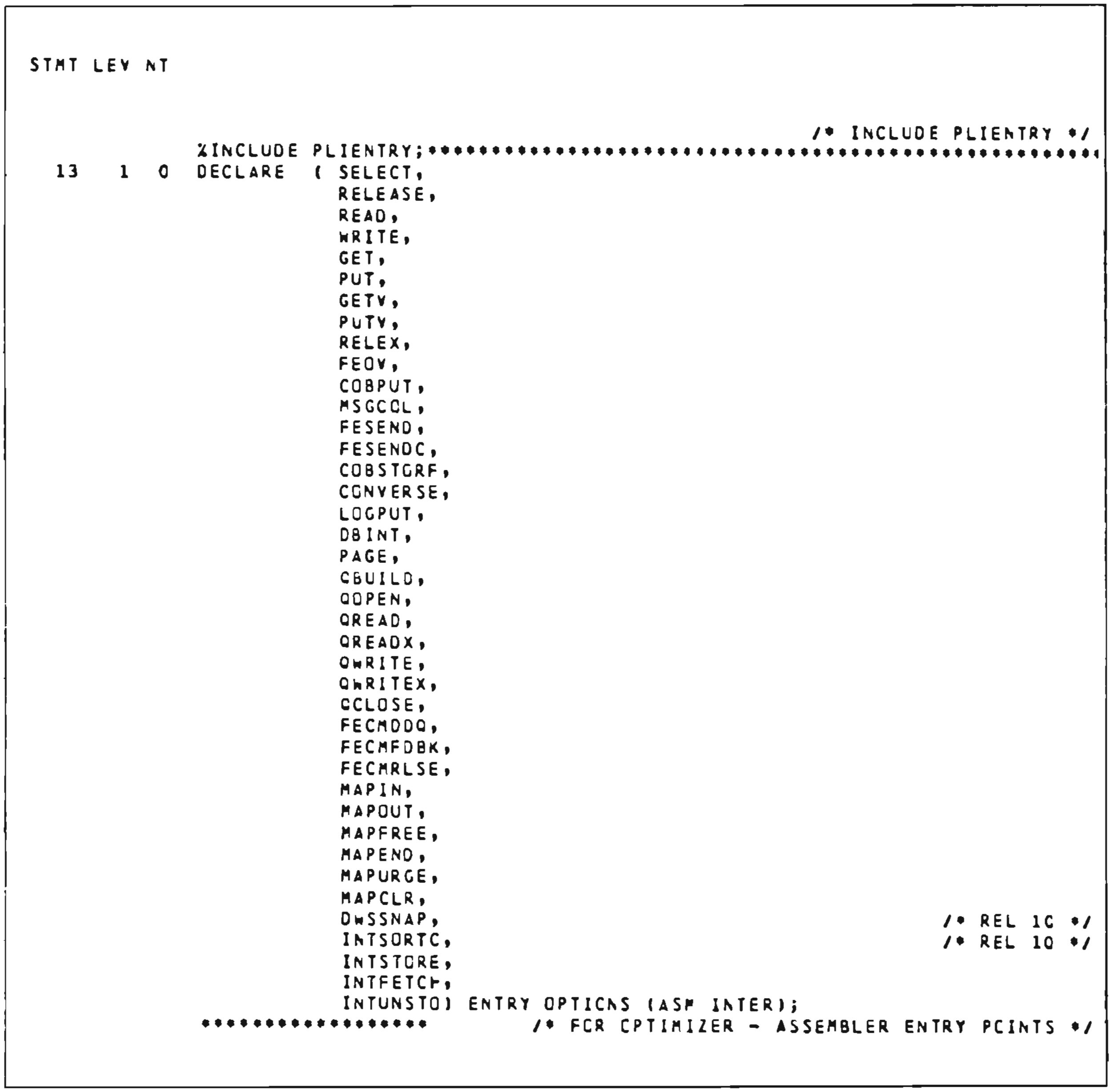
Note: to implement the sample subsystem for on-line execution, it would be necessary to code a BTVERB macro (in USRBTVRB--see Chapter 11) as follows:

```
BTVERB VERB=RTRP,SSCH=R,SSC=P,CONV=18000,EDIT=YES
```



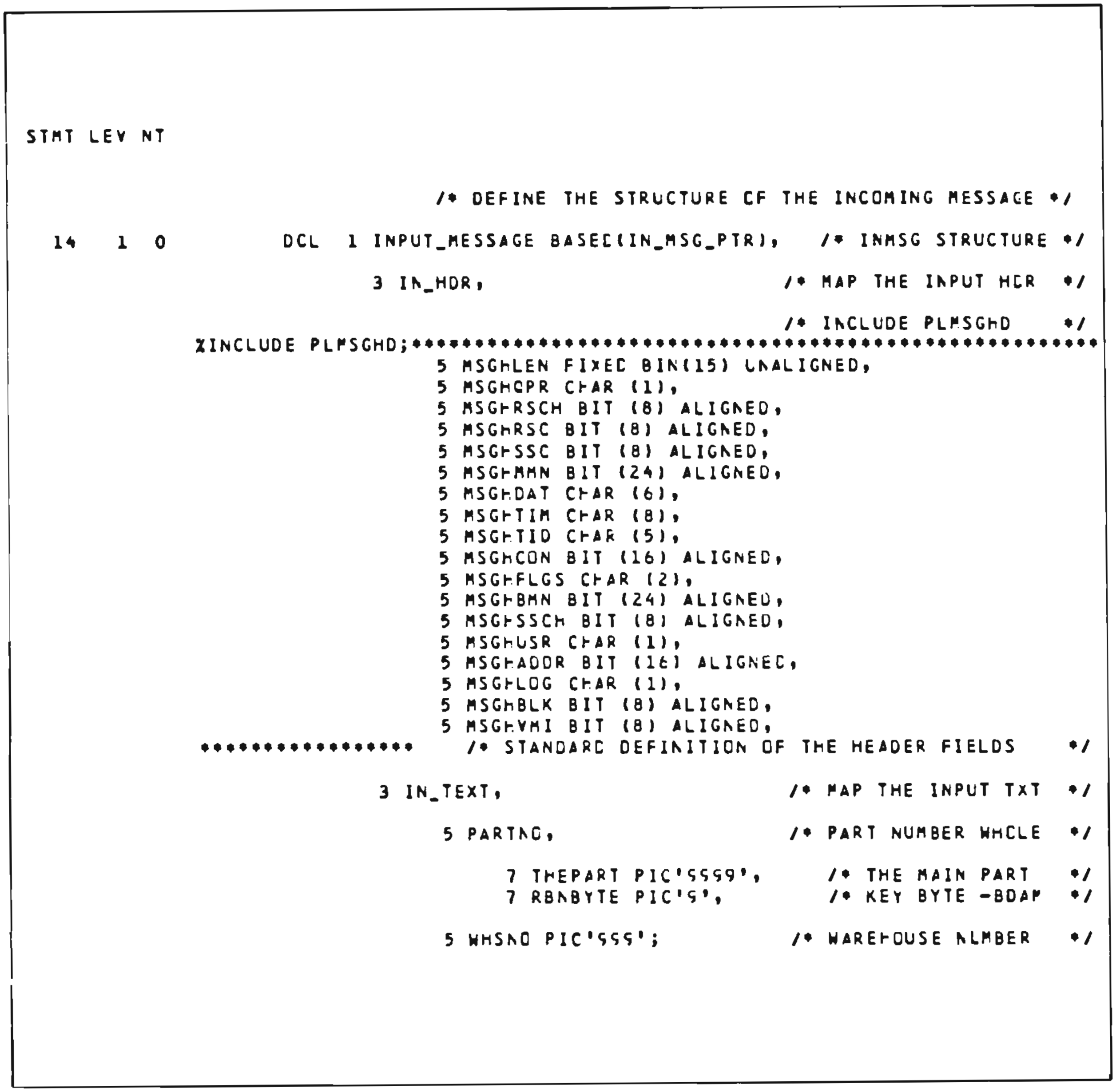
```
STMT LEV NT

                /* PROCEDURE SQPL1 USING EDIT, OUTPUT, CHANGE/DISPLAY UTILITIES */

   1      0 SQPL1: PROC (IN_MSG_PTR,SPA,SCT,RC)
                         OPTIONS(MAIN,REENTRANT); /* SUBSYSTEM 'RP' - INQUIRY   */

                                   /*  DEFINE THE INCOMING PARAMETERS           */

   2   1  0         DCL  (IN_MSG_PTR,         /* INPUT PARM 1 - INPUT MSG ADDRESS */
                          SPA,                /* INPUT PARM 2 - SYSTEM PARM AREA  */
                          SCT) PTR;           /* INPUT PARM 3 - SUBSYSTEM ENTRY   */
   3   1  0         DCL  RC    FIXED BIN(31); /* INPUT PARM 4 - RETURN CODE       */

                  /*  DEFINE GENERAL FIELDS USED IN THE PROCESSING OF AN INPUT MSG */

   4   1  0         DCL  1 DATE,                                   /* DATE EDITING */

                           3 MONTH  CHAR(2),               /*  TO HOLD THE MONTH  */
                           3 SLASH1 CHAR(1),               /*      SLASH          */
                           3 DAY    CHAR(2),               /*  TO HOLD THE DAY    */
                           3 SLASH2 CHAR(1),               /*      SLASH          */
                           3 YEAR   CHAR(2);               /*  TO HOLD THE YEAR   */

   5   1  0         DCL  CURRENT_FILE CHAR(8);
                                     /* CONTAINS FILE NAME TO BE ACCESSED */

   6   1  0         DCL  RBN  CHAR(3);               /* 3 BYTE RBN FOR BDAM READ */

   7   1  0         DCL  RBNWORD FIXED BIN(31);      /* FIELD FOR RBN CONVERSION */

   8   1  0         DCL  KEY_FIELD CHAR(8);            /* WILL CONTAIN VSAM KEY */

   9   1  0         DCL  ERROR_FLAG FIXED DECIMAL(1) INIT(0);      /* ERROR FLAG */

  10   1  0         DCL  COBPUT_RETURN CHAR(2);                     /* COBPUT RC */

  11   1  0         DCL  OUT_MSG_PTR PTR;                /* TO POINT TO OUTAREA */

  12   1  0         DCL  OUTAREA CHAR(200); /*TO CONTAIN AN OUTPUT/ERROR MESSAGE */
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 1 of 13)



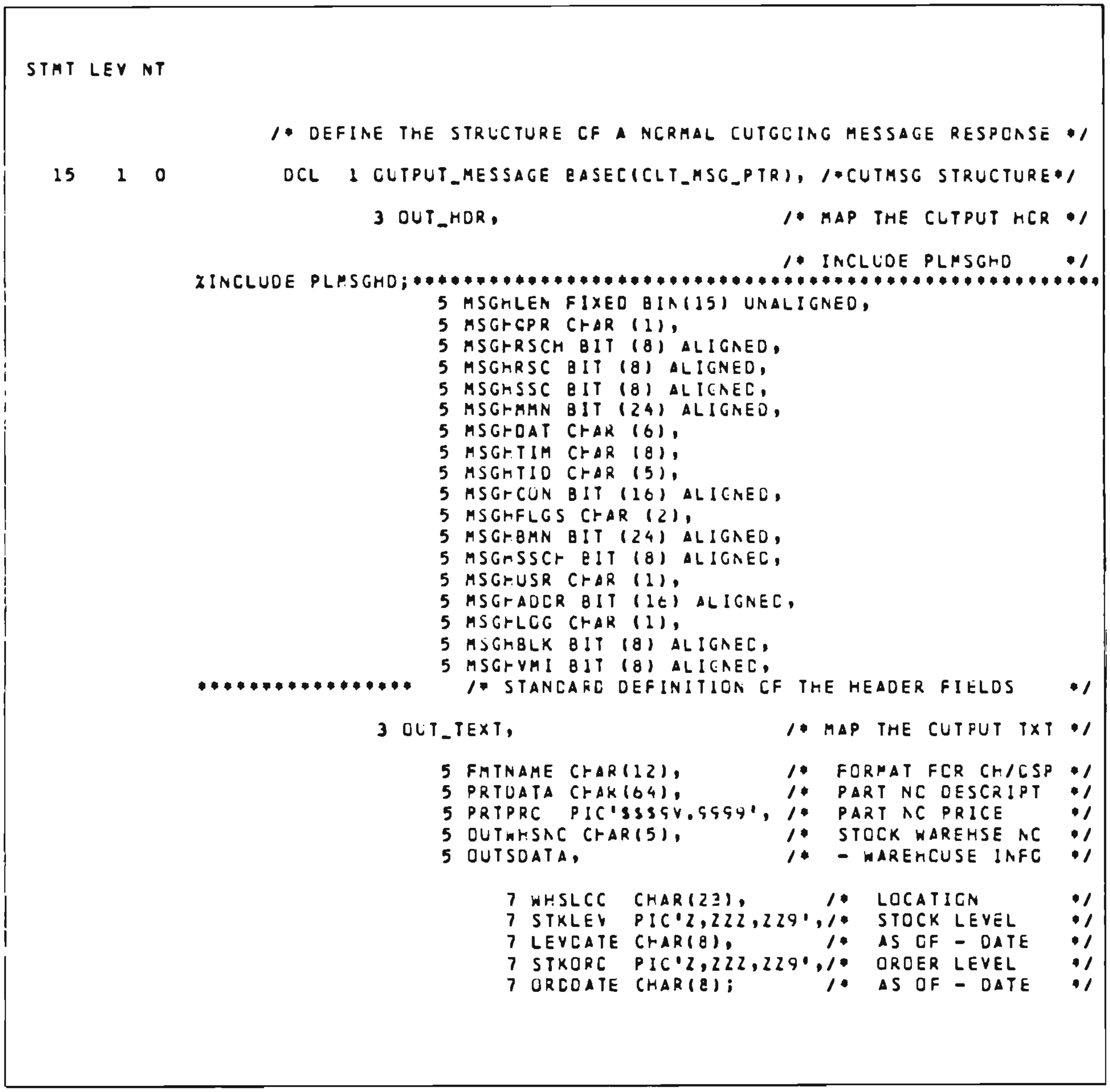
```
STMT LEV NT


                                                                           /* INCLUDE PLIENTRY */
              %INCLUDE PLIENTRY;************************************************************
  13    1  0  DECLARE  ( SELECT,
                         RELEASE,
                         READ,
                         WRITE,
                         GET,
                         PUT,
                         GETV,
                         PUTV,
                         RELEX,
                         FEOV,
                         COBPUT,
                         MSGCOL,
                         FESEND,
                         FESENDC,
                         COBSTGRF,
                         CONVERSE,
                         LOGPUT,
                         DBINT,
                         PAGE,
                         QBUILD,
                         QOPEN,
                         QREAD,
                         QREADX,
                         QWRITE,
                         QWRITEX,
                         QCLOSE,
                         FECMDDQ,
                         FECMFDBK,
                         FECMRLSE,
                         MAPIN,
                         MAPOUT,
                         MAPFREE,
                         MAPEND,
                         MAPURGE,
                         MAPCLR,
                         DWSSNAP,                                            /* REL 1C */
                         INTSORTC,                                           /* REL 10 */
                         INTSTORE,
                         INTFETCH,
                         INTUNSTO) ENTRY OPTICNS (ASM INTER);
              ******************         /* FCR CPTIMIZER - ASSEMBLER ENTRY PCINTS */
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 2 of 13)



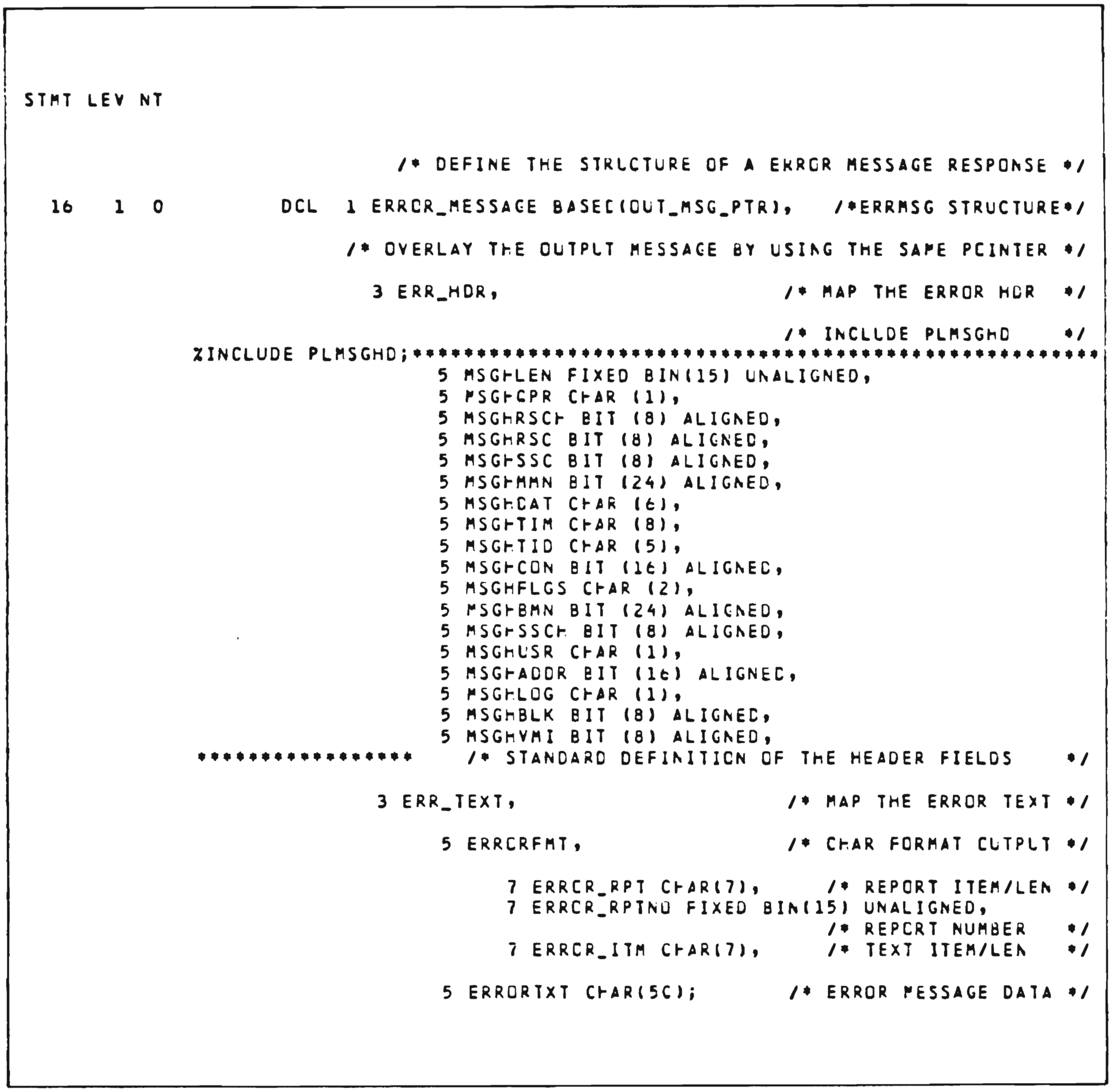
```
STMT LEV NT

                                /* DEFINE THE STRUCTURE OF THE INCOMING MESSAGE */

  14    1  0          DCL  1 INPUT_MESSAGE BASED(IN_MSG_PTR),   /* INMSG STRUCTURE */

                             3 IN_HDR,                        /* MAP THE INPUT HDR   */

                                                              /* INCLUDE PLMSGHD     */
             %INCLUDE PLMSGHD;******************************************************
                                 5 MSGHLEN FIXED BIN(15) UNALIGNED,
                                 5 MSGHQPR CHAR (1),
                                 5 MSGHRSCH BIT (8) ALIGNED,
                                 5 MSGHRSC BIT (8) ALIGNED,
                                 5 MSGHSSC BIT (8) ALIGNED,
                                 5 MSGHMMN BIT (24) ALIGNED,
                                 5 MSGHDAT CHAR (6),
                                 5 MSGHTIM CHAR (8),
                                 5 MSGHTID CHAR (5),
                                 5 MSGHCON BIT (16) ALIGNED,
                                 5 MSGHFLGS CHAR (2),
                                 5 MSGHBMN BIT (24) ALIGNED,
                                 5 MSGHSSCH BIT (8) ALIGNED,
                                 5 MSGHUSR CHAR (1),
                                 5 MSGHADDR BIT (16) ALIGNED,
                                 5 MSGHLOG CHAR (1),
                                 5 MSGHBLK BIT (8) ALIGNED,
                                 5 MSGHVMI BIT (8) ALIGNED,
              *****************    /* STANDARD DEFINITION OF THE HEADER FIELDS    */

                            3 IN_TEXT,                        /* MAP THE INPUT TXT   */

                                 5 PARTNO,                    /* PART NUMBER WHOLE   */

                                      7 THEPART PIC'9999',       /* THE MAIN PART    */
                                      7 RBNBYTE PIC'9',          /* KEY BYTE -BDAM   */

                                 5 WHSNO PIC'999';            /* WAREHOUSE NUMBER    */
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 3 of 13)



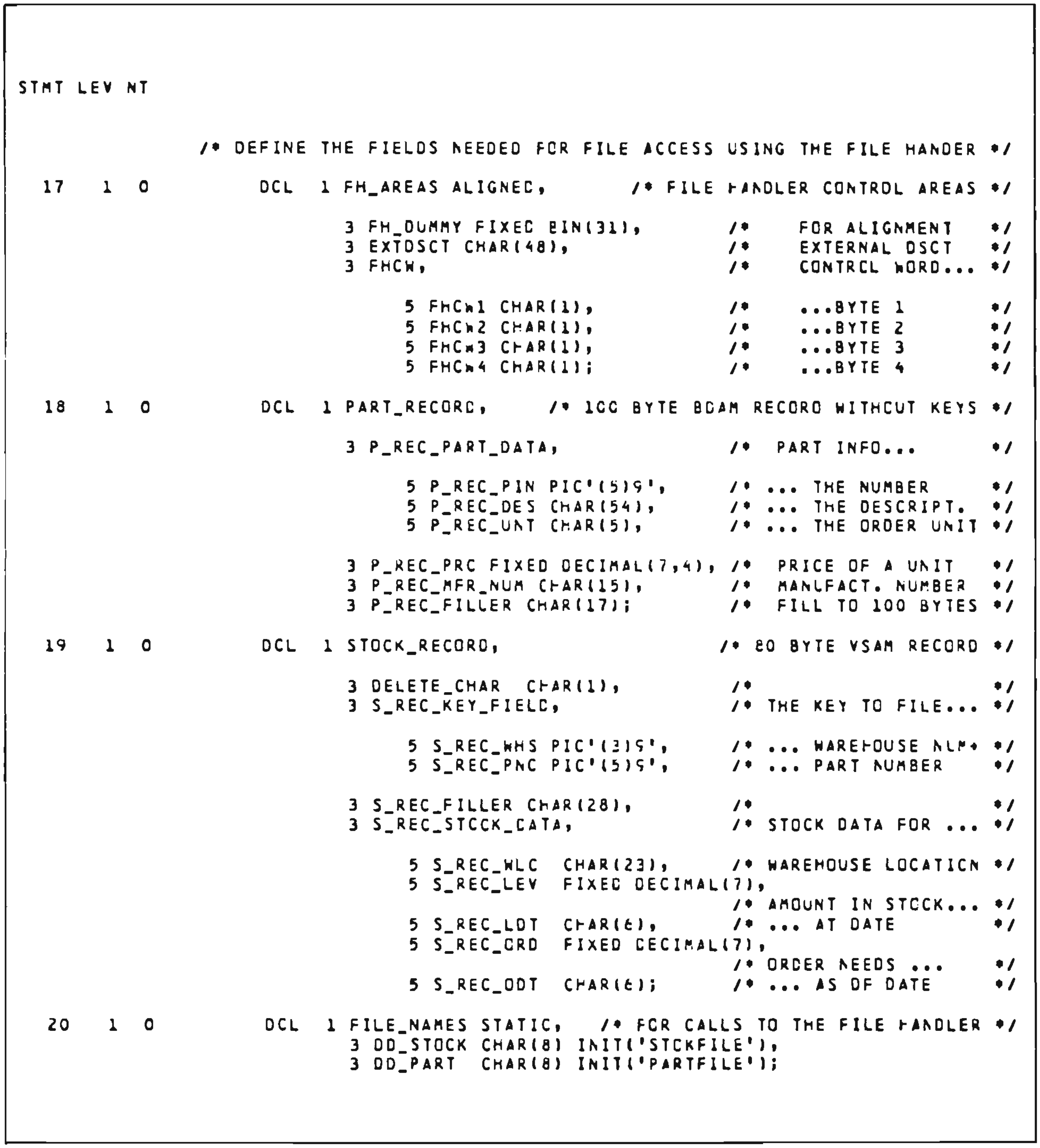
```
STMT LEV NT

                   /* DEFINE THE STRUCTURE OF A NORMAL OUTGOING MESSAGE RESPONSE */

  15    1  0        DCL  1 OUTPUT_MESSAGE BASED(OUT_MSG_PTR), /*OUTMSG STRUCTURE*/

                           3 OUT_HDR,                     /* MAP THE OUTPUT HDR */

                                                          /* INCLUDE PLMSGHD    */
             %INCLUDE PLMSGHD;***************************************************************
                                5 MSGHLEN FIXED BIN(15) UNALIGNED,
                                5 MSGHCPR CHAR (1),
                                5 MSGHRSCH BIT (8) ALIGNED,
                                5 MSGHRSC BIT (8) ALIGNED,
                                5 MSGHSSC BIT (8) ALIGNED,
                                5 MSGHMMN BIT (24) ALIGNED,
                                5 MSGHDAT CHAR (6),
                                5 MSGHTIM CHAR (8),
                                5 MSGHTID CHAR (5),
                                5 MSGHCON BIT (16) ALIGNED,
                                5 MSGHFLGS CHAR (2),
                                5 MSGHBMN BIT (24) ALIGNED,
                                5 MSGHSSCH BIT (8) ALIGNED,
                                5 MSGHUSR CHAR (1),
                                5 MSGHADDR BIT (16) ALIGNED,
                                5 MSGHLOG CHAR (1),
                                5 MSGHBLK BIT (8) ALIGNED,
                                5 MSGHVMI BIT (8) ALIGNED,
             *****************    /* STANDARD DEFINITION OF THE HEADER FIELDS     */

                           3 OUT_TEXT,                    /* MAP THE OUTPUT TXT */

                                5 FMTNAME CHAR(12),       /*  FORMAT FOR CH/CSP */
                                5 PRTDATA CHAR(64),       /*  PART NO DESCRIPT  */
                                5 PRTPRC  PIC'$$$$V.9999', /*  PART NO PRICE     */
                                5 OUTWHSNO CHAR(5),       /*  STOCK WAREHSE NO  */
                                5 OUTSDATA,               /*  - WAREHOUSE INFO  */

                                     7 WHSLOC  CHAR(23),      /*  LOCATION        */
                                     7 STKLEV  PIC'Z,ZZZ,ZZ9',/*  STOCK LEVEL     */
                                     7 LEVDATE CHAR(8),       /*  AS OF - DATE    */
                                     7 STKORD  PIC'Z,ZZZ,ZZ9',/*  ORDER LEVEL     */
                                     7 ORDDATE CHAR(8);       /*  AS OF - DATE    */
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 4 of 13)



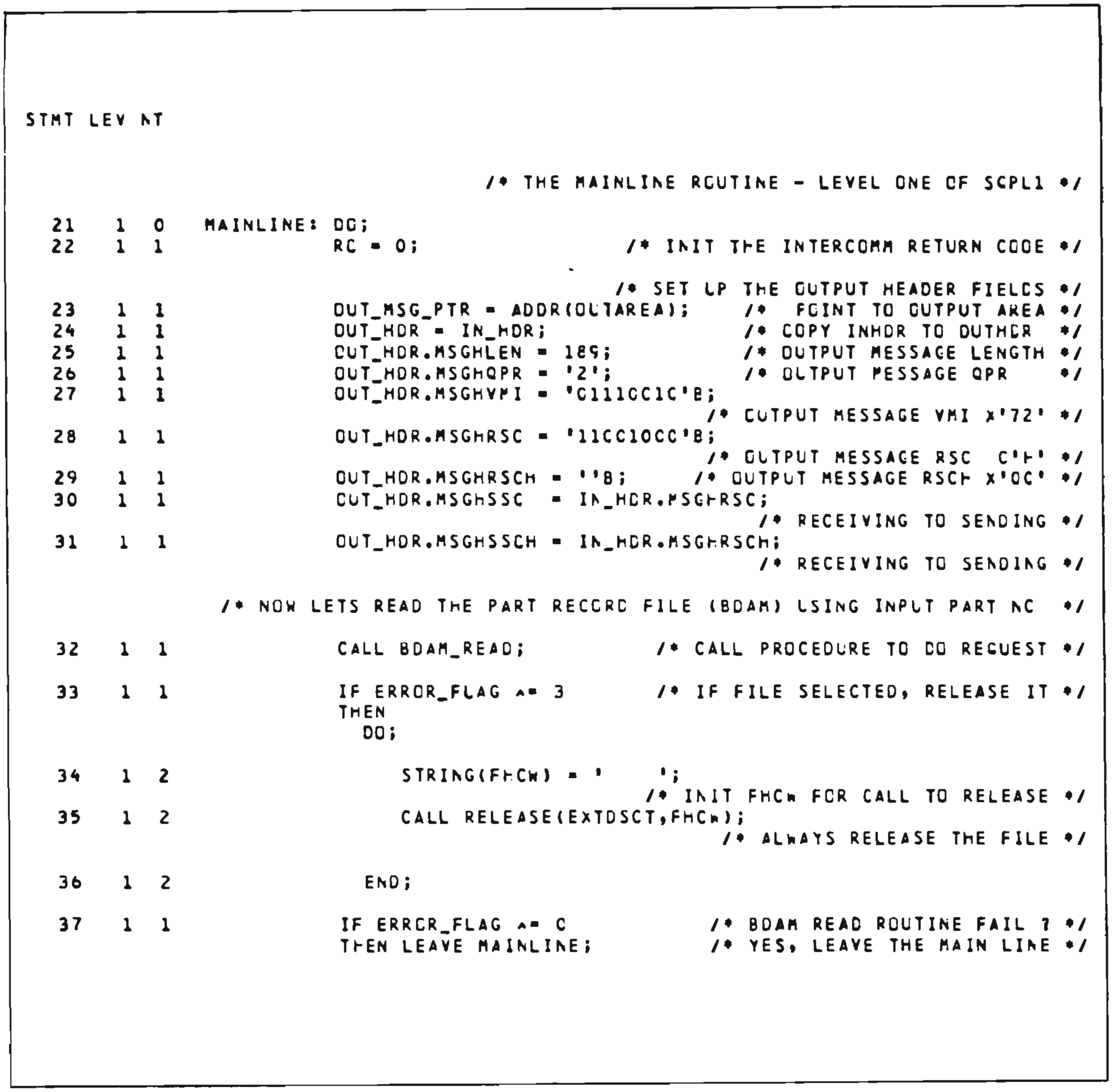
```
STMT LEV NT

                              /* DEFINE THE STRUCTURE OF A ERROR MESSAGE RESPONSE */

  16    1  0         DCL  1 ERROR_MESSAGE BASED(OUT_MSG_PTR),    /*ERRMSG STRUCTURE*/

                          /* OVERLAY THE OUTPUT MESSAGE BY USING THE SAME POINTER */

                            3 ERR_HDR,                       /* MAP THE ERROR HDR   */

                                                             /* INCLUDE PLMSGHD     */
             %INCLUDE PLMSGHD;********************************************************
                                 5 MSGHLEN FIXED BIN(15) UNALIGNED,
                                 5 MSGHCPR CHAR (1),
                                 5 MSGHRSCH BIT (8) ALIGNED,
                                 5 MSGHRSC BIT (8) ALIGNED,
                                 5 MSGHSSC BIT (8) ALIGNED,
                                 5 MSGHMMN BIT (24) ALIGNED,
                                 5 MSGHDAT CHAR (6),
                                 5 MSGHTIM CHAR (8),
                                 5 MSGHTID CHAR (5),
                                 5 MSGHCON BIT (16) ALIGNED,
                                 5 MSGHFLGS CHAR (2),
                                 5 MSGHBMN BIT (24) ALIGNED,
                                 5 MSGHSSCH BIT (8) ALIGNED,
                                 5 MSGHUSR CHAR (1),
                                 5 MSGHADDR BIT (16) ALIGNED,
                                 5 MSGHLOG CHAR (1),
                                 5 MSGHBLK BIT (8) ALIGNED,
                                 5 MSGHVMI BIT (8) ALIGNED,
              *****************    /* STANDARD DEFINITION OF THE HEADER FIELDS    */

                            3 ERR_TEXT,                      /* MAP THE ERROR TEXT */

                                 5 ERRORFMT,                 /* CHAR FORMAT OUTPUT */

                                      7 ERROR_RPT CHAR(7),      /* REPORT ITEM/LEN */
                                      7 ERROR_RPTNO FIXED BIN(15) UNALIGNED,
                                                                /* REPORT NUMBER   */
                                      7 ERROR_ITM CHAR(7),      /* TEXT ITEM/LEN   */

                                 5 ERRORTXT CHAR(50);        /* ERROR MESSAGE DATA */
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 5 of 13)



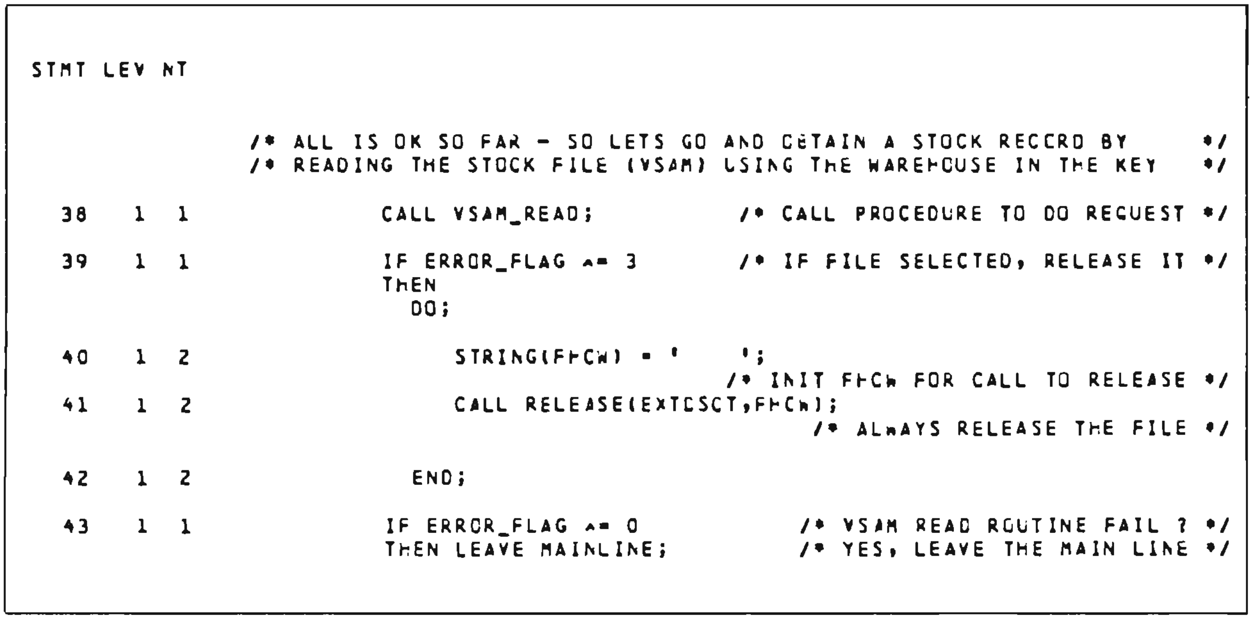
```
STMT LEV NT


                 /* DEFINE THE FIELDS NEEDED FOR FILE ACCESS USING THE FILE HANDER */

  17   1  0          DCL  1 FH_AREAS ALIGNED,       /* FILE HANDLER CONTROL AREAS */

                            3 FH_DUMMY FIXED BIN(31),        /*    FOR ALIGNMENT   */
                            3 EXTDSCT CHAR(48),              /*    EXTERNAL DSCT   */
                            3 FHCW,                          /*    CONTROL WORD... */

                               5 FHCW1 CHAR(1),              /*    ...BYTE 1       */
                               5 FHCW2 CHAR(1),              /*    ...BYTE 2       */
                               5 FHCW3 CHAR(1),              /*    ...BYTE 3       */
                               5 FHCW4 CHAR(1);              /*    ...BYTE 4       */

  18   1  0          DCL  1 PART_RECORD,     /* 100 BYTE BDAM RECORD WITHOUT KEYS */

                            3 P_REC_PART_DATA,               /*  PART INFO...      */

                               5 P_REC_PIN PIC'(5)9',        /* ... THE NUMBER     */
                               5 P_REC_DES CHAR(54),         /* ... THE DESCRIPT.  */
                               5 P_REC_UNT CHAR(5),          /* ... THE ORDER UNIT */

                            3 P_REC_PRC FIXED DECIMAL(7,4), /*  PRICE OF A UNIT   */
                            3 P_REC_MFR_NUM CHAR(15),        /*  MANUFACT. NUMBER  */
                            3 P_REC_FILLER CHAR(17);         /*  FILL TO 100 BYTES */

  19   1  0          DCL  1 STOCK_RECORD,                   /* 80 BYTE VSAM RECORD */

                            3 DELETE_CHAR  CHAR(1),          /*                    */
                            3 S_REC_KEY_FIELD,               /* THE KEY TO FILE... */

                               5 S_REC_WHS PIC'(3)9',        /* ... WAREHOUSE NUM. */
                               5 S_REC_PNO PIC'(5)9',        /* ... PART NUMBER    */

                            3 S_REC_FILLER CHAR(28),         /*                    */
                            3 S_REC_STOCK_DATA,              /* STOCK DATA FOR ... */

                               5 S_REC_WLC  CHAR(23),      /* WAREHOUSE LOCATION */
                               5 S_REC_LEV  FIXED DECIMAL(7),
                                                             /* AMOUNT IN STOCK... */
                               5 S_REC_LDT  CHAR(6),       /* ... AT DATE        */
                               5 S_REC_ORD  FIXED DECIMAL(7),
                                                             /* ORDER NEEDS ...    */
                               5 S_REC_ODT  CHAR(6);       /* ... AS OF DATE     */

  20   1  0          DCL  1 FILE_NAMES STATIC,   /* FOR CALLS TO THE FILE HANDLER */
                            3 DD_STOCK CHAR(8) INIT('STCKFILE'),
                            3 DD_PART  CHAR(8) INIT('PARTFILE');
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 6 of 13)



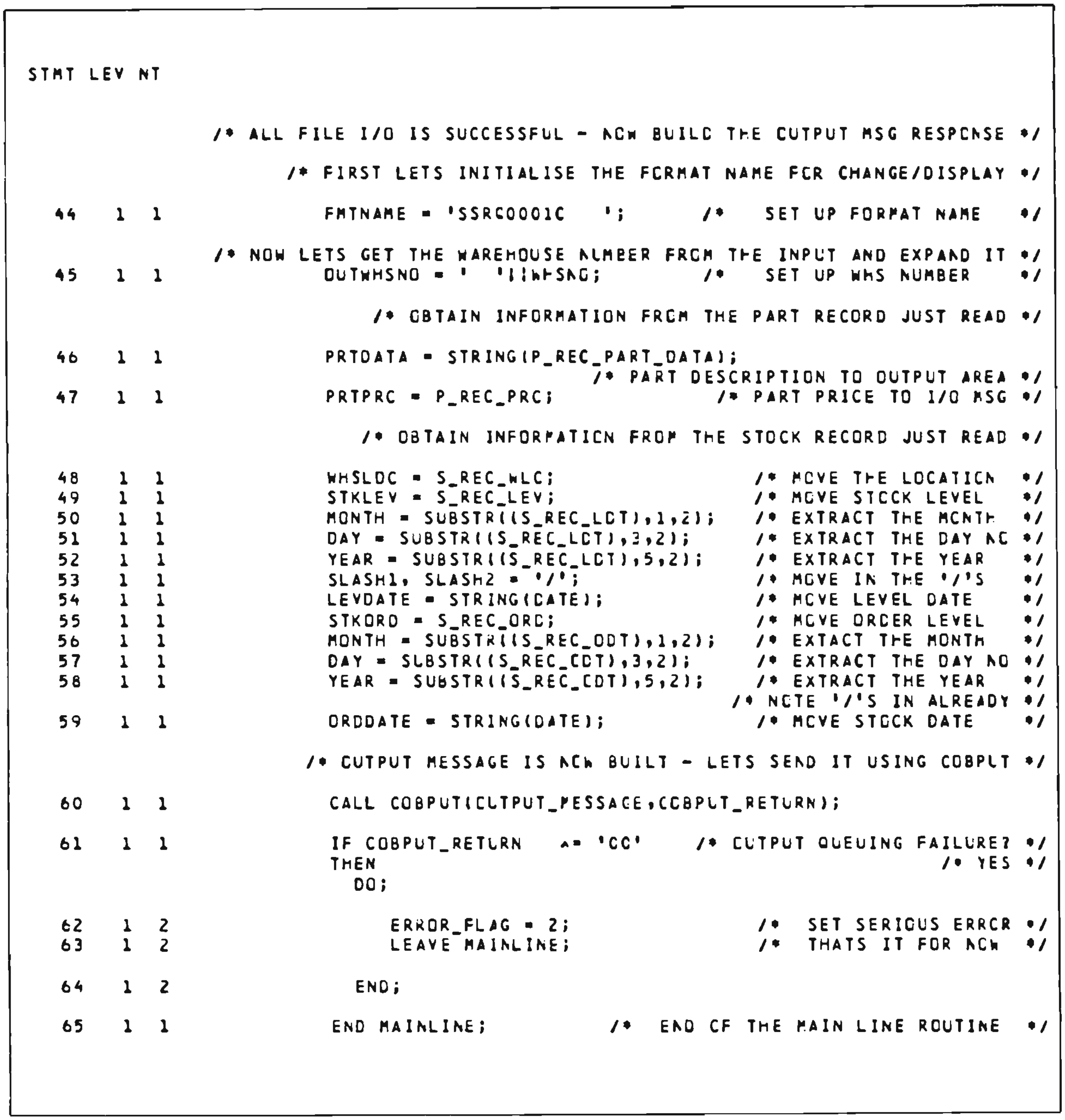
```
STMT LEV NT

                                                   /* THE MAINLINE ROUTINE - LEVEL ONE OF SQPL1 */

  21   1  0   MAINLINE: DO;
  22   1  1             RC = 0;                    /* INIT THE INTERCOMM RETURN CODE */

                                                   /* SET UP THE OUTPUT HEADER FIELDS */
  23   1  1             OUT_MSG_PTR = ADDR(OUTAREA);     /*  POINT TO OUTPUT AREA */
  24   1  1             OUT_HDR = IN_HDR;                /* COPY INHDR TO OUTHDR  */
  25   1  1             OUT_HDR.MSGHLEN = 189;           /* OUTPUT MESSAGE LENGTH */
  26   1  1             OUT_HDR.MSGHQPR = '2';           /* OUTPUT MESSAGE QPR    */
  27   1  1             OUT_HDR.MSGHVMI = '01110010'B;
                                                      /* OUTPUT MESSAGE VMI X'72' */
  28   1  1             OUT_HDR.MSGHRSC = '11001000'B;
                                                      /* OUTPUT MESSAGE RSC  C'H' */
  29   1  1             OUT_HDR.MSGHRSCH = ''B;     /* OUTPUT MESSAGE RSCH X'00' */
  30   1  1             OUT_HDR.MSGHSSC  = IN_HDR.MSGHRSC;
                                                        /* RECEIVING TO SENDING */
  31   1  1             OUT_HDR.MSGHSSCH = IN_HDR.MSGHRSCH;
                                                        /* RECEIVING TO SENDING */

          /* NOW LETS READ THE PART RECORD FILE (BDAM) USING INPUT PART NO  */

  32   1  1             CALL BDAM_READ;           /* CALL PROCEDURE TO DO REQUEST */

  33   1  1             IF ERROR_FLAG ¬= 3        /* IF FILE SELECTED, RELEASE IT */
                        THEN
                          DO;

  34   1  2                 STRING(FHCW) = '    ';
                                                /* INIT FHCW FOR CALL TO RELEASE */
  35   1  2                 CALL RELEASE(EXTDSCT,FHCW);
                                                     /* ALWAYS RELEASE THE FILE */

  36   1  2               END;

  37   1  1             IF ERROR_FLAG ¬= 0           /* BDAM READ ROUTINE FAIL ? */
                        THEN LEAVE MAINLINE;         /* YES, LEAVE THE MAIN LINE */
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 7 of 13)



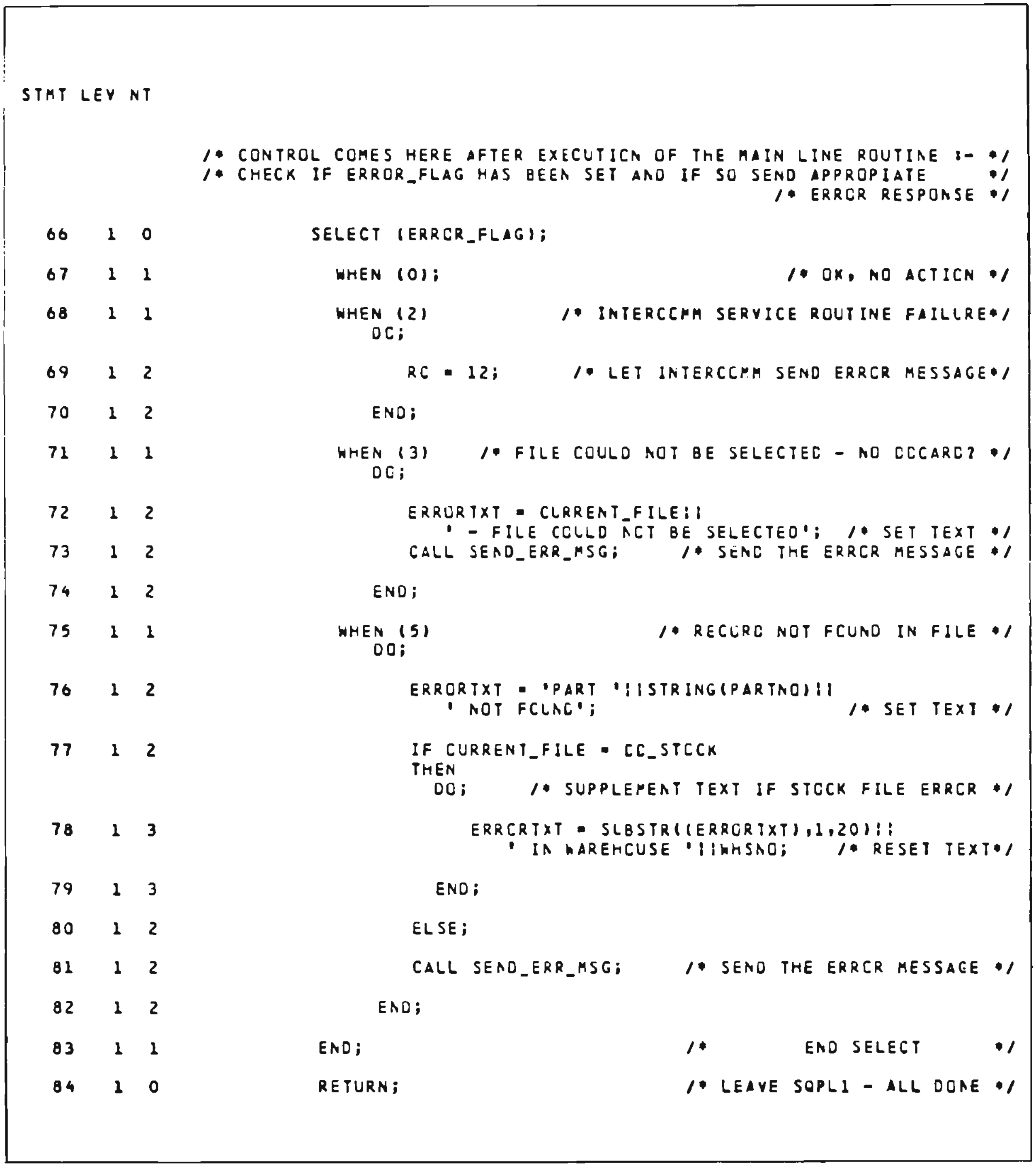
```
STMT LEV NT

                   /* ALL IS OK SO FAR - SO LETS GO AND OBTAIN A STOCK RECORD BY      */
                   /* READING THE STOCK FILE (VSAM) USING THE WAREHOUSE IN THE KEY    */

  38   1  1                 CALL VSAM_READ;          /* CALL PROCEDURE TO DO REQUEST */

  39   1  1                 IF ERROR_FLAG ¬= 3       /* IF FILE SELECTED, RELEASE IT */
                            THEN
                              DO;

  40   1  2                     STRING(FHCW) = '    ';
                                                    /* INIT FHCW FOR CALL TO RELEASE */
  41   1  2                     CALL RELEASE(EXTDSCT,FHCW);
                                                           /* ALWAYS RELEASE THE FILE */

  42   1  2                   END;

  43   1  1                 IF ERROR_FLAG ¬= 0           /* VSAM READ ROUTINE FAIL ? */
                            THEN LEAVE MAINLINE;         /* YES, LEAVE THE MAIN LINE */
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 8 of 13)



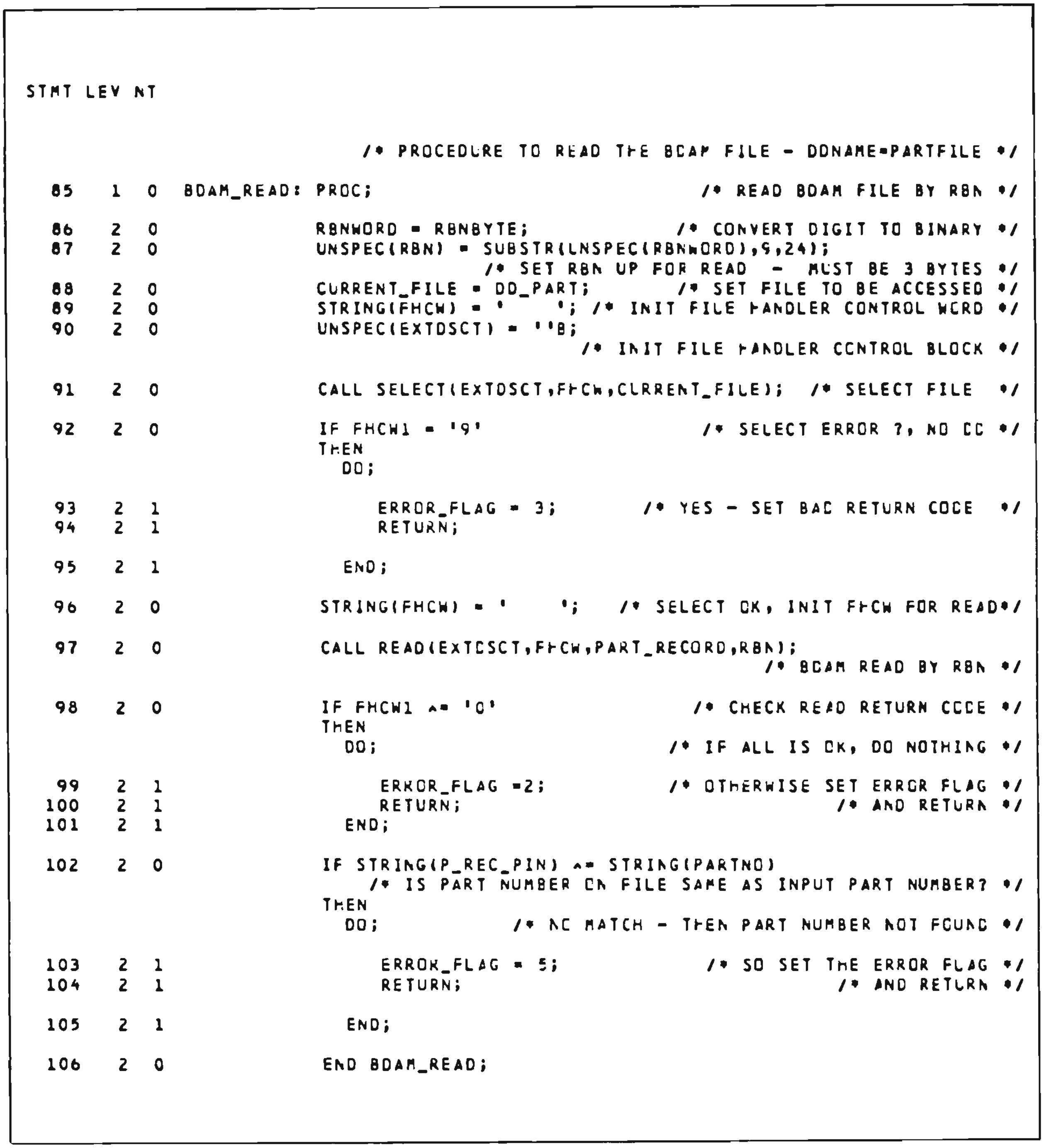
```
STMT LEV NT

                      /* ALL FILE I/O IS SUCCESSFUL - NOW BUILD THE OUTPUT MSG RESPONSE */

                          /* FIRST LETS INITIALISE THE FORMAT NAME FOR CHANGE/DISPLAY */

  44    1  1                FMTNAME = 'SSRC0001C   ';       /*    SET UP FORMAT NAME    */

                      /* NOW LETS GET THE WAREHOUSE NUMBER FROM THE INPUT AND EXPAND IT */
  45    1  1                OUTWHSNO = '  '||WHSNO;         /*    SET UP WHS NUMBER     */

                               /* OBTAIN INFORMATION FROM THE PART RECORD JUST READ */

  46    1  1                PRTDATA = STRING(P_REC_PART_DATA);
                                                /* PART DESCRIPTION TO OUTPUT AREA */
  47    1  1                PRTPRC = P_REC_PRC;              /* PART PRICE TO I/O MSG */

                              /* OBTAIN INFORMATION FROM THE STOCK RECORD JUST READ */

  48    1  1                WHSLOC = S_REC_WLC;                 /* MOVE THE LOCATION   */
  49    1  1                STKLEV = S_REC_LEV;                 /* MOVE STOCK LEVEL    */
  50    1  1                MONTH = SUBSTR((S_REC_LDT),1,2);    /* EXTRACT THE MONTH   */
  51    1  1                DAY = SUBSTR((S_REC_LDT),3,2);      /* EXTRACT THE DAY NO  */
  52    1  1                YEAR = SUBSTR((S_REC_LDT),5,2);     /* EXTRACT THE YEAR    */
  53    1  1                SLASH1, SLASH2 = '/';               /* MOVE IN THE '/'S    */
  54    1  1                LEVDATE = STRING(DATE);             /* MOVE LEVEL DATE     */
  55    1  1                STKORD = S_REC_ORD;                 /* MOVE ORDER LEVEL    */
  56    1  1                MONTH = SUBSTR((S_REC_ODT),1,2);    /* EXTACT THE MONTH    */
  57    1  1                DAY = SUBSTR((S_REC_ODT),3,2);      /* EXTRACT THE DAY NO  */
  58    1  1                YEAR = SUBSTR((S_REC_ODT),5,2);     /* EXTRACT THE YEAR    */
                                                           /* NOTE '/'S IN ALREADY */
  59    1  1                ORDDATE = STRING(DATE);             /* MOVE STOCK DATE     */

                         /* OUTPUT MESSAGE IS NOW BUILT - LETS SEND IT USING COBPUT */

  60    1  1                CALL COBPUT(OUTPUT_MESSAGE,COBPUT_RETURN);

  61    1  1                IF COBPUT_RETURN   ¬= 'CC'    /* OUTPUT QUEUING FAILURE? */
                            THEN                                             /* YES */
                              DO;

  62    1  2                    ERROR_FLAG = 2;               /*  SET SERIOUS ERROR */
  63    1  2                    LEAVE MAINLINE;               /*  THATS IT FOR NOW  */

  64    1  2                  END;

  65    1  1                END MAINLINE;          /*  END OF THE MAIN LINE ROUTINE  */
```

Figure 56.   Sample Inquiry Subsystem SQPL1 (Page 9 of 13)



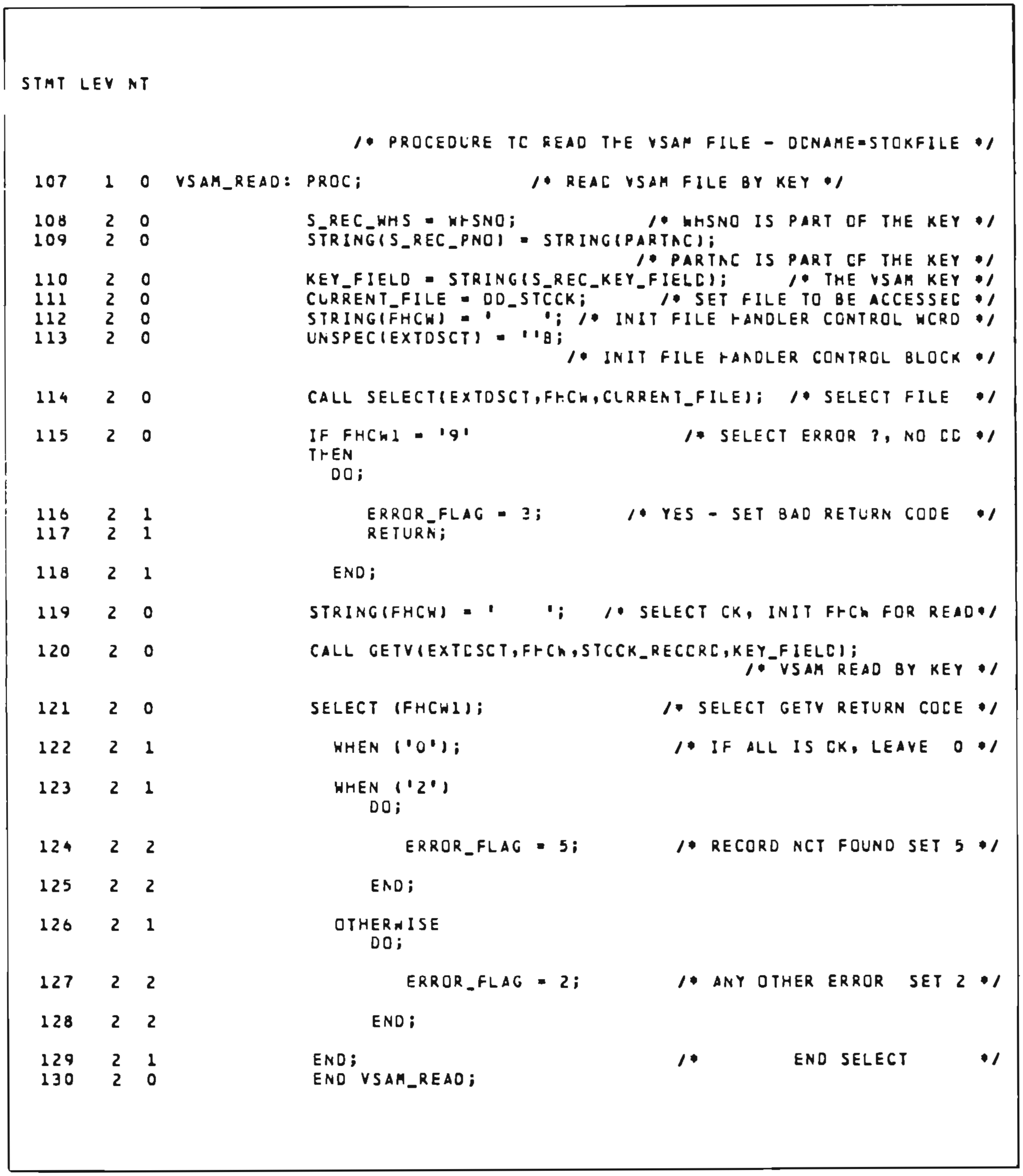
```
STMT LEV NT

                        /* CONTROL COMES HERE AFTER EXECUTION OF THE MAIN LINE ROUTINE :- */
                        /* CHECK IF ERROR_FLAG HAS BEEN SET AND IF SO SEND APPROPIATE     */
                                                                      /* ERROR RESPONSE */

  66    1  0           SELECT (ERROR_FLAG);

  67    1  1             WHEN (0);                                    /* OK, NO ACTION */

  68    1  1             WHEN (2)              /* INTERCOMM SERVICE ROUTINE FAILURE*/
                           DO;

  69    1  2               RC = 12;        /* LET INTERCOMM SEND ERROR MESSAGE*/

  70    1  2             END;

  71    1  1             WHEN (3)     /* FILE COULD NOT BE SELECTED - NO DDCARD? */
                           DO;

  72    1  2               ERRORTXT = CURRENT_FILE||
                             ' - FILE COULD NOT BE SELECTED';  /* SET TEXT */
  73    1  2               CALL SEND_ERR_MSG;      /* SEND THE ERROR MESSAGE */

  74    1  2             END;

  75    1  1             WHEN (5)                        /* RECORD NOT FOUND IN FILE */
                           DO;

  76    1  2               ERRORTXT = 'PART '||STRING(PARTNO)||
                             ' NOT FOUND';                             /* SET TEXT */

  77    1  2               IF CURRENT_FILE = DC_STOCK
                           THEN
                             DO;      /* SUPPLEMENT TEXT IF STOCK FILE ERROR */

  78    1  3                   ERRORTXT = SUBSTR((ERRORTXT),1,20)||
                                 ' IN WAREHOUSE '||WHSNO;      /* RESET TEXT*/

  79    1  3                 END;

  80    1  2               ELSE;

  81    1  2               CALL SEND_ERR_MSG;      /* SEND THE ERROR MESSAGE */

  82    1  2             END;

  83    1  1           END;                          /*        END SELECT      */

  84    1  0           RETURN;                       /* LEAVE SQPL1 - ALL DONE */
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 10 of 13)



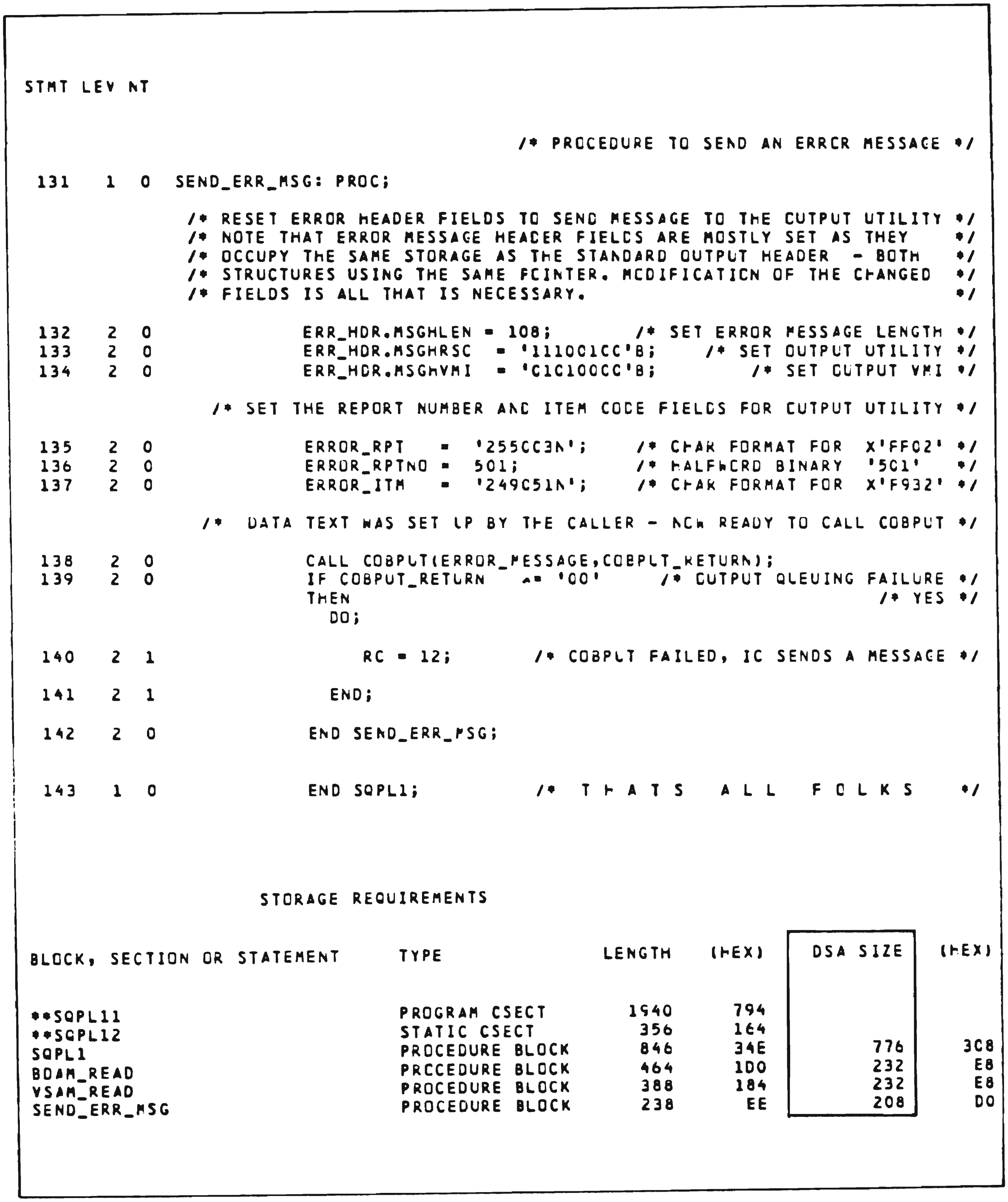
```
STMT LEV NT

                                      /* PROCEDURE TO READ THE BDAM FILE - DDNAME=PARTFILE */

  85    1   0  BDAM_READ: PROC;                               /* READ BDAM FILE BY RBN */

  86    2   0             RBNWORD = RBNBYTE;              /* CONVERT DIGIT TO BINARY */
  87    2   0             UNSPEC(RBN) = SUBSTR(UNSPEC(RBNWORD),9,24);
                                       /* SET RBN UP FOR READ  -  MUST BE 3 BYTES */
  88    2   0             CURRENT_FILE = DD_PART;         /* SET FILE TO BE ACCESSED */
  89    2   0             STRING(FHCW) = '    '; /* INIT FILE HANDLER CONTROL WORD */
  90    2   0             UNSPEC(EXTDSCT) = ''B;
                                              /* INIT FILE HANDLER CONTROL BLOCK */

  91    2   0             CALL SELECT(EXTDSCT,FHCW,CURRENT_FILE);  /* SELECT FILE  */

  92    2   0             IF FHCW1 = '9'                  /* SELECT ERROR ?, NO DC */
                          THEN
                            DO;

  93    2   1                 ERROR_FLAG = 3;       /* YES - SET BAD RETURN CODE  */
  94    2   1                 RETURN;

  95    2   1               END;

  96    2   0             STRING(FHCW) = '    ';   /* SELECT OK, INIT FHCW FOR READ*/

  97    2   0             CALL READ(EXTDSCT,FHCW,PART_RECORD,RBN);
                                                             /* BDAM READ BY RBN */

  98    2   0             IF FHCW1 ¬= '0'                /* CHECK READ RETURN CODE */
                          THEN
                            DO;                        /* IF ALL IS OK, DO NOTHING */

  99    2   1                 ERROR_FLAG =2;           /* OTHERWISE SET ERROR FLAG */
 100    2   1                 RETURN;                               /* AND RETURN */
 101    2   1               END;

 102    2   0             IF STRING(P_REC_PIN) ¬= STRING(PARTNO)
                              /* IS PART NUMBER ON FILE SAME AS INPUT PART NUMBER? */
                          THEN
                            DO;          /* NO MATCH - THEN PART NUMBER NOT FOUND */

 103    2   1                 ERROR_FLAG = 5;           /* SO SET THE ERROR FLAG */
 104    2   1                 RETURN;                               /* AND RETURN */

 105    2   1               END;

 106    2   0             END BDAM_READ;
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 11 of 13)



```
 STMT LEV NT

                                         /* PROCEDURE TO READ THE VSAM FILE - DDNAME=STOKFILE */

  107   1  0  VSAM_READ: PROC;                /* READ VSAM FILE BY KEY */

  108   2  0             S_REC_WHS = WHSNO;             /* WHSNO IS PART OF THE KEY */
  109   2  0             STRING(S_REC_PNO) = STRING(PARTNO);
                                                        /* PARTNO IS PART OF THE KEY */
  110   2  0             KEY_FIELD = STRING(S_REC_KEY_FIELD);     /* THE VSAM KEY */
  111   2  0             CURRENT_FILE = DD_STOCK;       /* SET FILE TO BE ACCESSED */
  112   2  0             STRING(FHCW) = '    '; /* INIT FILE HANDLER CONTROL WORD */
  113   2  0             UNSPEC(EXTDSCT) = ''B;
                                             /* INIT FILE HANDLER CONTROL BLOCK */

  114   2  0             CALL SELECT(EXTDSCT,FHCW,CURRENT_FILE);  /* SELECT FILE  */

  115   2  0             IF FHCW1 = '9'                 /* SELECT ERROR ?, NO DD */
                         THEN
                           DO;

  116   2  1                 ERROR_FLAG = 3;        /* YES - SET BAD RETURN CODE  */
  117   2  1                 RETURN;

  118   2  1               END;

  119   2  0             STRING(FHCW) = '    ';   /* SELECT OK, INIT FHCW FOR READ*/

  120   2  0             CALL GETV(EXTDSCT,FHCW,STOCK_RECORD,KEY_FIELD);
                                                              /* VSAM READ BY KEY */

  121   2  0             SELECT (FHCW1);                /* SELECT GETV RETURN CODE */

  122   2  1               WHEN ('0');                  /* IF ALL IS OK, LEAVE  0 */

  123   2  1               WHEN ('2')
                              DO;

  124   2  2                    ERROR_FLAG = 5;         /* RECORD NOT FOUND SET 5 */

  125   2  2                 END;

  126   2  1               OTHERWISE
                              DO;

  127   2  2                    ERROR_FLAG = 2;         /* ANY OTHER ERROR  SET 2 */

  128   2  2                 END;

  129   2  1             END;                         /*        END SELECT       */
  130   2  0             END VSAM_READ;
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 12 of 13)

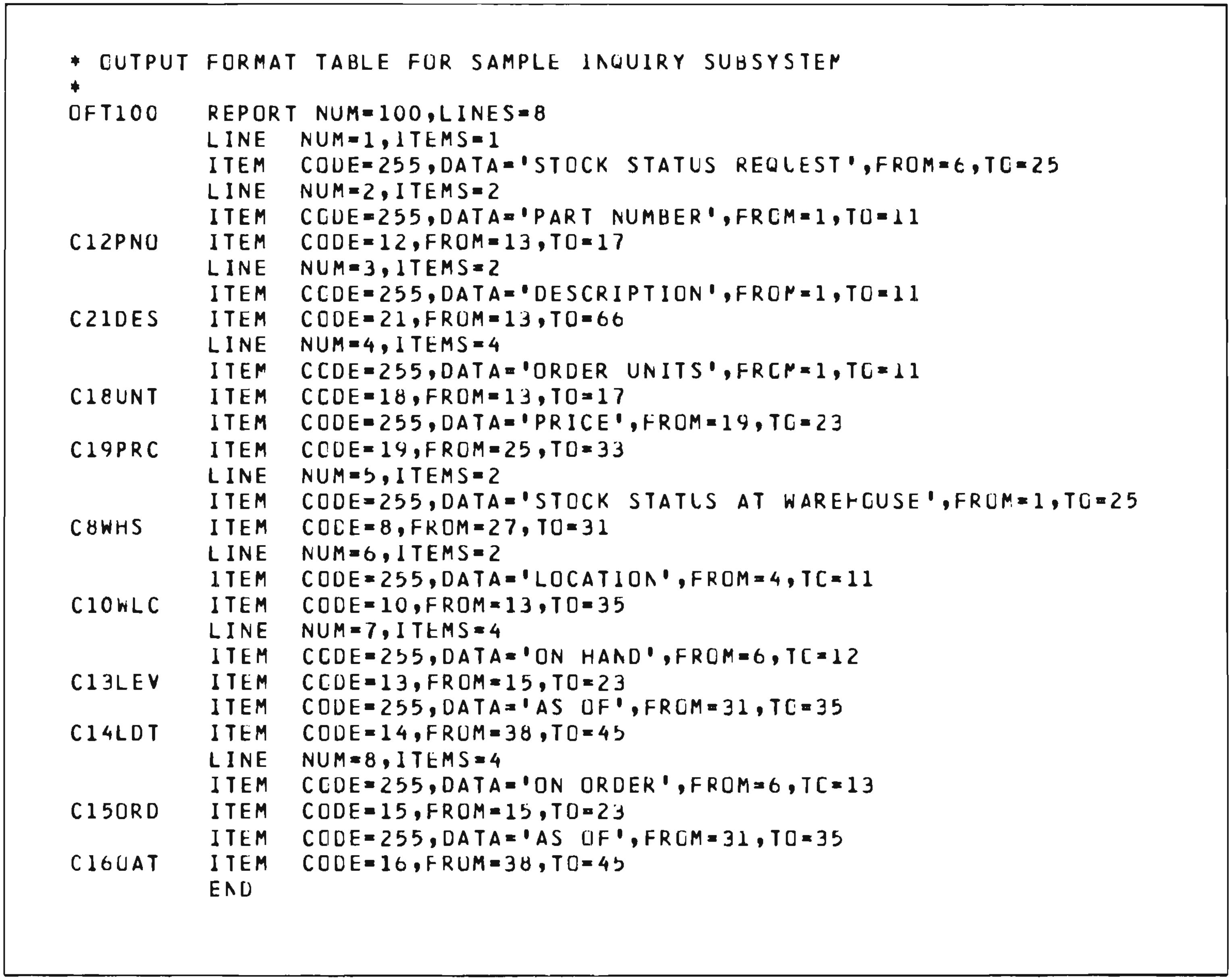


```
STMT LEV NT

                                           /* PROCEDURE TO SEND AN ERROR MESSAGE */

 131   1  0  SEND_ERR_MSG: PROC;

               /* RESET ERROR HEADER FIELDS TO SEND MESSAGE TO THE OUTPUT UTILITY */
               /* NOTE THAT ERROR MESSAGE HEADER FIELDS ARE MOSTLY SET AS THEY    */
               /* OCCUPY THE SAME STORAGE AS THE STANDARD OUTPUT HEADER  - BOTH   */
               /* STRUCTURES USING THE SAME POINTER. MODIFICATION OF THE CHANGED  */
               /* FIELDS IS ALL THAT IS NECESSARY.                                */

 132   2  0            ERR_HDR.MSGHLEN = 108;        /* SET ERROR MESSAGE LENGTH */
 133   2  0            ERR_HDR.MSGHRSC  = '11100100'B;    /* SET OUTPUT UTILITY */
 134   2  0            ERR_HDR.MSGHVMI  = '01010000'B;        /* SET OUTPUT VMI */

                 /* SET THE REPORT NUMBER AND ITEM CODE FIELDS FOR OUTPUT UTILITY */

 135   2  0            ERROR_RPT    =  '255003N';     /* CHAR FORMAT FOR  X'FF02' */
 136   2  0            ERROR_RPTNO =  501;           /* HALFWORD BINARY  '501'   */
 137   2  0            ERROR_ITM    =  '249051N';     /* CHAR FORMAT FOR  X'F932' */

                /*  DATA TEXT WAS SET UP BY THE CALLER - NOW READY TO CALL COBPUT */

 138   2  0            CALL COBPUT(ERROR_MESSAGE,COBPUT_RETURN);
 139   2  0            IF COBPUT_RETURN   ¬= '00'     /* OUTPUT QUEUING FAILURE */
                       THEN                                              /* YES */
                         DO;

 140   2  1                RC = 12;       /* COBPUT FAILED, IC SENDS A MESSAGE */

 141   2  1              END;

 142   2  0            END SEND_ERR_MSG;


 143   1  0            END SQPL1;         /*  THATS  ALL  FOLKS    */



                      STORAGE REQUIREMENTS

BLOCK, SECTION OR STATEMENT     TYPE                LENGTH    (HEX)    DSA SIZE    (HEX)


**SQPL11                        PROGRAM CSECT         1940      794
**SQPL12                        STATIC CSECT           356      164
SQPL1                           PROCEDURE BLOCK        846      34E         776      308
BDAM_READ                       PROCEDURE BLOCK        464      1D0         232       E8
VSAM_READ                       PROCEDURE BLOCK        388      184         232       E8
SEND_ERR_MSG                    PROCEDURE BLOCK        238       EE         208       D0
```

Figure 56. Sample Inquiry Subsystem SQPL1 (Page 13 of 13)

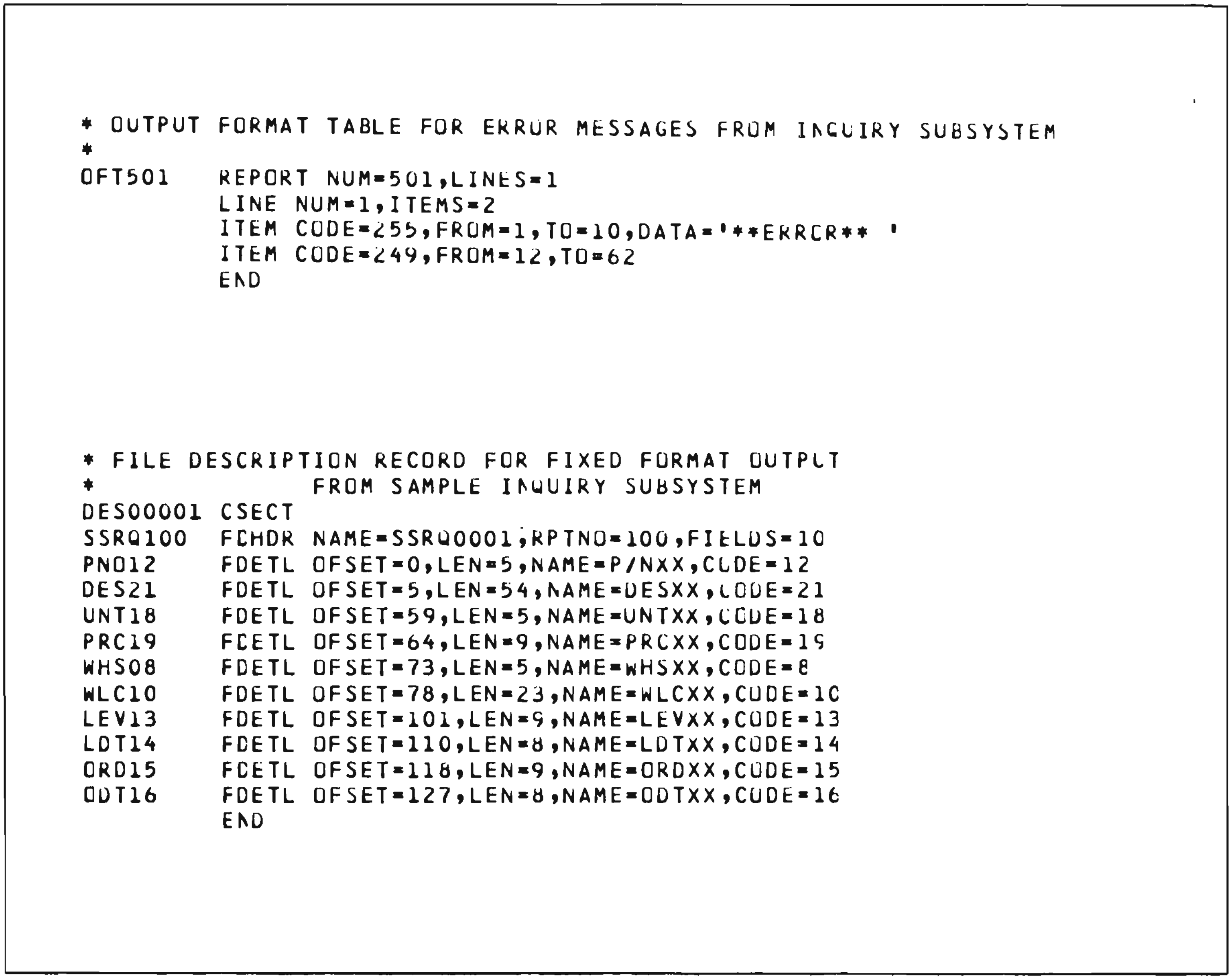
```
//TABLES      JOB
//*
//*                 DEFINE SYCTTBL FOR SUBSYSTEM
//*
//STEP1       EXEC  LIBELINK,Q=TEST,NAME=INTSCT,LMOD=INTSCT
//LIB.SYSIN   DD    *
./ ADD NAME=USRSCTS
./ NUMBER     NEW1=100,INCR=100
USRSCTS       DS    0H
RP            SYCTTBL SUBH=R,SUBC=P,SBSP=SQPL1,LANG=RPL1,OVLY=0,        X
                    NUMCL=10,MNCL=1,TCTV=60,SPACE=4096
/*
//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    0H
RTRPECT       VERB  RTRP,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 PL/1 PGM. GUIDE S/S
              DC    F'0'
              PMISTOP
              END
//
```

Figure 57.   Table Updates to Implement Test Mode Testing



```
* CUTPUT FORMAT TABLE FOR SAMPLE 1NQUIRY SUBSYSTEM
*
OFT100   REPORT NUM=100,LINES=8
         LINE  NUM=1,ITEMS=1
         ITEM  CODE=255,DATA='STOCK STATUS REQLEST',FROM=6,TC=25
         LINE  NUM=2,ITEMS=2
         ITEM  CGDE=255,DATA='PART NUMBER',FRCM=1,TO=11
C12PNO   ITEM  CODE=12,FROM=13,TO=17
         LINE  NUM=3,ITEMS=2
         ITEM  CCDE=255,DATA='DESCRIPTION',FROM=1,TO=11
C21DES   ITEM  CODE=21,FROM=13,TO=66
         LINE  NUM=4,ITEMS=4
         ITEM  CCDE=255,DATA='ORDER UNITS',FRCM=1,TC=11
C18UNT   ITEM  CCDE=18,FROM=13,TO=17
         ITEM  CODE=255,DATA='PRICE',FROM=19,TC=23
C19PRC   ITEM  CCDE=19,FROM=25,TO=33
         LINE  NUM=5,ITEMS=2
         ITEM  CODE=255,DATA='STOCK STATLS AT WAREHCUSE',FROM=1,TC=25
C8WHS    ITEM  CODE=8,FROM=27,TO=31
         LINE  NUM=6,ITEMS=2
         ITEM  CODE=255,DATA='LOCATION',FROM=4,TC=11
C10WLC   ITEM  CODE=10,FROM=13,TO=35
         LINE  NUM=7,ITEMS=4
         ITEM  CCDE=255,DATA='ON HAND',FROM=6,TC=12
C13LEV   ITEM  CCDE=13,FROM=15,TO=23
         ITEM  CODE=255,DATA='AS OF',FROM=31,TC=35
C14LDT   ITEM  CODE=14,FROM=38,TO=45
         LINE  NUM=8,ITEMS=4
         ITEM  CCDE=255,DATA='ON ORDER',FROM=6,TC=13
C15ORD   ITEM  CODE=15,FROM=15,TO=23
         ITEM  CODE=255,DATA='AS OF',FROM=31,TO=35
C16UAT   ITEM  CODE=16,FRUM=38,TO=45
         END
```

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



```
* OUTPUT FORMAT TABLE FOR ERRUR MESSAGES FROM INCUIRY SUBSYSTEM
*
OFT501   REPORT NUM=501,LINES=1
         LINE NUM=1,ITEMS=2
         ITEM CODE=255,FROM=1,TO=10,DATA='**ERRCR** '
         ITEM CODE=249,FROM=12,TO=62
         END




* FILE DESCRIPTION RECORD FOR FIXED FORMAT OUTPLT
*                    FROM SAMPLE INQUIRY SUBSYSTEM
DES00001 CSECT
SSRQ100  FCHDR NAME=SSRQ0001,RPTNO=100,FIELDS=10
PND12    FDETL OFSET=0,LEN=5,NAME=P/NXX,CLDE=12
DES21    FDETL OFSET=5,LEN=54,NAME=DESXX,CODE=21
UNT18    FDETL OFSET=59,LEN=5,NAME=UNTXX,CODE=18
PRC19    FCETL OFSET=64,LEN=9,NAME=PRCXX,CODE=19
WHS08    FDETL OFSET=73,LEN=5,NAME=WHSXX,CODE=8
WLC10    FDETL OFSET=78,LEN=23,NAME=WLCXX,CUDE=10
LEV13    FDETL OFSET=101,LEN=9,NAME=LEVXX,CUDE=13
LDT14    FCETL OFSET=110,LEN=8,NAME=LDTXX,CUDE=14
ORD15    FCETL OFSET=118,LEN=9,NAME=ORDXX,CUDE=15
ODT16    FDETL OFSET=127,LEN=8,NAME=ODTXX,CUDE=16
         END
```

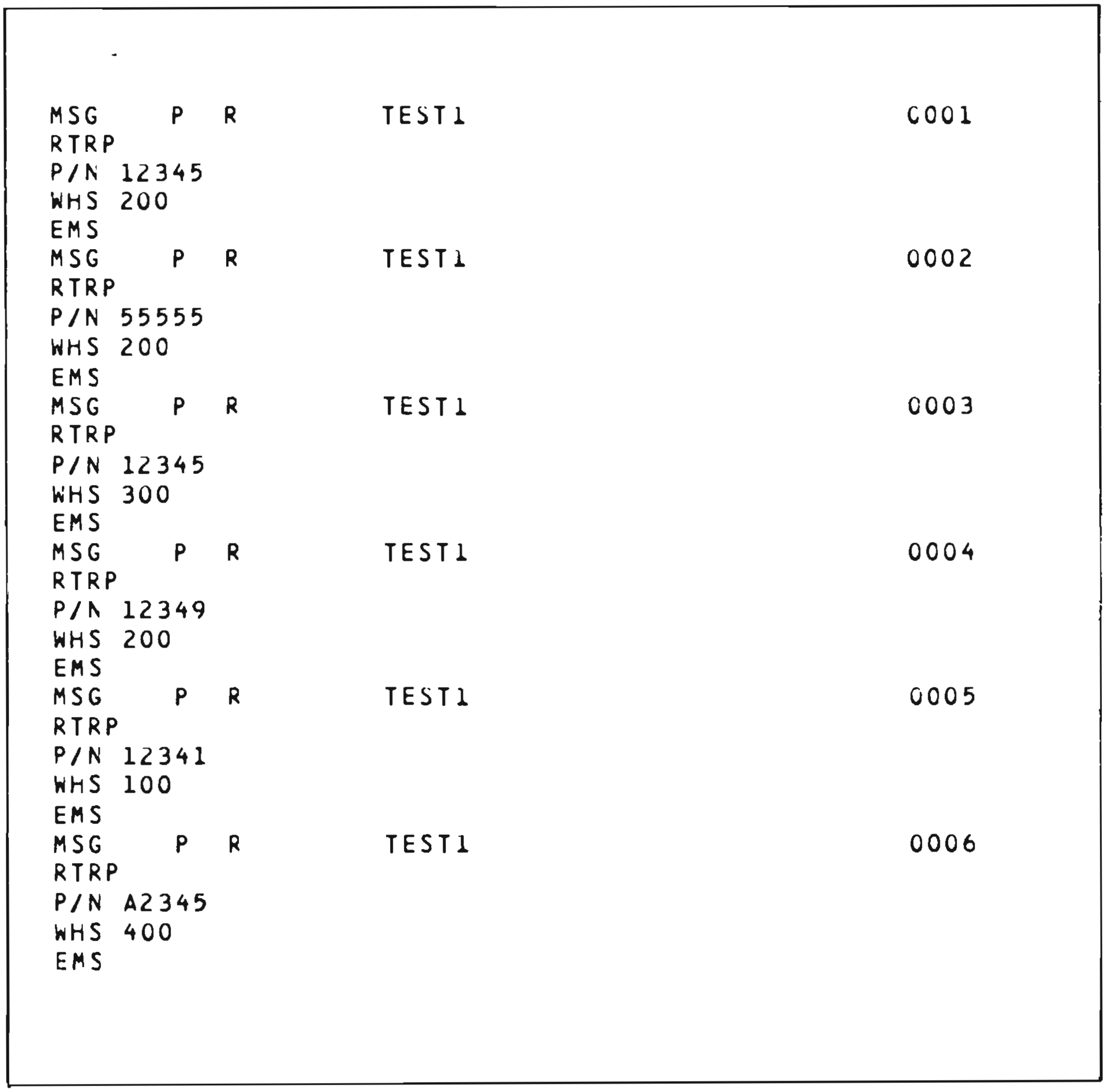
Figure 58. Utilities Table Coding for Test Mode Subsystem (Page 2 of 2)

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. Contents of Card — Ending Character EMS — EOT (X'37') EOT — EOT (X'37') ETX — ETX (X'03') ETB — ETB (X'26')

*3-digit integer values (from 000 to 255) or a corresponding single alphanumeric character in low-order field position.

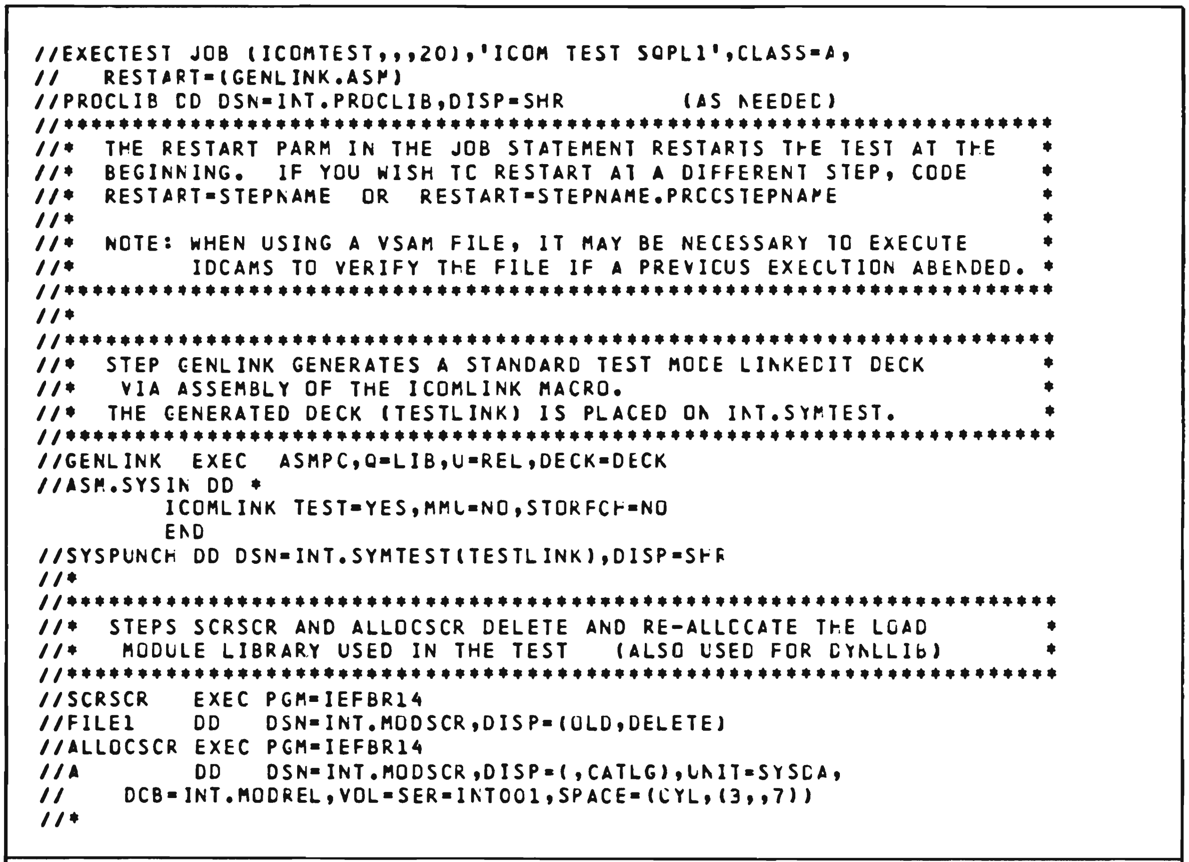
**64 is default maximum. See the Operating Reference Manual if necessary to alter this specification.

Figure 59. Test Mode Message Card Formats



```
MSG    P  R        TEST1                         C001
RTRP
P/N 12345
WHS 200
EMS
MSG    P  R        TEST1                         0002
RTRP
P/N 55555
WHS 200
EMS
MSG    P  R        TEST1                         0003
RTRP
P/N 12345
WHS 300
EMS
MSG    P  R        TEST1                         0004
RTRP
P/N 12349
WHS 200
EMS
MSG    P  R        TEST1                         0005
RTRP
P/N 12341
WHS 100
EMS
MSG    P  R        TEST1                         0006
RTRP
P/N A2345
WHS 400
EMS
```

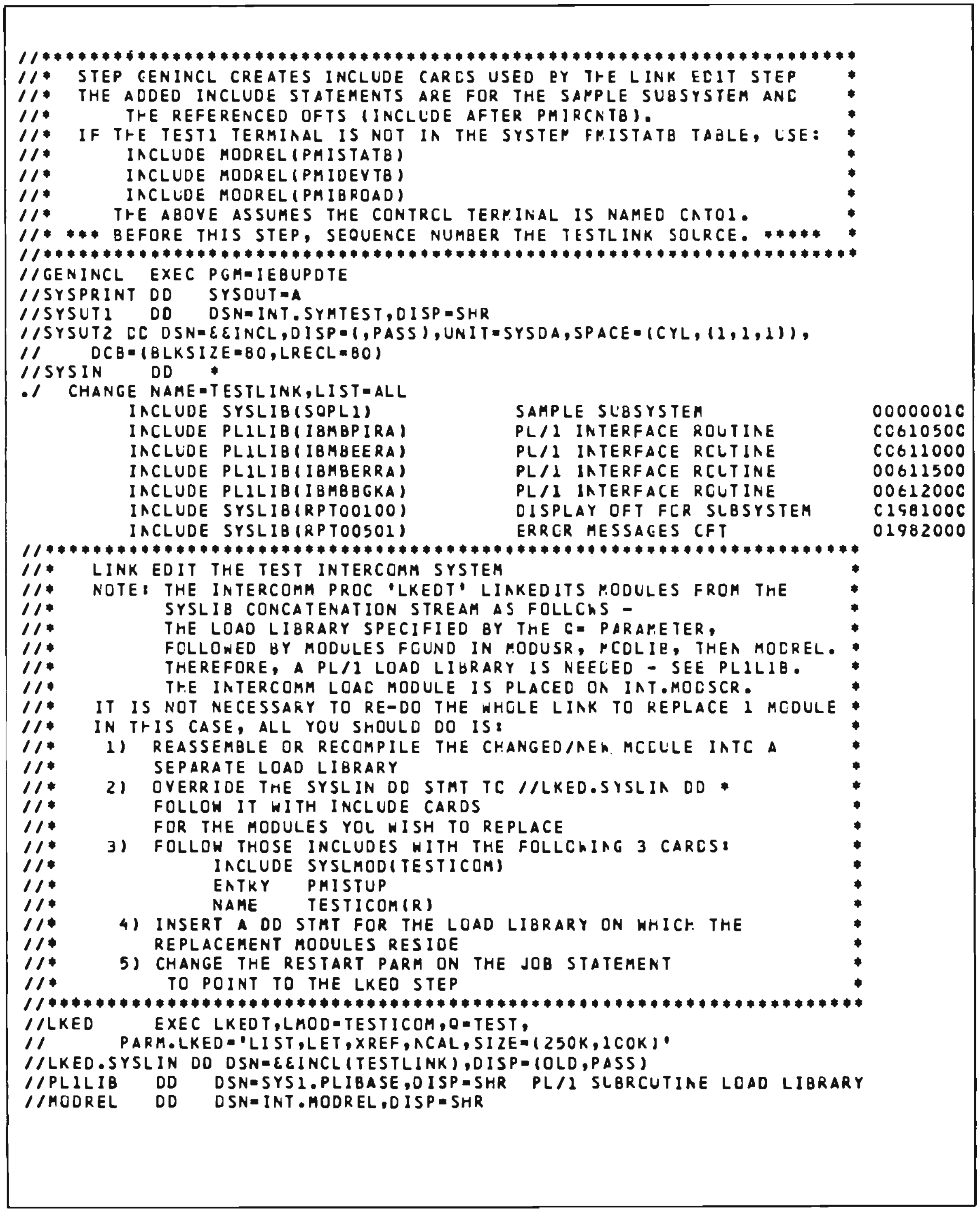
Figure 60. Sample Input Test Messages for Test Mode



```
//EXECTEST JOB (ICOMTEST,,,20),'ICOM TEST SQPL1',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  ASMPC,Q=LIB,U=REL,DECK=DECK
//ASM.SYSIN DD *
         ICOMLINK TEST=YES,MML=NO,STORFCH=NO
         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=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=(CYL,(3,,7))
//*
```

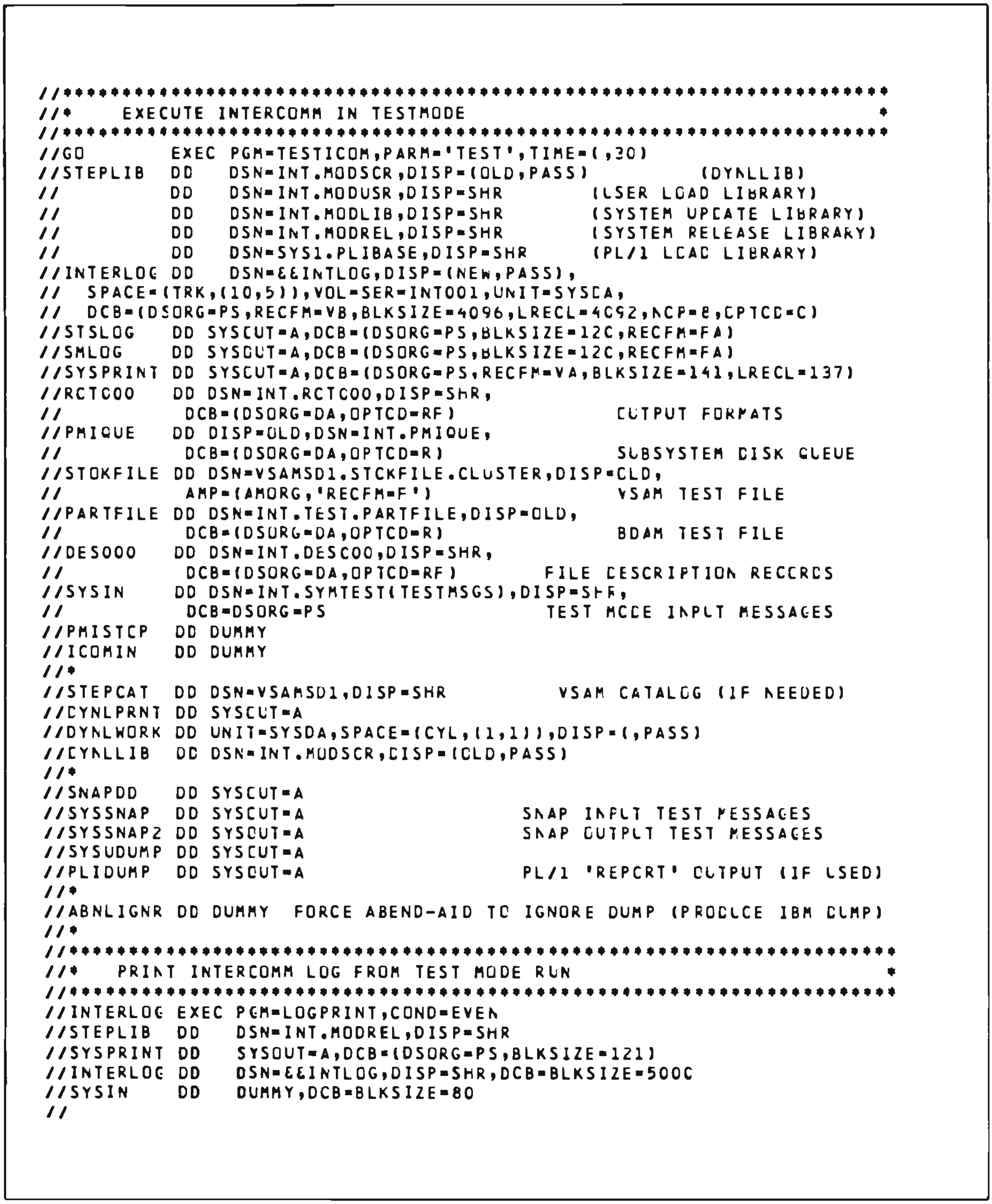
NOTE: JCL requirements vary by installation requirements. The above example illustrates representative JCL. The installation System Manager should verify JCL to use.

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



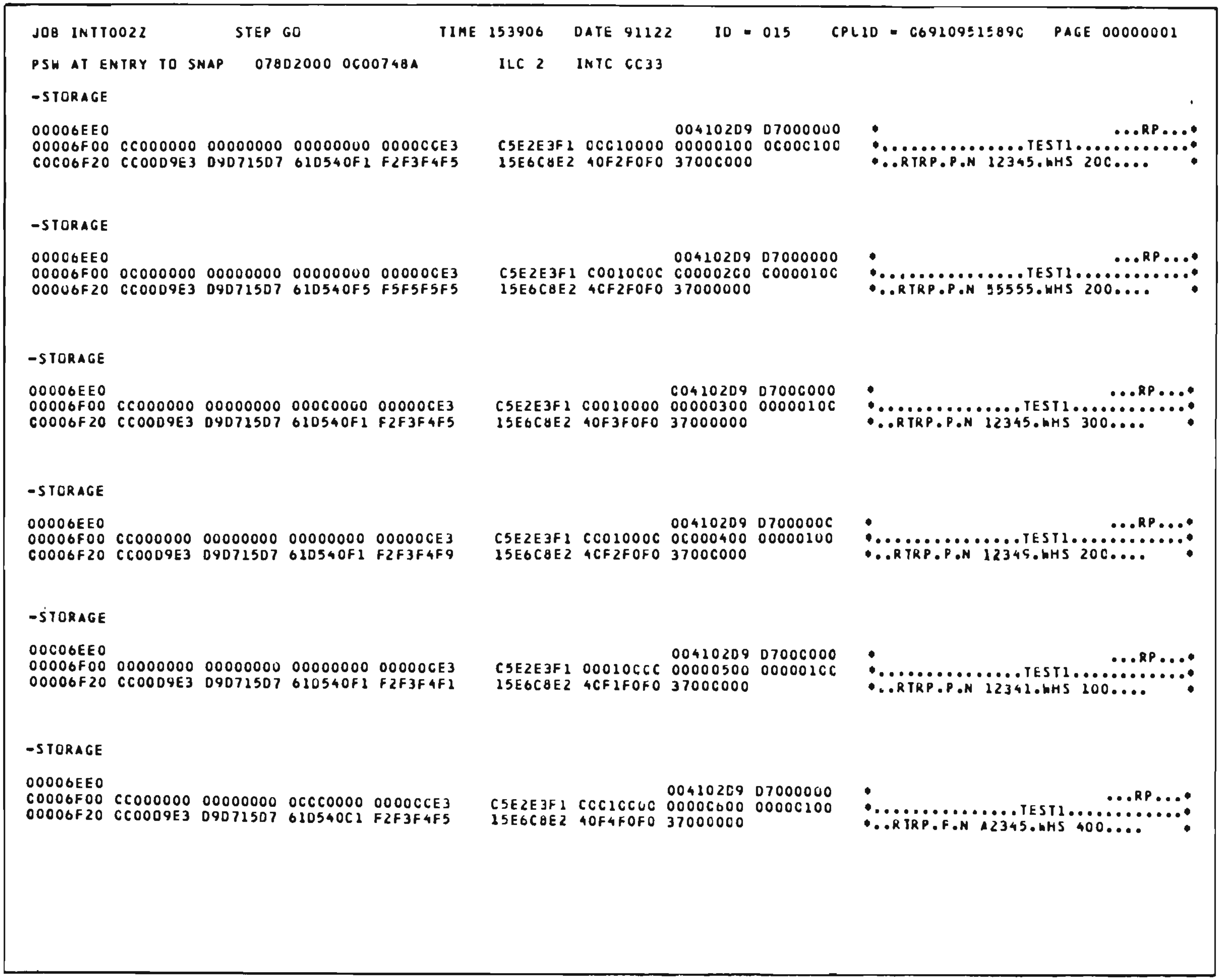
```
//**********************************************************************
//*  STEP GENINCL CREATES INCLUDE CARDS USED BY THE LINK EDIT STEP     *
//*  THE ADDED INCLUDE STATEMENTS ARE FOR THE SAMPLE SUBSYSTEM AND     *
//*      THE REFERENCED OFTS (INCLUDE AFTER PMIRCNTB).                 *
//*  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.SYMTEST,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(SQPL1)           SAMPLE SUBSYSTEM               00000010
         INCLUDE PL1LIB(IBMBPIRA)        PL/1 INTERFACE ROUTINE         00610500
         INCLUDE PL1LIB(IBMBEERA)        PL/1 INTERFACE ROUTINE         00611000
         INCLUDE PL1LIB(IBMBERRA)        PL/1 INTERFACE ROUTINE         00611500
         INCLUDE PL1LIB(IBMBBGKA)        PL/1 INTERFACE ROUTINE         00612000
         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.  *
//*         THEREFORE, A PL/1 LOAD LIBRARY IS NEEDED - SEE PL1LIB.     *
//*         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)
//PL1LIB   DD   DSN=SYS1.PLIBASE,DISP=SHR  PL/1 SUBROUTINE LOAD LIBRARY
//MODREL   DD   DSN=INT.MODREL,DISP=SHR
```

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



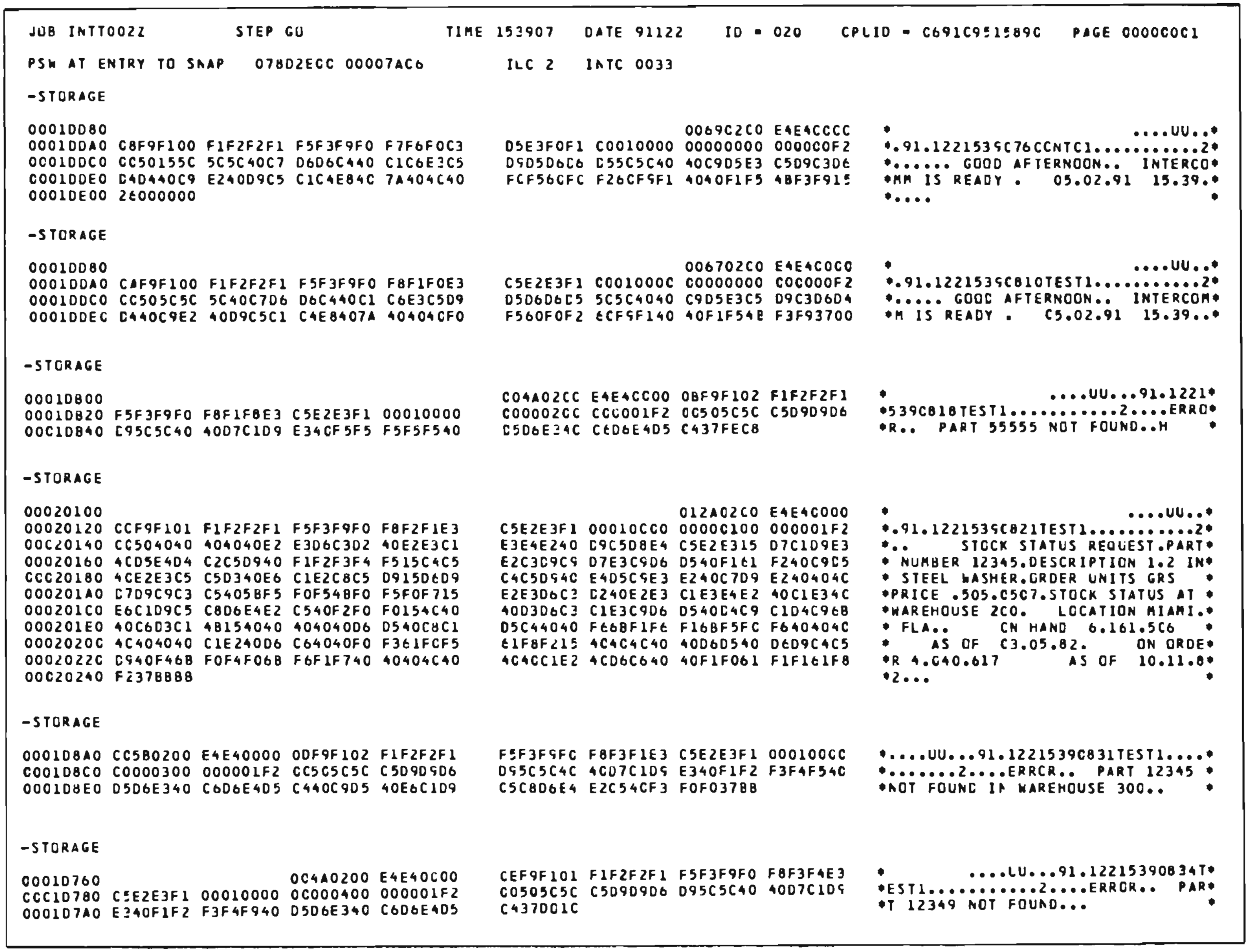
```
//*********************************************************************
//*    EXECUTE INTERCOMM IN TESTMODE                                  *
//*********************************************************************
//GO        EXEC PGM=TESTICOM,PARM='TEST',TIME=(,30)
//STEPLIB   DD   DSN=INT.MODSCR,DISP=(OLD,PASS)          (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.PLIBASE,DISP=SHR      (PL/1 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)
//RCTC00    DD DSN=INT.RCTC00,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.STOKFILE.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.DESC00,DISP=SHR,
//             DCB=(DSORG=DA,OPTCD=RF)        FILE DESCRIPTION RECORDS
//SYSIN     DD DSN=INT.SYMTEST(TESTMSGS),DISP=SHR,
//             DCB=DSORG=PS                   TEST MODE INPUT MESSAGES
//PMISTOP   DD DUMMY
//ICOMIN    DD DUMMY
//*
//STEPCAT   DD DSN=VSAMSD1,DISP=SHR           VSAM CATALOG (IF NEEDED)
//DYNLPRNT DD SYSOUT=A
//DYNLWORK DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(,PASS)
//DYNLLIB   DD DSN=INT.MODSCR,DISP=(OLD,PASS)
//*
//SNAPDD    DD SYSOUT=A
//SYSSNAP   DD SYSOUT=A                     SNAP INPUT TEST MESSAGES
//SYSSNAP2 DD SYSOUT=A                     SNAP OUTPUT TEST MESSAGES
//SYSUDUMP DD SYSOUT=A
//PLIDUMP   DD SYSOUT=A                     PL/1 'REPORT' OUTPUT (IF USED)
//*
//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)



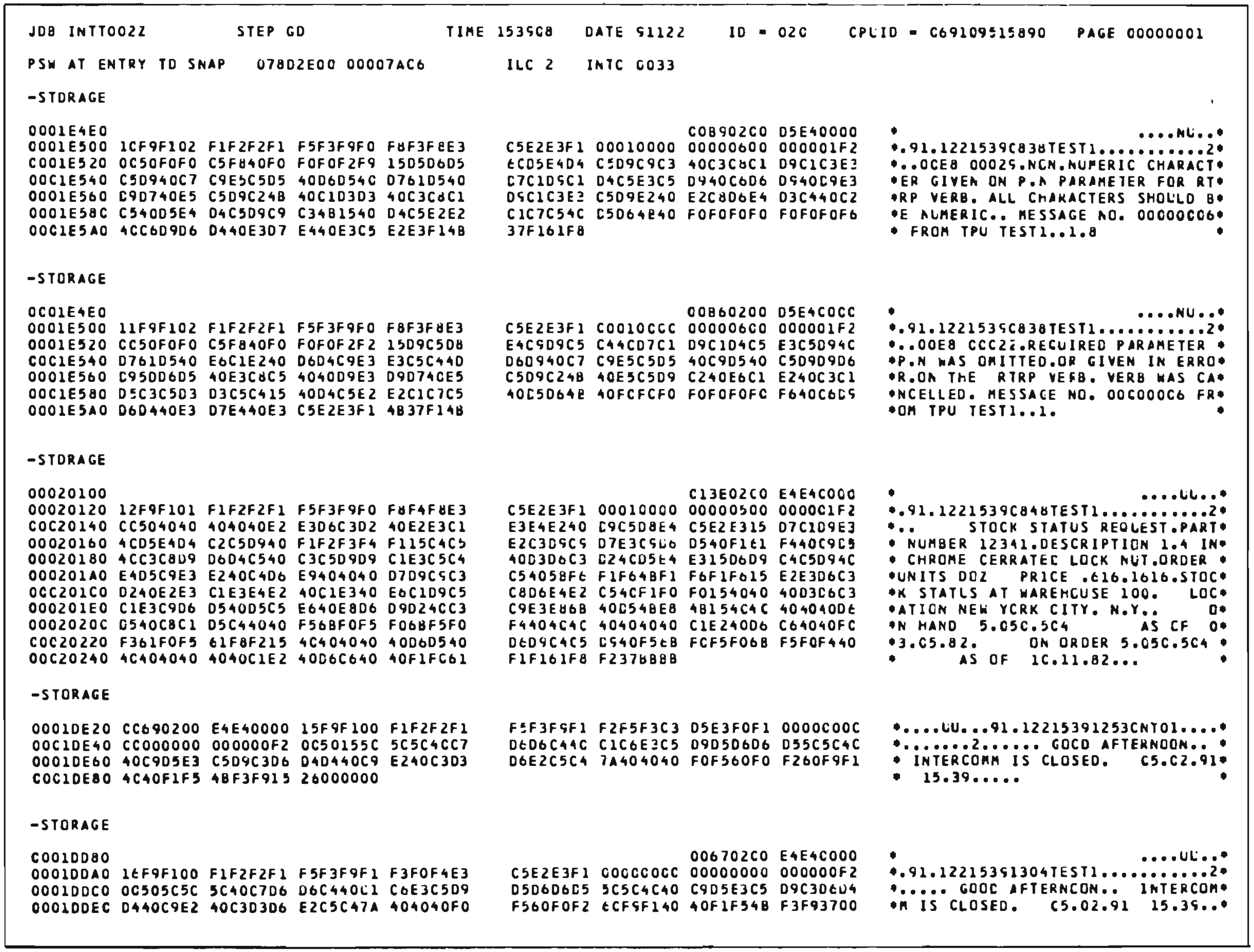
```
JOB INTT0022          STEP GO              TIME 153906   DATE 91122    ID = 015    CPLID = C6910951589C   PAGE 00000001

PSW AT ENTRY TO SNAP   078D2000 0C00748A         ILC 2   INTC CC33

-STORAGE

00006EE0                                                      004102D9 D7000000   *                          ...RP...*
00006F00 CC000000 00000000 00000000 0000CCE3   C5E2E3F1 0CC10000 00000100 0C000100   *..............TEST1............*
C0C06F20 CC00D9E3 D9D715D7 61D540F1 F2F3F4F5   15E6C8E2 40F2F0F0 3700C000            *..RTRP.P.N 12345.WHS 20C....    *


-STORAGE

00006EE0                                                      004102D9 D7000000   *                          ...RP...*
00006F00 0C000000 00000000 00000000 00000CE3   C5E2E3F1 C0010C0C C00002C0 C000010C   *..............TEST1............*
000U6F20 CC00D9E3 D9D715D7 61D540F5 F5F5F5F5   15E6C8E2 4CF2F0F0 37000000            *..RTRP.P.N 55555.WHS 200....    *


-STORAGE

00006EE0                                                      C04102D9 D700C000   *                          ...RP...*
00006F00 CC000000 00000000 000C00G0 00000CE3   C5E2E3F1 C0010000 00000300 0000010C   *..............TEST1............*
C0006F20 CC00D9E3 D9D715D7 61D540F1 F2F3F4F5   15E6C8E2 40F3F0F0 37000000            *..RTRP.P.N 12345.WHS 300....    *


-STORAGE

00006EE0                                                      004102D9 D700000C   *                          ...RP...*
00006F00 CC000000 00000000 00000000 00000CE3   C5E2E3F1 CC01000C 0C000400 00000100   *..............TEST1............*
C0006F20 CC00D9E3 D9D715D7 61D540F1 F2F3F4F9   15E6C8E2 4CF2F0F0 3700C000            *..RTRP.P.N 12349.WHS 20C....    *


-STORAGE

00C06EE0                                                      004102D9 D700C000   *                          ...RP...*
00006F00 00000000 00000000 00000000 00000CE3   C5E2E3F1 00010CCC 00000500 000001CC   *..............TEST1............*
00006F20 CC00D9E3 D9D715D7 61D540F1 F2F3F4F1   15E6C8E2 4CF1F0F0 3700C000            *..RTRP.P.N 12341.WHS 100....    *


-STORAGE

00006EE0                                                      004102D9 D7000000   *                          ...RP...*
C0006F00 CC000000 00000000 0CCC0000 0000CCE3   C5E2E3F1 CCC1CC0C 0000C600 0000C100   *..............TEST1............*
00006F20 CC00D9E3 D9D715D7 61D540C1 F2F3F4F5   15E6C8E2 40F4F0F0 37000000            *..RTRP.F.N A2345.WHS 400....    *
```

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



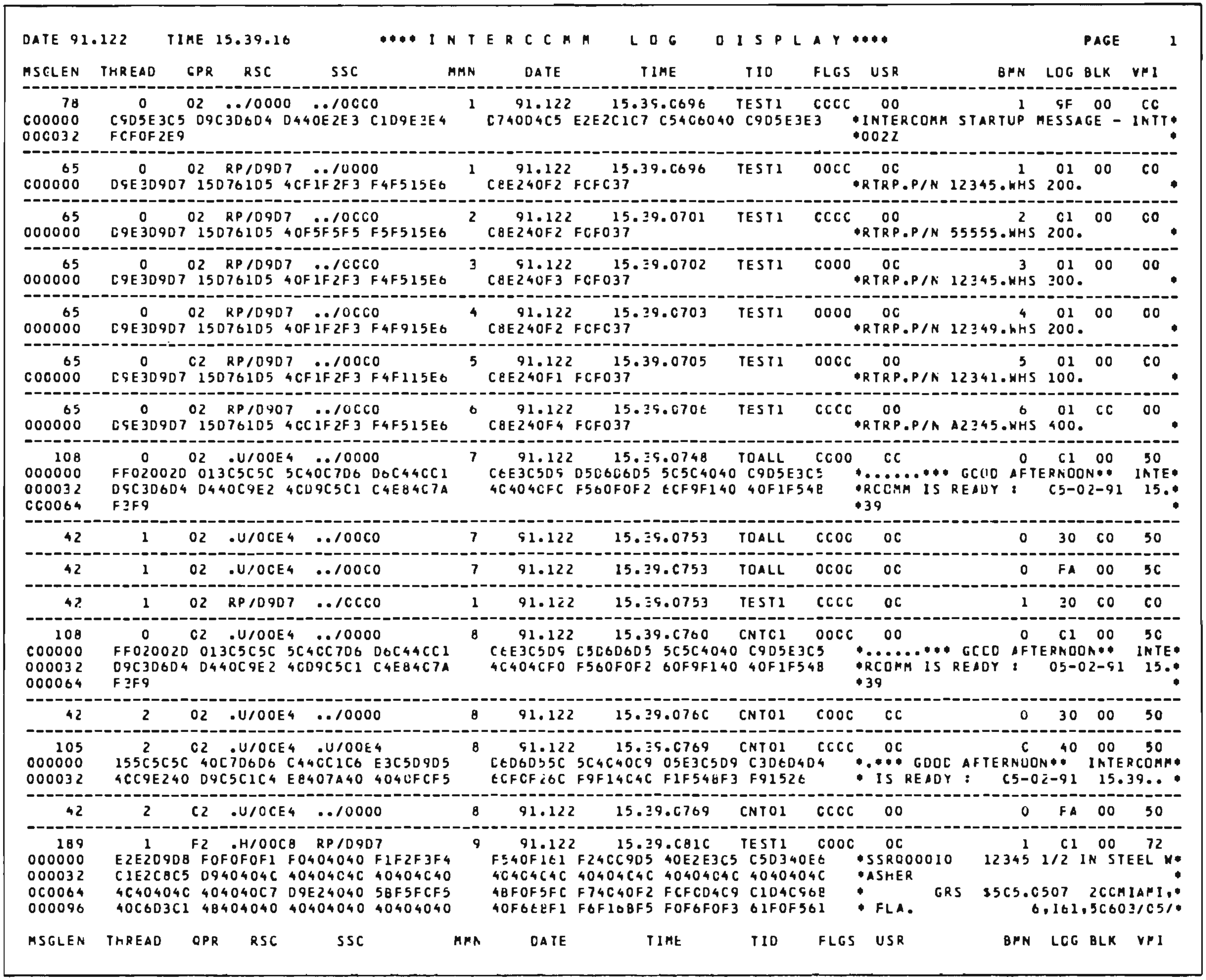
```
JOB INTT002Z          STEP GU                  TIME 153907   DATE 91122    ID = 020    CPLID = C691C951589C   PAGE 0000C0C1

PSW AT ENTRY TO SNAP   078D2ECC 00007AC6          ILC 2   INTC 0033

-STORAGE

0001DD80                                                        0069C2C0 E4E4CCCC   *                        ....UU..*
0001DDA0 C8F9F100 F1F2F2F1 F5F3F9F0 F7F6F0C3    D5E3F0F1 C0010000 00000000 0000C0F2   *.91.1221539C76CCNTC1...........2*
0C01DDC0 CC50155C 5C5C40C7 D6D6C440 C1C6E3C5    D9D5D6C6 C55C5C40 40C9D5E3 C5D9C3D6   *...... GOOD AFTERNOON..  INTERCO*
C001DDE0 C4D440C9 E240D9C5 C1C4E84C 7A404C40    FCF56CFC F26CF9F1 4040F1F5 4BF3F915   *MM IS READY .   05.02.91  15.39.*
0001DE00 26000000                                                                      *....                           *

-STORAGE

0001DD80                                                        006702C0 E4E4C0C0   *                        ....UU..*
0001DDA0 CAF9F100 F1F2F2F1 F5F3F9F0 F8F1F0E3    C5E2E3F1 C001000C C0000000 C0C000F2   *.91.1221539C810TEST1...........2*
0001DDC0 CC505C5C 5C40C7D6 D6C440C1 C6E3C5D9    D5D6D6C5 5C5C4040 C9D5E3C5 D9C3D6D4   *..... GOOC AFTERNOON..  INTERCOM*
0001DDEC C440C9E2 40D9C5C1 C4E8407A 40404CF0    F560F0F2 6CF9F140 40F1F54B F3F93700   *M IS READY .   C5.02.91  15.39..*


-STORAGE

0001D800                                        C04A02CC E4E4CC00 0BF9F102 F1F2F2F1   *                ....UU...91.1221*
0001D820 F5F3F9F0 F8F1F8E3 C5E2E3F1 00010000    C00002CC CCC001F2 0C505C5C C5D9D9D6   *539C818TEST1...........2....ERRO*
00C1D840 C95C5C40 40D7C1D9 E34CF5F5 F5F5F540    C5D6E34C C6D6E4D5 C437FEC8            *R..  PART 55555 NOT FOUND..H    *


-STORAGE

00020100                                                        012A02C0 E4E4C000   *                        ....UU..*
00020120 CCF9F101 F1F2F2F1 F5F3F9F0 F8F2F1E3    C5E2E3F1 00010CC0 0000C100 000001F2   *.91.1221539C821TEST1...........2*
00C20140 CC504040 404040E2 E3D6C3D2 40E2E3C1    E3E4E240 C9C5D8E4 C5E2E315 D7C1D9E3   *..      STOCK STATUS REQUEST.PART*
00020160 4CD5E4D4 C2C5D940 F1F2F3F4 F515C4C5    E2C3C9C9 D7E3C9D6 D540F161 F240C9C5   * NUMBER 12345.DESCRIPTION 1.2 IN*
CCC20180 4CE2E3C5 C5D340E6 C1E2C8C5 D915D6D9    C4C5D94C E4D5C9E3 E240C7D9 E240404C   * STEEL WASHER.ORDER UNITS GRS   *
000201A0 C7D9C9C3 C5405BF5 F0F54BF0 F5F0F715    E2E3D6C3 C240E2E3 C1E3E4E2 40C1E34C   *PRICE .505.C5C7.STOCK STATUS AT *
000201C0 E6C1D9C5 C8D6E4E2 C540F2F0 F0154C40    40D3D6C3 C1E3C9D6 D540C4C9 C1D4C96B   *WAREHOUSE 2C0.   LOCATION MIAMI.*
000201E0 40C6D3C1 4B154040 404040D6 D540C8C1    D5C44040 F66BF1F6 F16BF5FC F640404C   * FLA..     CN HAND  6.161.5C6   *
0002020C 4C404040 C1E240D6 C64040F0 F361FCF5    61F8F215 4C4C4C40 40D6D540 D6D9C4C5   *    AS OF  C3.05.82.     ON ORDE*
0002022C C940F46B F0F4F06B F6F1F740 40404C40    4C4CC1E2 4CD6C640 40F1F061 F1F161F8   *R 4.C40.617       AS OF  10.11.8*
00C20240 F237BBB8                                                                      *2...                            *


-STORAGE

0001D8A0 CC5B0200 E4E40000 0DF9F102 F1F2F2F1    F5F3F9FC F8F3F1E3 C5E2E3F1 000100CC   *....UU...91.1221539C831TEST1....*
C001D8C0 C0000300 000001F2 CC5C5C5C C5D9D9D6    D95C5C4C 4CD7C1D9 E340F1F2 F3F4F54C   *........2....ERRCR..  PART 12345 *
0001D8E0 D5D6E340 C6D6E4D5 C440C9D5 40E6C1D9    C5C8D6E4 E2C54CF3 F0F037BB            *NOT FOUNC IN WAREHOUSE 300..    *


-STORAGE

0001D760                            0C4A0200 E4E40C00    CEF9F101 F1F2F2F1 F5F3F9F0 F8F3F4E3   *        ....LU...91.12215390834T*
CCC1D780 C5E2E3F1 00010000 0C000400 000001F2    C0505C5C C5D9D9D6 D95C5C40 40D7C1D9   *EST1...........2....ERROR..  PAR*
0001D7A0 E340F1F2 F3F4F940 D5D6E340 C6D6E4D5    C437D01C                              *T 12349 NOT FOUND...            *
```

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



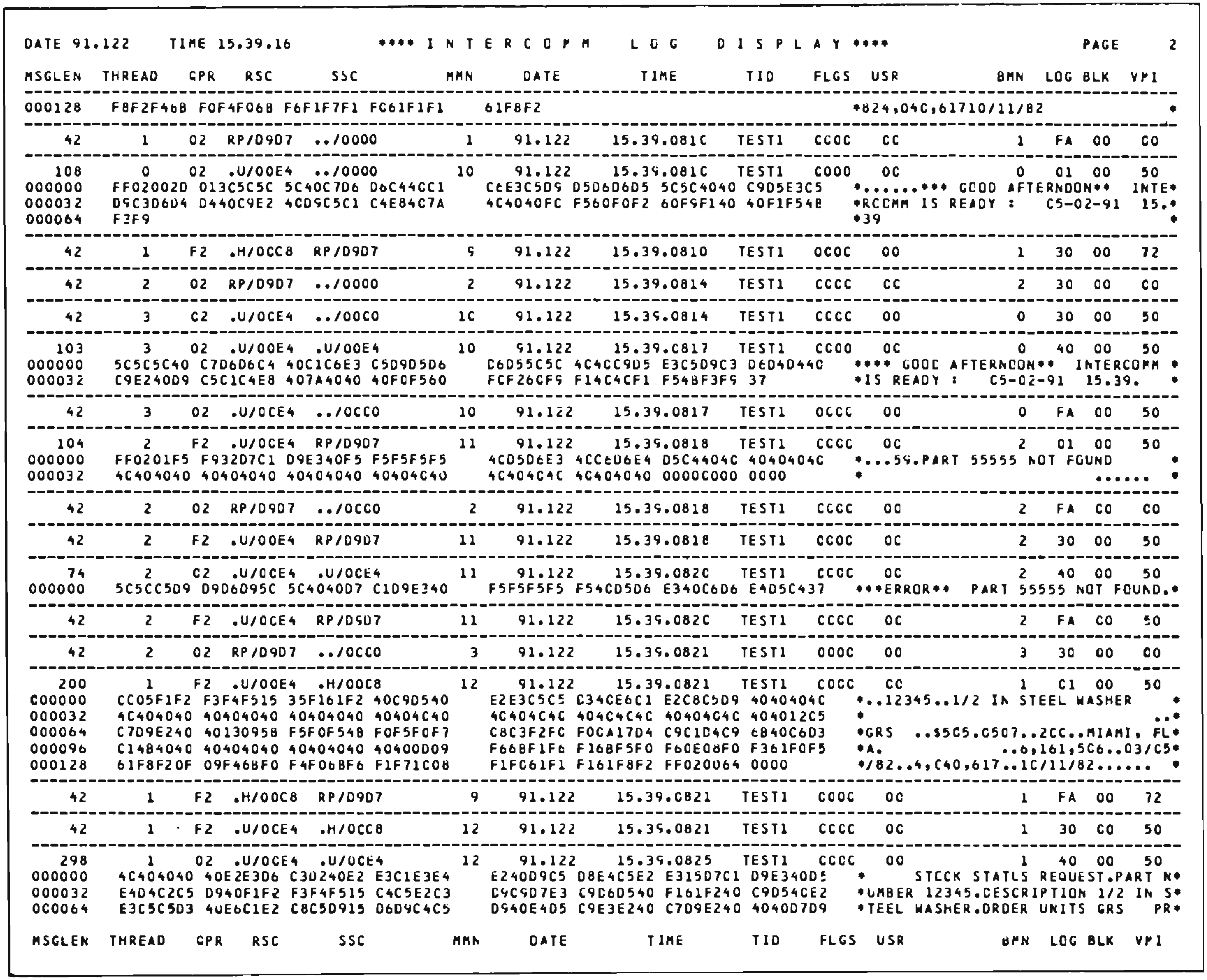
```
JDB INTT002Z          STEP GD               TIME 1539C8   DATE 91122    ID = 02C    CPUID = C69109515890   PAGE 00000001

PSW AT ENTRY TD SNAP   078D2E00 00007AC6          ILC 2   INTC G033

-STDRAGE

0001E4E0                                                         C0B902C0 D5E40000   *                    ....NU..*
0001E500 1CF9F102 F1F2F2F1 F5F3F9F0 F8F3F8E3    C5E2E3F1 00010000 00000600 000001F2   *.91.1221539C838TEST1...........2*
C001E520 0C50F0F0 C5F840F0 F0F0F2F9 15D5D6D5    6CD5E4D4 C5D9C9C3 40C3C8C1 D9C1C3E3   *..0CE8 00029.NCN.NUMERIC CHARACT*
00C1E540 C5D940C7 C9E5C5D5 40D6D54C D761D540    C7C1D9C1 D4C5E3C5 D940C6D6 D940C9E3   *ER GIVEN ON P.N PARAMETER FOR RT*
0001E560 C9D740E5 C5D9C24B 40C1D3D3 40C3C8C1    D9C1C3E3 C5D9E240 E2C8D6E4 D3C440C2   *RP VERB. ALL CHARACTERS SHOULD B*
0001E58C C540D5E4 D4C5D9C9 C34B1540 D4C5E2E2    C1C7C54C C5D64B40 F0F0F0F0 F0F0F0F6   *E NUMERIC.. MESSAGE NO. 00000C06*
00C1E5A0 4CC6D9D6 D440E3D7 E440E3C5 E2E3F14B    37F161F8                              * FROM TPU TEST1..1.8            *


-STORAGE

0C01E4E0                                                         00B60200 D5E4C0CC   *                    ....NU..*
0001E500 11F9F102 F1F2F2F1 F5F3F9F0 F8F3F8E3    C5E2E3F1 C0010CCC 000006C0 000001F2   *.91.1221539C838TEST1...........2*
0001E520 CC50F0F0 C5F840F0 F0F0F2F2 15D9C5D8    E4C9D9C5 C44CD7C1 D9C1D4C5 E3C5D94C   *..00E8 CCC22.RECUIRED PARAMETER *
C0C1E540 D761D540 E6C1E240 D6D4C9E3 E3C5C44D    D6D940C7 C9E5C5D5 40C9D540 C5D9D9D6   *P.N WAS OMITTED.OR GIVEN IN ERRO*
0001E560 C95DD6D5 40E3C8C5 4040D9E3 D9D74CE5    C5D9C24B 40E5C5D9 C240E6C1 E240C3C1   *R.ON THE  RTRP VEFB. VERB WAS CA*
00C1E580 D5C3C5D3 D3C5C415 40D4C5E2 E2C1C7C5    40C5D64B 40FCFCF0 F0F0F0FC F640C6D9   *NCELLED. MESSAGE NO. 00C000C6 FR*
0001E5A0 D6D440E3 D7E440E3 C5E2E3F1 4B37F14B                                          *OM TPU TEST1..1.               *


-STDRAGE

00020100                                                         C13E02C0 E4E4C000   *                    ....UU..*
00020120 12F9F101 F1F2F2F1 F5F3F9F0 F8F4F8E3    C5E2E3F1 00010000 00000500 0000C1F2   *.91.1221539C848TEST1...........2*
C0C20140 CC504040 404040E2 E3D6C3D2 40E2E3C1    E3E4E240 C9C5D8E4 C5E2E315 D7C1D9E3   *..       STOCK STATUS REQUEST.PART*
00020160 4CD5E4D4 C2C5D940 F1F2F3F4 F115C4C5    E2C3D9C9 D7E3C9D6 D540F161 F440C9C5   * NUMBER 12341.DESCRIPTION 1.4 IN*
00020180 4CC3C8D9 D6D4C540 C3C5D9D9 C1E3C5C4    40D3D6C3 D24CD5E4 E315D6D9 C4C5D94C   * CHROME CERRATEC LOCK NUT.ORDER *
000201A0 E4D5C9E3 E240C4D6 E9404040 D7D9C9C3    C54058F6 F1F64BF1 F6F1F615 E2E3D6C3   *UNITS DOZ    PRICE .616.1616.STOC*
0CC201C0 D240E2E3 C1E3E4E2 40C1E340 E6C1D9C5    C8D6E4E2 C54CF1F0 F0154040 40D3C6C3   *K STATLS AT WAREHCUSE 100.   LOC*
000201E0 C1E3C9D6 D540D5C5 E640E8D6 D9D24CC3    C9E3E86B 40D54BE8 4B154C4C 404040D6   *ATION NEW YCRK CITY. N.Y..     O*
0002020C D540C8C1 D5C44040 F56BF0F5 F06BF5F0    F4404C4C 40404040 C1E240D6 C64040FC   *N HAND  5.05C.5C4        AS CF  0*
C0C20220 F361F0F5 61F8F215 4C404040 40D6D540    D6D9C4C5 D940F56B FCF5F06B F5F0F440   *3.C5.82.      ON ORDER 5.05C.5C4 *
00C20240 4C404040 4040C1E2 40D6C640 40F1FC61    F1F161F8 F2376B8B                     *      AS OF  1C.11.82...          *

-STORAGE

0001DE20 CC690200 E4E40000 15F9F100 F1F2F2F1    F5F3F9F1 F2F5F3C3 D5E3F0F1 0000C00C   *....UU...91.12215391253CNT01....*
00C1DE40 CC000000 000000F2 0C50155C 5C5C4CC7    D6D6C44C C1C6E3C5 D9D5D6D6 D55C5C4C   *.......2...... GOCD AFTERNOON.. *
0001DE60 40C9D5E3 C5D9C3D6 D4D440C9 E240C3D3    D6E2C5C4 7A404040 F0F560F0 F260F9F1   * INTERCOMM IS CLOSED.   C5.C2.91*
C0C1DE80 4C40F1F5 4BF3F915 26000000                                                   *  15.39.....                      *


-STORAGE

C001DD80                                                         006702C0 E4E4C000   *                    ....UU..*
0001DDA0 16F9F100 F1F2F2F1 F5F3F9F1 F3F0F4E3    C5E2E3F1 00CCC0CC 00000000 000000F2   *.91.12215391304TEST1...........2*
0001DDC0 0C505C5C 5C40C7D6 D6C440C1 C6E3C5D9    D5D6D6D5 5C5C4C40 C9D5E3C5 D9C3D6D4   *..... GOOC AFTERNCON..  INTERCOM*
0001DDEC D440C9E2 40C3D3D6 E2C5C47A 404040F0    F560F0F2 6CF9F140 40F1F54B F3F93700   *M IS CLOSED.   C5.02.91  15.39..*
```

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



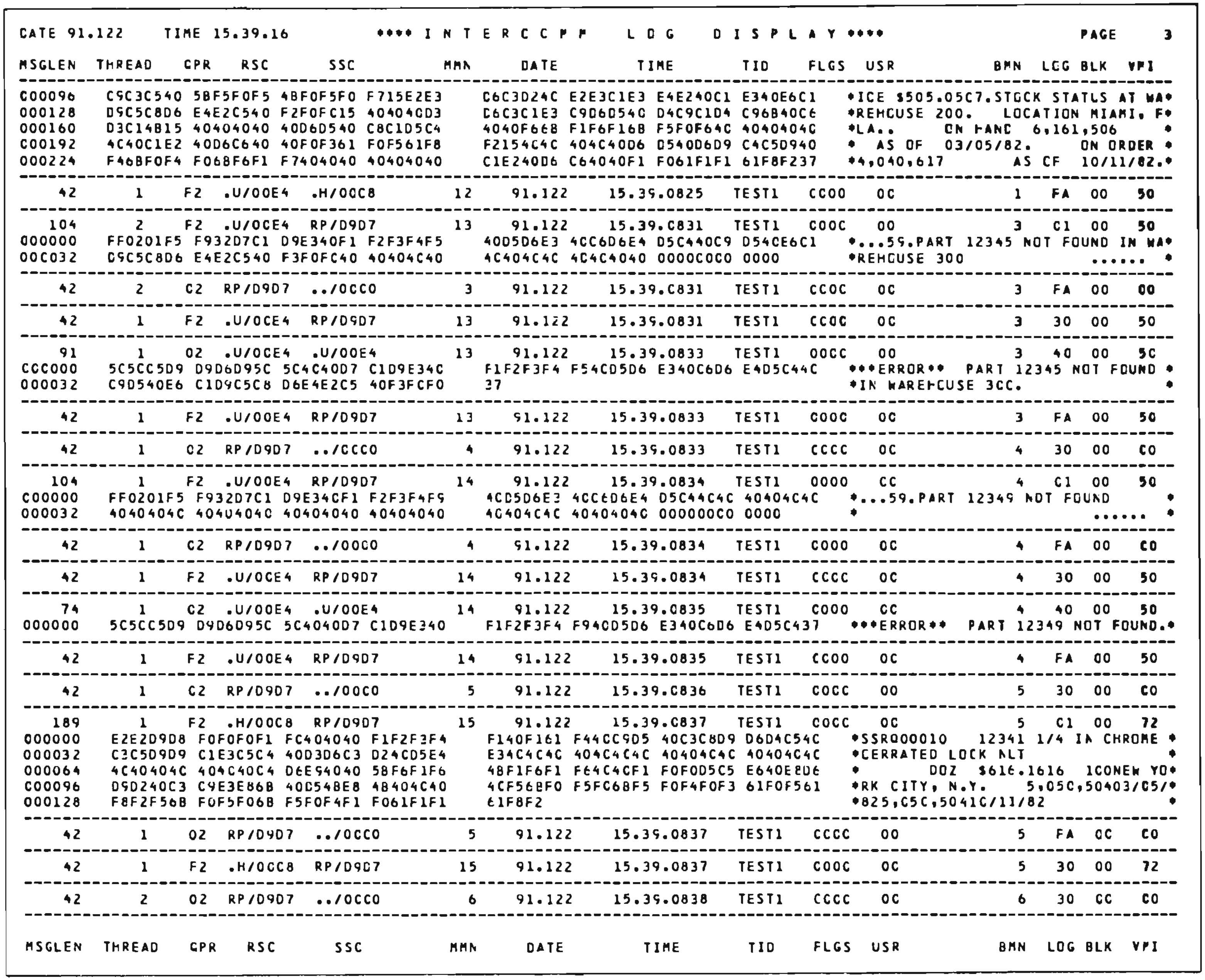
```
DATE 91.122    TIME 15.39.16         **** I N T E R C C M M   L O G   O I S P L A Y ****                   PAGE     1

MSGLEN  THREAD   GPR   RSC      SSC           MMN     DATE       TIME        TID     FLGS  USR              BPN  LOG BLK  VPI
------------------------------------------------------------------------------------------------------------------------------
    78       0    02  ../0000  ../0CC0          1    91.122    15.39.C696   TEST1    CCCC   00                 1   9F  00   CC
C00000   C9D5E3C5 D9C3D6D4 D440E2E3 C1D9E3E4    C74004C5 E2E2C1C7 C54C6040 C9D5E3E3   *INTERCOMM STARTUP MESSAGE - INTT*
00C032   FCF0F2E9                                                                      *002Z                              *
------------------------------------------------------------------------------------------------------------------------------
    65       0    02  RP/D9D7  ../U000          1    91.122    15.39.C696   TEST1    00CC   0C                 1   01  00   C0
C00000   D9E3D9D7 15D761D5 4CF1F2F3 F4F515E6    C8E240F2 FCFC37                       *RTRP.P/N 12345.WHS 200.           *
------------------------------------------------------------------------------------------------------------------------------
    65       0    02  RP/D9D7  ../0CC0          2    91.122    15.39.0701   TEST1    CCCC   00                 2   C1  00   C0
000000   C9E3D9D7 15D761D5 40F5F5F5 F5F515E6    C8E240F2 FCF037                       *RTRP.P/N 55555.WHS 200.           *
------------------------------------------------------------------------------------------------------------------------------
    65       0    02  RP/D9D7  ../CCC0          3    91.122    15.39.0702   TEST1    C000   0C                 3   01  00   00
000000   C9E3D9D7 15D761D5 40F1F2F3 F4F515E6    C8E240F3 FCF037                       *RTRP.P/N 12345.WHS 300.           *
------------------------------------------------------------------------------------------------------------------------------
    65       0    02  RP/D9D7  ../0CC0          4    91.122    15.39.C703   TEST1    0000   0C                 4   01  00   00
000000   C9E3D9D7 15D761D5 40F1F2F3 F4F915E6    C8E240F2 FCFC37                       *RTRP.P/N 12349.WHS 200.           *
------------------------------------------------------------------------------------------------------------------------------
    65       0    C2  RP/D9D7  ../00C0          5    91.122    15.39.0705   TEST1    00CC   00                 5   01  00   C0
C00000   D9E3D9D7 15D761D5 4CF1F2F3 F4F115E6    C8E240F1 FCF037                       *RTRP.P/N 12341.WHS 100.           *
------------------------------------------------------------------------------------------------------------------------------
    65       0    02  RP/D907  ../0CC0          6    91.122    15.39.C706   TEST1    CCCC   00                 6   01  CC   00
000000   D9E3D9D7 15D761D5 4CC1F2F3 F4F515E6    C8E240F4 FCF037                       *RTRP.P/N A2345.WHS 400.           *
------------------------------------------------------------------------------------------------------------------------------
   108       0    02  .U/00E4  ../0000          7    91.122    15.39.0748   TOALL    C000   CC                 0   C1  00   50
000000   FF02002D 013C5C5C 5C40C7D6 D6C44CC1    C6E3C5D9 D5D6D6D5 5C5C4040 C9D5E3C5   *.......*** GCOD AFTERNOON**  INTE*
000032   D9C3D6D4 D440C9E2 4CD9C5C1 C4E84C7A    4C404CFC F560F0F2 6CF9F140 40F1F54B   *RCOMM IS READY :    C5-02-91  15.*
CC0064   F3F9                                                                          *39                                *
------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .U/0CE4  ../00C0          7    91.122    15.39.0753   TOALL    CC0C   0C                 0   30  C0   50
------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .U/0CE4  ../00C0          7    91.122    15.39.C753   TOALL    0C0C   0C                 0   FA  00   5C
------------------------------------------------------------------------------------------------------------------------------
    42       1    02  RP/D9D7  ../CCC0          1    91.122    15.39.0753   TEST1    CCCC   0C                 1   20  C0   C0
------------------------------------------------------------------------------------------------------------------------------
   108       0    C2  .U/00E4  ../0000          8    91.122    15.39.C760   CNTC1    00CC   00                 0   C1  00   5C
C00000   FF02002D 013C5C5C 5C4CC7D6 D6C44CC1    C6E3C5D9 C5D6D6D5 5C5C4040 C9D5E3C5   *.......*** GCCD AFTERNOON**  INTE*
000032   D9C3D6D4 D440C9E2 4CD9C5C1 C4E84C7A    4C404CF0 F560F0F2 60F9F140 40F1F54B   *RCOMM IS READY :    05-02-91  15.*
000064   F3F9                                                                          *39                                *
------------------------------------------------------------------------------------------------------------------------------
    42       2    02  .U/00E4  ../0000          8    91.122    15.39.076C   CNT01    C00C   CC                 0   30  00   50
------------------------------------------------------------------------------------------------------------------------------
   105       2    C2  .U/0CE4  .U/00E4          8    91.122    15.39.C769   CNT01    CCCC   0C                 C   40  00   50
000000   155C5C5C 40C7D6D6 C44CC1C6 E3C5D9D5    C6D6D55C 5C4C40C9 05E3C5D9 C3D6D4D4   *.*** GDOC AFTERNUON**  INTERCOMM*
000032   4CC9E240 D9C5C1C4 E8407A40 404CFCF5    6CFCF26C F9F14C4C F1F548F3 F91526     * IS READY :   C5-02-91  15.39.. *
------------------------------------------------------------------------------------------------------------------------------
    42       2    C2  .U/0CE4  ../0000          8    91.122    15.39.C769   CNT01    CCCC   00                 0   FA  00   50
------------------------------------------------------------------------------------------------------------------------------
   189       1    F2  .H/00C8  RP/D9D7          9    91.122    15.39.C81C   TEST1    C00C   0C                 1   C1  00   72
000000   E2E2D9D8 F0F0F0F1 F0404040 F1F2F3F4    F540F161 F24CC9D5 40E2E3C5 C5D340E6   *SSRQ00010   12345 1/2 IN STEEL W*
000032   C1E2C8C5 D940404C 40404C4C 40404C40    4C4C4C4C 40404C4C 4040404C 4040404C   *ASHER                            *
0C0064   4C40404C 404040C7 D9E24040 5BF5FCF5    4BF0F5FC F74C40F2 FCFCD4C9 C1D4C968   *         GRS  $5C5.C507  2CCMIAMI,*
000096   40C6D3C1 4B404040 40404040 40404040    40F66BF1 F6F16BF5 F0F6F0F3 61F0F561   * FLA.              6,161,5C603/C5/*

MSGLEN  THREAD   QPR   RSC      SSC           MMN     DATE       TIME        TID     FLGS  USR              BPN  LOG BLK  VPI
```

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



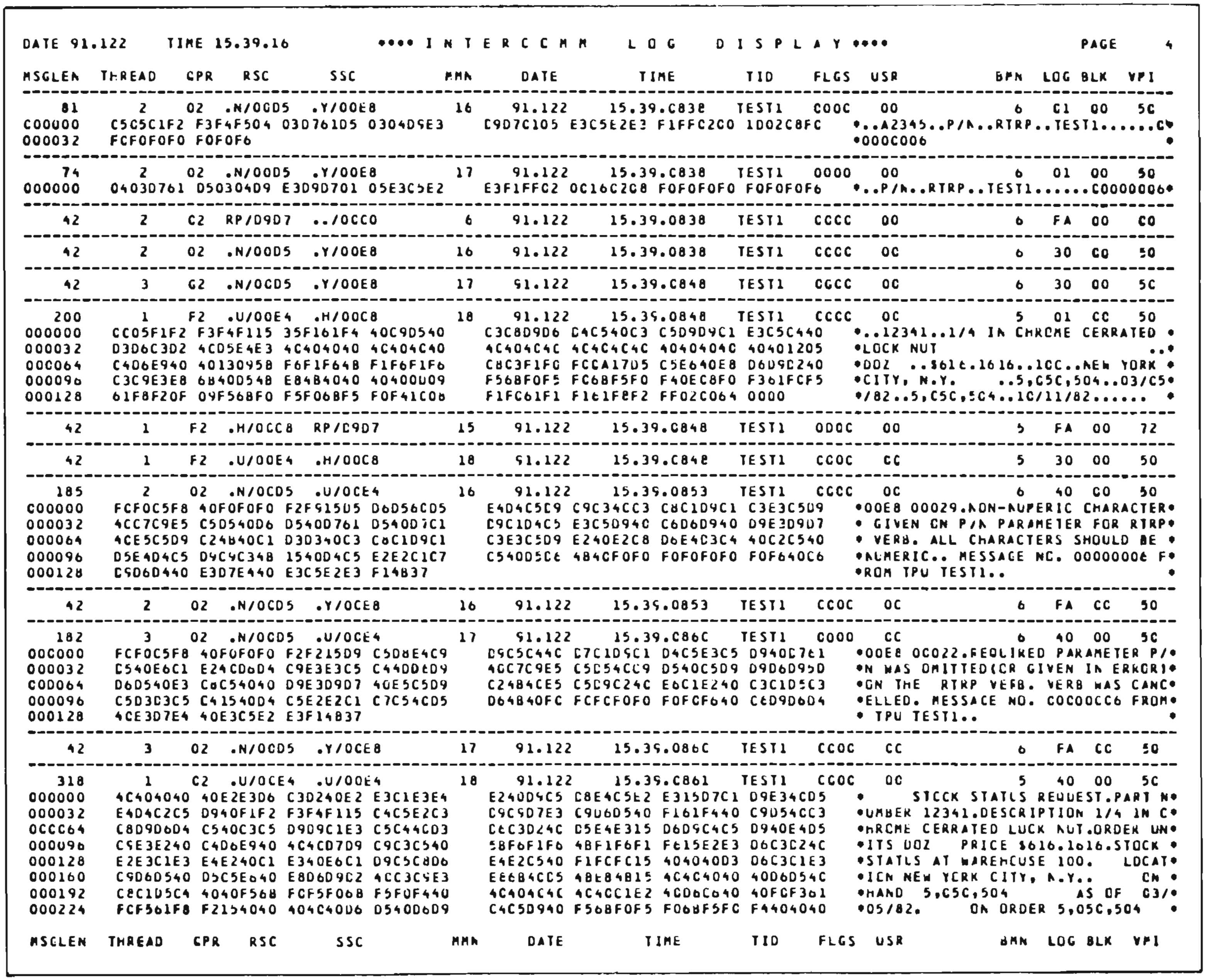
```
DATE 91.122    TIME 15.39.16        **** I N T E R C O M M   L O G   D I S P L A Y ****              PAGE     2

MSGLEN  THREAD   GPR   RSC        SSC           MMN      DATE         TIME        TID     FLGS  USR            BMN  LOG BLK  VPI
-------------------------------------------------------------------------------------------------------------------------------
000128   F8F2F46B F0F4F06B F6F1F7F1 FC61F1F1     61F8F2                                 *824,04C,61710/11/82                *
-------------------------------------------------------------------------------------------------------------------------------
    42       1     02  RP/D9D7  ../0000          1      91.122      15.39.081C   TEST1   CCOC   CC                1   FA  00   CO
-------------------------------------------------------------------------------------------------------------------------------
   108       0     02  .U/00E4  ../0000         10      91.122      15.39.081C   TEST1   C000   OC                0   01  00   50
000000   FF02002D 013C5C5C 5C40C7D6 D6C44CC1     C6E3C5D9 D5D6D6D5 5C5C4040 C9D5E3C5   *.......*** GOOD AFTERNOON**  INTE*
000032   D9C3D6D4 D440C9E2 4CD9C5C1 C4E84C7A     4C4040FC F560F0F2 60F9F140 40F1F54B   *RCOMM IS READY :   C5-02-91  15.*
000064   F3F9                                                                          *39                                 *
-------------------------------------------------------------------------------------------------------------------------------
    42       1     F2  .H/0CC8  RP/D9D7          9      91.122      15.39.0810   TEST1   OCOC   00                1   30  00   72
-------------------------------------------------------------------------------------------------------------------------------
    42       2     02  RP/D9D7  ../0000          2      91.122      15.39.0814   TEST1   CCCC   CC                2   30  00   CO
-------------------------------------------------------------------------------------------------------------------------------
    42       3     C2  .U/0CE4  ../00C0         1C      91.122      15.39.0814   TEST1   CCCC   00                0   30  00   50
-------------------------------------------------------------------------------------------------------------------------------
   103       3     02  .U/00E4  .U/00E4         10      91.122      15.39.C817   TEST1   CC00   OC                0   40  00   50
000000   5C5C5C40 C7D6D6C4 40C1C6E3 C5D9D5D6     D6D55C5C 4C4CC9D5 E3C5D9C3 D6D4D440   **** GOOD AFTERNOON**  INTERCOMM *
000032   C9E240D9 C5C1C4E8 407A4040 40F0F560     FCF26CF9 F14C4CF1 F54BF3F9 37         *IS READY :   C5-02-91  15.39.   *
-------------------------------------------------------------------------------------------------------------------------------
    42       3     02  .U/0CE4  ../0CC0         10      91.122      15.39.0817   TEST1   OCCC   00                0   FA  00   50
-------------------------------------------------------------------------------------------------------------------------------
   104       2     F2  .U/0CE4  RP/D9D7         11      91.122      15.39.0818   TEST1   CCCC   OC                2   01  00   50
000000   FF0201F5 F932D7C1 D9E340F5 F5F5F5F5     4CD5D6E3 4CC6D6E4 D5C4404C 4040404C   *...59.PART 55555 NOT FOUND         *
000032   4C404040 40404040 40404040 40404C40     4C404C4C 4C404040 0000C000 0000       *                            ......  *
-------------------------------------------------------------------------------------------------------------------------------
    42       2     02  RP/D9D7  ../0CC0          2      91.122      15.39.0818   TEST1   CCCC   00                2   FA  CO   CO
-------------------------------------------------------------------------------------------------------------------------------
    42       2     F2  .U/00E4  RP/D9D7         11      91.122      15.39.0818   TEST1   CCOC   OC                2   30  00   50
-------------------------------------------------------------------------------------------------------------------------------
    74       2     C2  .U/0CE4  .U/0CE4         11      91.122      15.39.082C   TEST1   CCCC   OC                2   40  00   50
000000   5C5CC5D9 D9D6D95C 5C4040D7 C1D9E340     F5F5F5F5 F54CD5D6 E340C6D6 E4D5C437   ***ERROR**  PART 55555 NOT FOUND.*
-------------------------------------------------------------------------------------------------------------------------------
    42       2     F2  .U/0CE4  RP/D9D7         11      91.122      15.39.082C   TEST1   CCCC   OC                2   FA  CO   50
-------------------------------------------------------------------------------------------------------------------------------
    42       2     02  RP/D9D7  ../0CC0          3      91.122      15.39.0821   TEST1   000C   00                3   30  00   CO
-------------------------------------------------------------------------------------------------------------------------------
   200       1     F2  .U/00E4  .H/00C8         12      91.122      15.39.0821   TEST1   COCC   CC                1   C1  00   50
C00000   CC05F1F2 F3F4F515 35F161F2 40C9D540     E2E3C5C5 C34CE6C1 E2C8C5D9 4040404C   *..12345..1/2 IN STEEL WASHER      *
000032   4C404040 40404040 40404040 40404C40     4C404C4C 404C4C4C 4040404C 404012C5   *                                ..*
000064   C7D9E240 4013095B F5F0F54B F0F5F0F7     C8C3F2FC F0CA17D4 C9C1D4C9 6840C6D3   *GRS  ..$5C5.C507..2CC..MIAMI, FL*
000096   C1484040 40404040 40404040 40400D09     F66BF1F6 F16BF5F0 F60E08F0 F361F0F5   *A.                ..6,161,5C6..03/C5*
000128   61F8F20F 09F46BF0 F4F06BF6 F1F71C08     F1FC61F1 F161F8F2 FF020064 0000       */82..4,C40,617..1C/11/82......  *
-------------------------------------------------------------------------------------------------------------------------------
    42       1     F2  .H/00C8  RP/D9D7          9      91.122      15.39.C821   TEST1   COOC   OC                1   FA  00   72
-------------------------------------------------------------------------------------------------------------------------------
    42       1   · F2  .U/0CE4  .H/0CC8         12      91.122      15.39.0821   TEST1   CCCC   OC                1   30  CO   50
-------------------------------------------------------------------------------------------------------------------------------
   298       1     02  .U/0CE4  .U/0CE4         12      91.122      15.39.0825   TEST1   CCCC   00                1   40  00   50
000000   4C404040 40E2E3D6 C3D240E2 E3C1E3E4     E240D9C5 D8E4C5E2 E315D7C1 D9E340D5   *     STOCK STATUS REQUEST.PART N*
000032   E4D4C2C5 D940F1F2 F3F4F515 C4C5E2C3     C9C9D7E3 C9D6D540 F161F240 C9D54CE2   *UMBER 12345.DESCRIPTION 1/2 IN S*
0C0064   E3C5C5D3 40E6C1E2 C8C5D915 D6D9C4C5     D940E4D5 C9E3E240 C7D9E240 4040D7D9   *TEEL WASHER.ORDER UNITS GRS   PR*

MSGLEN  THREAD   GPR   RSC        SSC           MMN      DATE         TIME        TID     FLGS  USR            BMN  LOG BLK  VPI
```

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



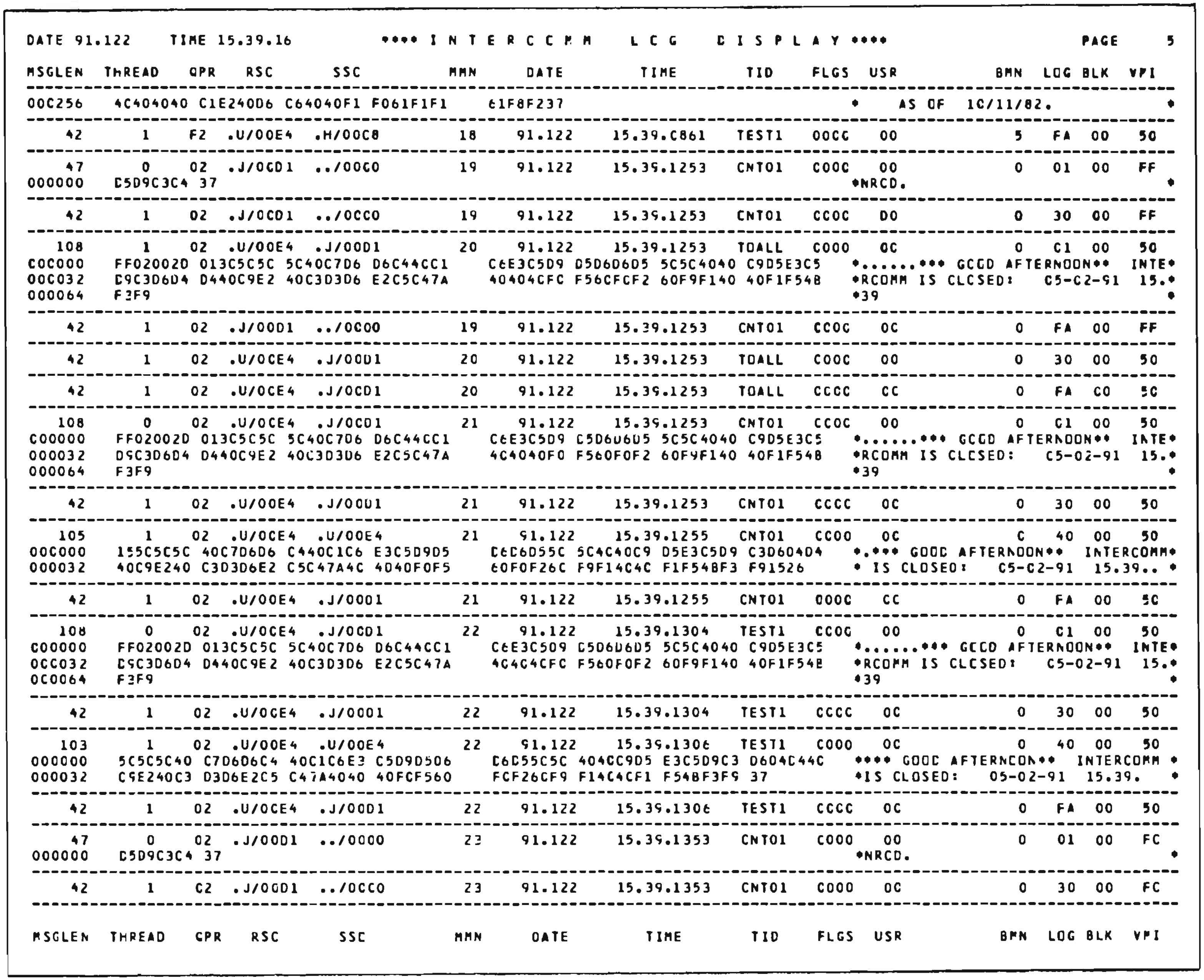
```
CATE 91.122     TIME 15.39.16           **** I N T E R C C P P     L O G     D I S P L A Y ****                     PAGE     3

MSGLEN  THREAD   CPR   RSC        SSC             MMN       DATE            TIME          TID     FLGS  USR             BMN  LCG BLK  VPI
--------------------------------------------------------------------------------------------------------------------------------------------
C00096    C9C3C540 5BF5F0F5 4BF0F5F0 F715E2E3     C6C3D24C E2E3C1E3 E4E240C1 E340E6C1    *ICE $505.05C7.STGCK STATLS AT WA*
000128    D9C5C8D6 E4E2C540 F2F0FC15 404040D3     C6C3C1E3 C9D6D54C D4C9C1D4 C96B40C6    *REHCUSE 200.   LOCATION MIAMI, F*
000160    D3C14B15 40404040 40D6D540 C8C1D5C4     4040F66B F1F6F16B F5F0F64C 4040404C    *LA..       CN HANC  6,161,506    *
C00192    4C40C1E2 40D6C640 40F0F361 F0F561F8     F2154C4C 404C40D6 D540D6D9 C4C5D940    *  AS OF  03/05/82.      ON ORDER *
000224    F46BF0F4 F068F6F1 F7404040 40404040     C1E240D6 C64040F1 F061F1F1 61F8F237    *4,040,617          AS CF  10/11/82.*
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  .U/00E4   .H/00C8          12    91.122     15.39.0825    TEST1    CC00    0C                 1   FA   00   50
--------------------------------------------------------------------------------------------------------------------------------------------
   104       2    F2  .U/0CE4   RP/D9D7          13    91.122     15.39.C831    TEST1    C00C    00                 3   C1   00   50
000000    FF0201F5 F932D7C1 D9E340F1 F2F3F4F5     40D5D6E3 4CC6D6E4 D5C440C9 D540E6C1    *...59.PART 12345 NOT FOUND IN WA*
00C032    C9C5C8D6 E4E2C540 F3F0FC40 40404C40     4C404C4C 4C4C4040 0000C0C0 0000        *REHCUSE 300                ...... *
--------------------------------------------------------------------------------------------------------------------------------------------
    42       2    C2  RP/D9D7   ../0CC0           3    91.122     15.39.C831    TEST1    CC0C    0C                 3   FA   00   00
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  .U/0CE4   RP/D9D7          13    91.122     15.39.0831    TEST1    CC0C    0C                 3   30   00   50
--------------------------------------------------------------------------------------------------------------------------------------------
    91       1    02  .U/0CE4   .U/00E4          13    91.122     15.39.0833    TEST1    00CC    00                 3   40   00   5C
CCC000    5C5CC5D9 D9D6D95C 5C4C40D7 C1D9E34C     F1F2F3F4 F54CD5D6 E340C6D6 E4D5C44C    ***ERROR**  PART 12345 NOT FOUND *
000032    C9D540E6 C1D9C5C8 D6E4E2C5 40F3FCF0     37                                     *IN WAREHCUSE 3CC.                *
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  .U/00E4   RP/D9D7          13    91.122     15.39.0833    TEST1    C00C    0C                 3   FA   00   50
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  RP/D9D7   ../CCC0           4    91.122     15.39.0833    TEST1    CCCC    0C                 4   30   00   C0
--------------------------------------------------------------------------------------------------------------------------------------------
   104       1    F2  .U/00E4   RP/D9D7          14    91.122     15.39.0834    TEST1    0000    CC                 4   C1   00   50
C00000    FF0201F5 F932D7C1 D9E34CF1 F2F3F4F9     4CD5D6E3 4CC6D6E4 D5C44C4C 40404C4C    *...59.PART 12349 NOT FOUND       *
000032    4040404C 40404040 40404040 40404040     4C404C4C 4040404C 000000C0 0000        *                           ...... *
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    C2  RP/D9D7   ../00C0           4    91.122     15.39.0834    TEST1    C000    0C                 4   FA   00   C0
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  .U/0CE4   RP/D9D7          14    91.122     15.39.0834    TEST1    CCCC    0C                 4   30   00   50
--------------------------------------------------------------------------------------------------------------------------------------------
    74       1    C2  .U/00E4   .U/00E4          14    91.122     15.39.0835    TEST1    C000    CC                 4   40   00   50
000000    5C5CC5D9 D9D6D95C 5C4040D7 C1D9E340     F1F2F3F4 F940D5D6 E340C6D6 E4D5C437    ***ERROR**  PART 12349 NOT FOUND.*
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  .U/00E4   RP/D9D7          14    91.122     15.39.0835    TEST1    CC00    0C                 4   FA   00   50
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    C2  RP/D9D7   ../00C0           5    91.122     15.39.C836    TEST1    C0CC    00                 5   30   00   C0
--------------------------------------------------------------------------------------------------------------------------------------------
   189       1    F2  .H/00C8   RP/D9D7          15    91.122     15.39.C837    TEST1    C0CC    0C                 5   C1   00   72
000000    E2E2D9D8 F0F0F0F1 FC404040 F1F2F3F4     F140F161 F44CC9D5 40C3C8D9 D6D4C54C    *SSRQ00010   12341 1/4 IN CHROME *
000032    C3C5D9D9 C1E3C5C4 40D3D6C3 D24CD5E4     E34C4C4C 404C4C4C 40404C4C 40404C4C    *CERRATED LOCK NLT                *
000064    4C40404C 404C40C4 D6E94040 5BF6F1F6     4BF1F6F1 F64C4CF1 F0F0D5C5 E640E8D6    *        DOZ  $616.1616  1CONEW YO*
C00096    D9D240C3 C9E3E86B 40D54BE8 4B404C40     4CF56BF0 F5FC6BF5 F0F4F0F3 61F0F561    *RK CITY, N.Y.      5,05C,50403/C5/*
000128    F8F2F56B F0F5F06B F5F0F4F1 F061F1F1     61F8F2                                 *825,C5C,5041C/11/82              *
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  RP/D9D7   ../0CC0           5    91.122     15.39.0837    TEST1    CCCC    00                 5   FA   0C   C0
--------------------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  .H/0CC8   RP/D9D7          15    91.122     15.39.0837    TEST1    C00C    0C                 5   30   00   72
--------------------------------------------------------------------------------------------------------------------------------------------
    42       2    02  RP/D9D7   ../0CC0           6    91.122     15.39.0838    TEST1    CCCC    0C                 6   30   CC   C0
--------------------------------------------------------------------------------------------------------------------------------------------

MSGLEN  THREAD   CPR   RSC        SSC             MMN       DATE            TIME          TID     FLGS  USR             BMN  LOG BLK  VPI
```

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



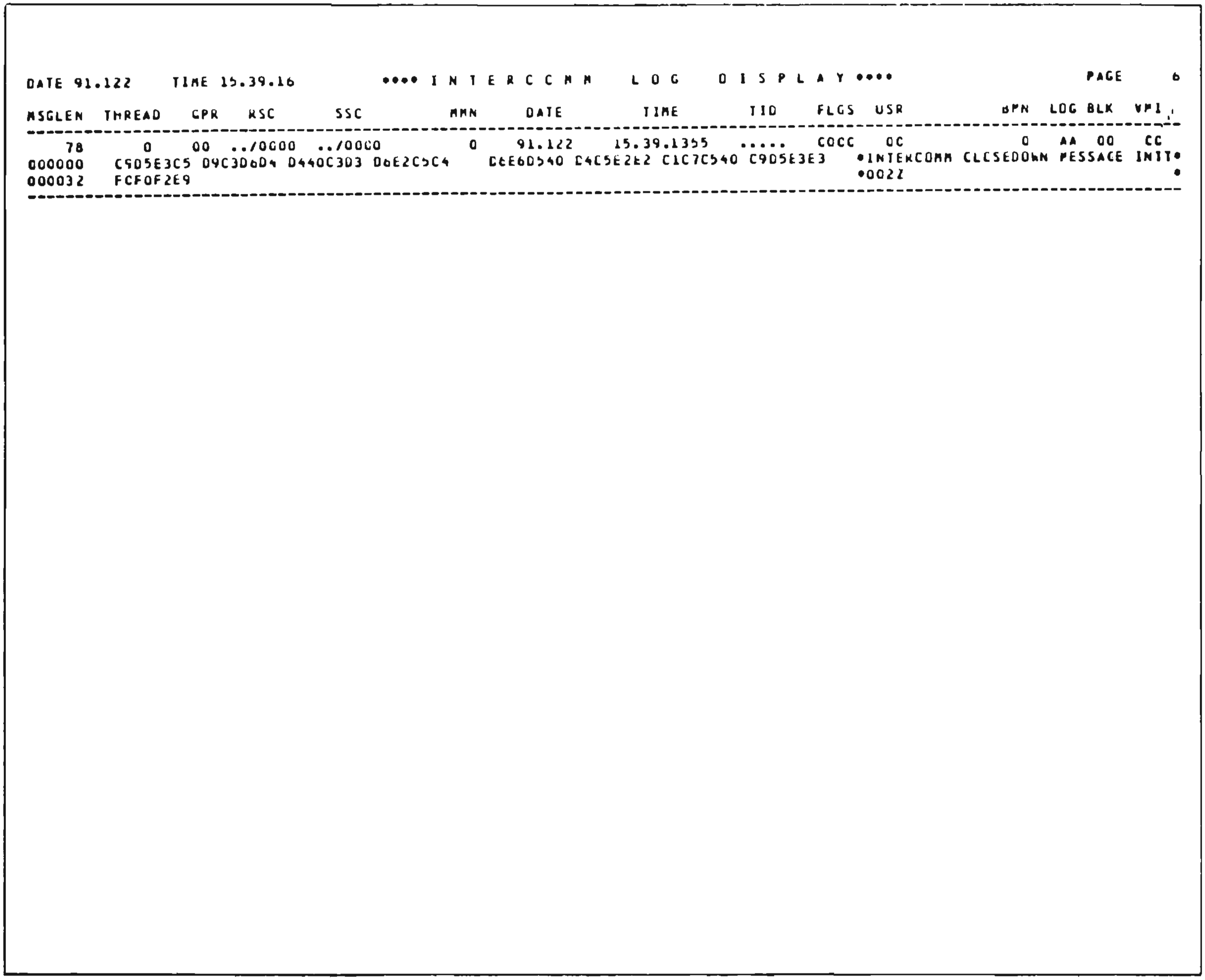
```
DATE 91.122     TIME 15.39.16          •••• I N T E R C C M M    L O G    D I S P L A Y ••••                PAGE     4

MSGLEN  THREAD   CPR   RSC       SSC           MMN     DATE          TIME        TID     FLGS  USR              BPN  LOG BLK  VPI
---------------------------------------------------------------------------------------------------------------------------------
    81       2    02  .N/0GD5   .Y/00E8          16     91.122      15.39.C83E   TEST1   C00C   00                6   C1  00   5C
C00000   C5C5C1F2 F3F4F504 03D761D5 0304D9E3   C9D7C105 E3C5E2E3 F1FFC2C0 1D02C8FC   •..A2345..P/N..RTRP..TEST1......C•
000032   FCF0F0F0 F0F0F6                                                             •000C006                          •
---------------------------------------------------------------------------------------------------------------------------------
    74       2    02  .N/00D5   .Y/00E8          17     91.122      15.39.C838   TEST1   0000   00                6   01  00   50
000000   0403D761 D50304D9 E3D9D701 05E3C5E2   E3F1FFC2 0C16C2C8 F0F0F0F0 F0F0F0F6   •..P/N..RTRP..TEST1......C0000006•
---------------------------------------------------------------------------------------------------------------------------------
    42       2    C2  RP/D9D7   ../0CC0           6     91.122      15.39.0838   TEST1   CCCC   00                6   FA  00   C0
---------------------------------------------------------------------------------------------------------------------------------
    42       2    02  .N/00D5   .Y/00E8          16     91.122      15.39.0838   TEST1   CCCC   0C                6   30  C0   50
---------------------------------------------------------------------------------------------------------------------------------
    42       3    G2  .N/0CD5   .Y/00E8          17     91.122      15.39.C848   TEST1   CGCC   0C                6   30  00   5C
---------------------------------------------------------------------------------------------------------------------------------
   200       1    F2  .U/00E4   .H/00C8          18     91.122      15.39.0848   TEST1   CCCC   0C                5   01  CC   50
000000   CC05F1F2 F3F4F115 35F161F4 40C9D540   C3C8D9D6 C4C540C3 C5D9D9C1 E3C5C440   •..12341..1/4 IN CHRCME CERRATED •
000032   D3D6C3D2 4CD5E4E3 4C404040 4C404C40   4C404C4C 4C4C4C4C 4040404C 40401205   •LOCK NUT                        ..•
000064   C4D6E940 4013095B F6F1F64B F1F6F1F6   C8C3F1F0 FCCA17D5 C5E640E8 D6D9C240   •DOZ  ..$616.1616..1CC..NEW YORK •
000096   C3C9E3E8 6B40D54B E84B4040 404000D9   F568F0F5 FC68F5F0 F40EC8F0 F361FCF5   •CITY, N.Y.     ..5,C5C,504..03/C5•
000128   61F8F20F 09F568F0 F5F068F5 F0F41C06   F1FC61F1 F161F8F2 FF02C064 0000        •/82..5,C5C,5C4..1C/11/82......  •
---------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  .H/0CC8   RP/D9D7          15     91.122      15.39.C848   TEST1   0D0C   00                5   FA  00   72
---------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  .U/00E4   .H/00C8          18     91.122      15.39.C84E   TEST1   CCOC   CC                5   30  00   50
---------------------------------------------------------------------------------------------------------------------------------
   185       2    02  .N/0CD5   .U/0CE4          16     91.122      15.39.0853   TEST1   CGCC   0C                6   40  C0   50
C00000   FCF0C5F8 40F0F0F0 F2F915D5 D6D56CD5   E4D4C5C9 C9C34CC3 C8C1D9C1 C3E3C5D9   •00E8 00029.NON-NUMERIC CHARACTER•
000032   4CC7C9E5 C5D540D6 D540D761 D540D7C1   C9C1D4C5 E3C5D94C C6D6D940 D9E3D9D7   • GIVEN CN P/N PARAMETER FOR RTRP•
000064   4CE5C5D9 C24B40C1 D3D340C3 C8C1D9C1   C3E3C5D9 E240E2C8 D6E4C3C4 40C2C540   • VERB. ALL CHARACTERS SHOULD BE •
000096   D5E4D4C5 D9C9C34B 1540D4C5 E2E2C1C7   C540D5C6 484CF0F0 F0F0F0F0 F0F640C6   •NUMERIC.. MESSAGE NC. 0000000E F•
000128   C9D6D440 E3D7E440 E3C5E2E3 F14B37                                            •ROM TPU TEST1..                  •
---------------------------------------------------------------------------------------------------------------------------------
    42       2    02  .N/0CD5   .Y/0CE8          16     91.122      15.39.0853   TEST1   CCOC   0C                6   FA  CC   50
---------------------------------------------------------------------------------------------------------------------------------
   182       3    02  .N/0CD5   .U/0CE4          17     91.122      15.39.C86C   TEST1   C000   CC                6   40  00   5C
00C000   FCF0C5F8 40F0F0F0 F2F215D9 C5D8E4C9   D9C5C44C C7C1D9C1 D4C5E3C5 D940C761   •00E8 0C022.REQUIRED PARAMETER P/•
000032   C540E6C1 E24CD6D4 C9E3E3C5 C44DD6D9   4CC7C9E5 C5C54CC9 D540C5D9 D9D6D95D   •N WAS OMITTED(CR GIVEN IN ERROR)•
C0D064   D6D540E3 C8C54040 D9E3D9D7 40E5C5D9   C24B4CE5 C5C9C24C E6C1E240 C3C1D5C3   •GN THE  RTRP VEFB. VERB WAS CANC•
000096   C5D3D3C5 C41540D4 C5E2E2C1 C7C54CD5   D64B40FC FCFCF0F0 F0FCF640 C6D9D6D4   •ELLED. MESSACE NO. COC00CC6 FROM•
000128   4CE3D7E4 40E3C5E2 E3F14B37                                                    • TPU TEST1..                     •
---------------------------------------------------------------------------------------------------------------------------------
    42       3    02  .N/0CD5   .Y/0CE8          17     91.122      15.39.086C   TEST1   CCOC   CC                6   FA  CC   50
---------------------------------------------------------------------------------------------------------------------------------
   318       1    C2  .U/0CE4   .U/00E4          18     91.122      15.39.C861   TEST1   CGOC   0C                5   40  00   5C
000000   4C404040 40E2E3D6 C3D240E2 E3C1E3E4   E240D9C5 C8E4C5E2 E315D7C1 D9E34CD5   •      STCCK STATUS REQUEST.PART N•
000032   E4D4C2C5 D940F1F2 F3F4F115 C4C5E2C3   C9C9D7E3 C9D6D540 F161F440 C9D54CC3   •UMBER 12341.DESCRIPTION 1/4 IN C•
0CCC64   C8D9D6D4 C540C3C5 D9D9C1E3 C5C44CD3   C6C3D24C C5E4E315 D6D9C4C5 D940E4D5   •HRCME CERRATED LUCK NUT.ORDER UN•
000096   C9E3E240 C4D6E940 4C4CD7D9 C9C3C540   5BF6F1F6 4BF1F6F1 F615E2E3 D6C3C24C   •ITS DOZ    PRICE $616.1616.STOCK •
000128   E2E3C1E3 E4E240C1 E340E6C1 D9C5C8D6   E4E2C540 F1FCFC15 404040D3 D6C3C1E3   •STATUS AT WAREHCUSE 100.   LOCAT•
000160   C9D6D540 D5C5E640 E8D6D9C2 4CC3C9E3   E6684CC5 4BE84B15 4C4C4040 40D6D54C   •ICN NEW YCRK CITY, N.Y..     CN •
000192   C8C1D5C4 4040F568 FCF5F068 F5F0F440   4C404C4C 4C4CC1E2 4CD6C640 40FCF361   •HAND  5,C5C,504        AS OF  C3/•
000224   FCF561F8 F2154040 404C40D6 D540D6D9   C4C5D940 F568F0F5 F068F5FC F4404040   •05/82.     ON ORDER 5,05C,504    •

MSGLEN  THREAD   CPR   RSC       SSC           MMN     DATE          TIME        TID     FLGS  USR              BMN  LOG BLK  VPI
```

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



```
DATE 91.122    TIME 15.39.16          **** I N T E R C C M M   L C G   C I S P L A Y ****                         PAGE     5

MSGLEN  ThREAD   QPR   RSC        SSC            MMN      DATE         TIME         TID     FLGS  USR               BMN  LOG BLK  VPI
-------------------------------------------------------------------------------------------------------------------------------------
00C256   4C404040 C1E240D6 C64040F1 F061F1F1    61F8F237                                   *    AS OF  1C/11/82.                *
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    F2  .U/00E4   .H/00C8          18     91.122    15.39.C861   TEST1   00CG    00                5   FA  00   50
-------------------------------------------------------------------------------------------------------------------------------------
    47       0    02  .J/0CD1   ../00G0          19     91.122    15.39.1253   CNT01   C00C    00                0   01  00   FF
000000   C5D9C3C4 37                                                                      *NRCD.                               *
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .J/0CD1   ../0CC0          19     91.122    15.39.1253   CNT01   CCOC    D0                0   30  00   FF
-------------------------------------------------------------------------------------------------------------------------------------
   108       1    02  .U/00E4   .J/00D1          20     91.122    15.39.1253   TOALL   C000    0C                0   C1  00   50
C0C000   FF02002D 013C5C5C 5C40C7D6 D6C44CC1    C6E3C5D9 D5D6D6D5 5C5C4040 C9D5E3C5   *.......*** GCGD AFTERNOON**   INTE*
00C032   C9C3D6D4 D440C9E2 40C3D3D6 E2C5C47A    40404CFC F56CFCF2 60F9F140 40F1F54B   *RCOMM IS CLCSED:    C5-C2-91   15.*
000064   F3F9                                                                         *39                                  *
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .J/00D1   ../0C00          19     91.122    15.39.1253   CNT01   CCOG    0C                0   FA  00   FF
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .U/0CE4   .J/00D1          20     91.122    15.39.1253   TOALL   COOC    00                0   30  00   50
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .U/0CE4   .J/0CD1          20     91.122    15.39.1253   TOALL   CCCC    CC                0   FA  C0   5C
-------------------------------------------------------------------------------------------------------------------------------------
   108       0    02  .U/0CE4   .J/0CD1          21     91.122    15.39.1253   CNT01   CCOC    00                0   C1  00   50
C00000   FF02002D 013C5C5C 5C40C7D6 D6C44CC1    C6E3C5D9 C5D6U6D5 5C5C4040 C9D5E3C5   *.......*** GCGD AFTERNOON**   INTE*
000032   D9C3D6D4 D440C9E2 40C3D3D6 E2C5C47A    4C4040F0 F560F0F2 60F9F140 40F1F548   *RCOMM IS CLCSED:    C5-02-91   15.*
000064   F3F9                                                                         *39                                  *
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .U/00E4   .J/00U1          21     91.122    15.39.1253   CNT01   CCCC    0C                0   30  00   50
-------------------------------------------------------------------------------------------------------------------------------------
   105       1    02  .U/0CE4   .U/00E4          21     91.122    15.39.1255   CNT01   CC00    0C                C   40  00   50
00C000   155C5C5C 40C7D6D6 C440C1C6 E3C5D9D5    C6C6D55C 5C4C40C9 D5E3C5D9 C3D604D4   *.*** GOOC AFTERNOON**  INTERCOMM*
000032   40C9E240 C3D3D6E2 C5C47A4C 4040F0F5    60F0F26C F9F14C4C F1F54BF3 F91526     * IS CLOSEO:   C5-C2-91   15.39.. *
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .U/00E4   .J/0001          21     91.122    15.39.1255   CNT01   000C    CC                0   FA  00   5C
-------------------------------------------------------------------------------------------------------------------------------------
   108       0    02  .U/0CE4   .J/0CD1          22     91.122    15.39.1304   TEST1   CCOG    00                0   C1  00   50
C00000   FF02002D 013C5C5C 5C40C7D6 D6C44CC1    C6E3C509 C5D6D6D5 5C5C4040 C9D5E3C5   *.......*** GCCD AFTERNOON**   INTE*
0CC032   C9C3D6D4 D440C9E2 40C3D3D6 E2C5C47A    4C4C4CFC F560F0F2 60F9F140 40F1F54E   *RCOMM IS CLCSED:    C5-02-91   15.*
0C0064   F3F9                                                                         *39                                  *
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .U/0CE4   .J/0001          22     91.122    15.39.1304   TEST1   CCCC    0C                0   30  00   50
-------------------------------------------------------------------------------------------------------------------------------------
   103       1    02  .U/00E4   .U/00E4          22     91.122    15.39.1306   TEST1   C000    0C                0   40  00   50
000000   5C5C5C40 C7D6D6C4 40C1C6E3 C5D9D506    C6C55C5C 404CC9D5 E3C5D9C3 D604C44C   **** GOOC AFTERNCON**  INTERCOMM *
000032   C9E240C3 D3D6E2C5 C47A4040 40FCF560    FCF26CF9 F14C4CF1 F54BF3F9 37         *IS CLOSED:   05-02-91  15.39.    *
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    02  .U/0CE4   .J/00D1          22     91.122    15.39.1306   TEST1   CCCC    0C                0   FA  00   50
-------------------------------------------------------------------------------------------------------------------------------------
    47       0    02  .J/00D1   ../0000          23     91.122    15.39.1353   CNT01   C000    00                0   01  00   FC
000000   C5D9C3C4 37                                                                      *NRCD.                               *
-------------------------------------------------------------------------------------------------------------------------------------
    42       1    C2  .J/0GD1   ../0CC0          23     91.122    15.39.1353   CNT01   C000    0C                0   30  00   FC
-------------------------------------------------------------------------------------------------------------------------------------

MSGLEN  THREAD   CPR   RSC        SSC            MMN      DATE         TIME         TID     FLGS  USR               BMN  LOG BLK  VPI
```

Figure 63. Test Mode Execution Log Printout (Page 5 of 6)



```
DATE 91.122    TIME 15.39.16         •••• I N T E R C C M M   L O G   D I S P L A Y ••••                PAGE     6

MSGLEN  THREAD   GPR   RSC        SSC          MMN      DATE          TIME         TID     FLGS  USR              BPN  LOG BLK  VMI
------------------------------------------------------------------------------------------------------------------------------------
    78       0    00  ../0G00  ../00C0           0     91.122      15.39.1355    .....    COCC    0C                 0    AA   00    CC
000000   C9D5E3C5 D9C3D6D4 D440C3D3 D6E2C5C4    C6E6D540 C4C5E2E2 C1C7C540 C9D5E3E3   •INTERCOMM CLCSEDOWN MESSAGE INIT•
000032   FCF0F2E9                                                                     •002Z                             •
------------------------------------------------------------------------------------------------------------------------------------
```

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

Appendix A

PL/1 JCL PROCEDURES

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. For compile steps, SYSLIB references the SYMPL1 data set containing the members to be copied into PL/1 programs via %INCLUDE statements. Optional compile parameters may be added by coding PARM2=',options' on the EXEC statement, for example:

```
        //  EXEC PLIX...,Q=...,NAME=...,PARM2=',MAP,LIST,STORAGE'
```

```
PLIXPC:     PL/1 compile

Example:    // EXEC PLIXPC,Q=TEST,NAME=PL1PROG
------------------------------------------------------------------
PLIXPCL:    PL/1 compile and linkedit for a resident program or
            dynamically loaded program which will be dynamically
            linkedited at Intercomm startup.  NCAL is a required
            linkedit parameter.  (The linkedit step, PARM override
            AMODE=31, RMODE=ANY causes the program to be loaded above
            the 16M line).

Example:    // EXEC PLIXPCL,Q=TEST,NAME=PL1PROG,LMOD=PL1PROG
            // PARM.LKED='LIST,XREF,LET,NCAL,REUS,AMODE=31,RMODE=ANY'

            For dynamically loaded PL/1 subsystems add the following:

            //LKED.SYSIN DD *
                INCLUDE USRLIB(PLIV)
                INCLUDE USRLIB(INTLOAD)
                INCLUDE USRLIB(PLISHRE)   (PL/1 V2 + Shared Library)
                ENTRY PLIV
                NAME program-name(R)

            For linkediting all PL/1 subroutines add the following:

            //LKED.SYSIN DD *
                ENTRY subroutine-name
                INCLUDE USRLIB(INTLOAD)   (if dynamically loaded)
                INCLUDE USRLIB(PLISHRE)   (PL/1 V2 - see below)
                NAME subroutine-name(R)

            Note that the INCLUDE statement for INTLOAD may be omitted
            if the only external call is to PMIPL1.
```

Figure A-1. Intercomm-supplied PL/1 JCL Procedures

Refer to the Intercomm Operating Reference Manual for further details on JCL parameter requirements, and Intercomm linkedit with PL/1 subsystems. Note that if PL/1 subsystems and subroutines are both included in the Intercomm load module, at least one PL/1 subsystem must be included before any PL/1 subroutines. IBMB... subroutines which may need to be included in the Intercomm load module can be determined from the program linkedit (unresolved external references; ignore WX (weak) references), or by using the ESD compile option.

If using PL/1 Version 2 with a Shared Library, add an INCLUDE SYSLIB(PLISHRE) to the Intercomm linkedit instead of including IBMB.... subroutines if resident PL/1 programs are used. Ensure the system library containing the PLISHRE module is concatenated after the Intercomm libraries for the SYSLIB DD statement. For dynamically loaded PL/1 programs, add an INCLUDE USRLIB(PLISHRE) just before the ENTRY PLIV and ensure the system library containing PLISHRE is concatenated after the Intercomm libraries for the USRLIB DD statement in PLIXPCL, or copy the module to MODLIB, or add a SYSLIB DD statement for the appropriate system library and include PLISHRE from SYSLIB instead of USRLIB.

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 PL/1 program simply by coding %INCLUDE member-name at the desired source line. SYMREL must be named in the DD statement concatenation for the SYSLIB data set for compilations (if Version 2 of the PL/1 compiler is used), or the members must be copied to a source library (SYMPL1) of the appropriate block size for the compiler. That source library must be defined for the SYSLIB data set in the compile JCL (SYMPL1 is the default in Intercomm supplied procedures JCL).

NOTE: The block size of SYMREL is 6160 as released.

PLIENTRY

The PLIENTRY member contains a DECLARE statement specifying Intercomm service routine names as ENTRY with OPTIONS (ASM INTER).

```
DECLARE   ( SELECT,
            RELEASE,
            READ,
            WRITE,
            GET,
            PUT,
            GETV,
            PUTV,
            RELEX,
            FEOV,
            COBPUT,
            ALLOCATE,
            ACCESS,
            MSGCOL,
            FESEND,
            FESENDC,
            COBSTORF,
            CONVERSE,
            LOGPUT,
            DBINT,
            PAGE,
            QBUILD,
            QOPEN,
            QREAD,
            QREADX,
            QWRITE,
            QWRITEX,
            QCLOSE,
            FECMDDQ,
            FECMFDBK,
            FECMRLSE,
            MAPIN,
            MAPOUT,
            MAPFREE,
            MAPEND,
            MAPURGE,
            MAPCLR,
            DWSSNAP,                  (Rel 10 only)
            INTSORTC,                 (Rel 10 only)
            INTSTORE,
            INTFETCH,
            INTUNSTO) ENTRY OPTIONS (ASM INTER);
```

PLMSGHD

The PLMSGHD member contains level 5 declaration clauses naming and listing the attributes for all of the Intercomm message header fields. This member may be used in conjunction with a level 1 DECLARE statement to define a message structure. See Figure 17.

```
      5 MSGHLEN FIXED BIN(15) UNALIGNED,
      5 MSGHQPR CHAR (1),
      5 MSGHRSCH BIT (8) ALIGNED,
      5 MSGHRSC BIT (8) ALIGNED,
      5 MSGHSSC BIT (8) ALIGNED,
      5 MSGHMMN BIT (24) ALIGNED,
      5 MSGHDAT CHAR (6),
      5 MSGHTIM CHAR (8),
      5 MSGHTID CHAR (5),
      5 MSGHCON BIT (16) ALIGNED,
      5 MSGHFLGS CHAR (2),
      5 MSGHBMN BIT (24) ALIGNED,               (MSGHADDR - Rel 9)
      5 MSGHSSCH BIT (8) ALIGNED,
      5 MSGHUSR CHAR (1),
      5 MSGHADDR BIT (16) ALIGNED,              (MSGHBMN - Rel 9)
      5 MSGHLOG CHAR (1),
      5 MSGHBLK BIT (8) ALIGNED,
      5 MSGHVMI BIT (8) ALIGNED,
```

PL1HDR

The PL1HDR member contains a BASED structure detailing the fields in the Intercomm message header, and can be used together with the address of the input message to reference individual header fields directly in the input message area. ADDRESS_OF_INPUT_MESSAGE must be declared as a pointer variable and initialized if necessary (see Chapter 3).

```
DCL 1 MESSAGE_IN BASED (ADDRESS_OF_INPUT_MESSAGE),
     2 MSGHLEN FIXED BIN(15) UNALIGNED,
     2 MSGHQPR CHAR (1),
     2 MSGHRSCH_MSGHRSC BIT (16) ALIGNED,
     2 MSGHSSC BIT (8) ALIGNED,
     2 MSGHMMN BIT (24) ALIGNED,
     2 MSGHDAT CHAR (6),
     2 MSGHTIM CHAR (8),
     2 MSGHTID CHAR (5),
     2 MSGHCON BIT (16) ALIGNED,
     2 MSGHFLGS CHAR (2),
     2 MSGHBMN BIT (24) ALIGNED,                (MSGHADDR - Rel 9)
     2 MSGHSSCH BIT (8) ALIGNED,
     2 MSGHUSR CHAR (1),
     2 MSGHADDR CHAR (2) ALIGNED,               (MSGHBMN - Rel 9)
     2 MSGHLOG CHAR (1),
     2 MSGHBLK BIT (8) ALIGNED,
     2 MSGHVMI BIT (8) ALIGNED,
     2 MSGHEND /* USER FIELDS START HERE */,
```

PENTRY

The PENTRY member declares and initializes static variables used in CALLs through PMIPL1 to the named Intercomm system service routines. See sample program using PMIPL1 CALLs in Appendix D.

```
DCL 1 PENTRY STATIC,
  2 ( /*IF OFFSET ODD,TRUE OFFSET=-(OFFSET+1)*/
      INTSORTC             INIT(99),            (Rel 10 only)
      DWSSNAP              INIT(95),            (Rel 10 only)
      MAPFREE              INIT(91),
      FECMRLSE             INIT(87),
      FESEND               INIT(83),
      FESENDC              INIT(79),
      ALLOCATE             INIT(75),
      ACCESS               INIT(71),
      MAPURGE              INIT(67),
      MAPCLR               INIT(63),
      MAPEND               INIT(59),
      MAPOUT               INIT(55),
      MAPIN                INIT(51),
      INTUNSTO             INIT(47),
      INTSTORE             INIT(43),
      INTFETCH             INIT(39),
      FECMFDBK             INIT(35),
      FECMDDQ              INIT(31),
      QWRITEX              INIT(27),
      QREADX               INIT(23),
      QWRITE               INIT(19),
      QREAD                INIT(15),
      QCLOSE               INIT(11),
      QOPEN                INIT( 7),
      QBUILD               INIT( 3),
      SELECT               INIT( 4),
      RELEASE              INIT( 8),
      READ                 INIT(12),
      WRITE                INIT(16),
      GET                  INIT(20),
      PUT                  INIT(24),
      RELEX                INIT(28),
      FEOV                 INIT(32),
      COBPUT               INIT(68),
      MSGCOL               INIT(72),
      COBSTORF             INIT(76),
      CONVERSE             INIT(80),
      DBINT                INIT(84),
      LOGPUT               INIT(88),
      PAGE                 INIT(92),
      GETV                 INIT(96),
      PUTV                 INIT(100) )
  FIXED BIN(15);
```

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)

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
KEYTABLE	*Display Utility Key Transformation Routing Table	DC's	None
PADDTBLE	*Edit Utility Pad Table	PADD macro	PADDTBLE
PAGETBL	*Page Facility Table	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	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
VERBTBL	*Edit Control Table	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
Message Mapping Utilities	MMUVTBL LOGCHARS PMIDEVTB PMISTATB User-coded Maps
Monitor	REENTSBS INTSPA INTSCT BROADCST
Output Utility	PMIALTRP PMIDEVTB PMIFILET PMIRCNTB PMIRPTAB PMISTATB REPTAPE RPTnnnnn (user-coded OFTs)
Page Facility	PAGETBL

Figure C-2. Components and Associated Table Names

Appendix D

USING PMIPL1

D.1 INTRODUCTION

PMIPL1 was originally developed as an Intercomm service routine interface for PL/1 F compiled programs. Intercomm (and user) subroutines could not be defined as Assembler routines, and therefore the function of PMIPL1 was to convert the PL/1 F parameter list for a call to an Assembler routine to a standard Assembler parameter list. The PL/1 parameter list contained Dope Vectors for all non-arithmetic parameters (for character and bit strings). Under the Optimizer compiler, the Dope Vectors became Locator/Descriptors, but the basic structure (function) is the same. For calls to Assembler routines (Intercomm or user), PMIPL1 creates a new parameter list with the data field addresses from the Dope Vectors (Locator/Descriptors). For calls to user PL/1 subroutines, the original parameter list is copied and passed. All calls to Intercomm and user routines are made via PMIPL1 which must be declared as follows:

```
DCL  PMIPL1  EXTERNAL  ENTRY;
```

All called subroutines (Intercomm and user) must be defined by SUBMODS macros in the REENTSBS table as discussed in Chapter 9. The subroutine to be called is given to PMIPL1 via the label of an index code into the REENTSBS table, declared as FIXED BIN(15) and passed as the first parameter on the call to PMIPL1, as follows:

```
CALL  PMIPL1(routine-code-name,parm1,...parmn);
```

The SUBMODS definitions for commonly used Intercomm service routines are provided in the system release version of REENTSBS. Those for other routines and user subroutines have to be added at the end of the table. In addition, a copy member PENTRY (see Appendix B) is provided which gives the routine code names and index values for the Intercomm routines defined in the released REENTSBS. This member is to be copied into each program that uses the PMIPL1 interface via the following statement:

```
%INCLUDE  PENTRY;
```

Labeled index codes for other subroutines may be added to PENTRY or declared separately, as described in Chapter 9. Note that the codes are absolute displacements (in increments of 4) to SUBMODS base definitions in the REENTSBS table. Once entries are added, they should never be deleted, new entries are always added at the end. This will ease maintenance of program code.

Under the Optimizing compiler, since Intercomm and user routines can be declared as ENTRY OPTIONS(ASM INTER) and thus will receive a standard Assembler parameter list for both arithmetic and character data fields, the use of the PMIPL1 interface is no longer necessary; direct calls can be made (see Chapter 3). Note, however, that when a pointer variable is passed as a parameter, the address of the pointer address is always passed no matter how the called routine is defined. Thus Optimizer PL/1 subroutines which receive only pointer variables and/or arithmetic fields as parameters can be declared as Assembler subroutines; the parameter list is the same. If character or bit strings are passed, Locators are generated and the Locator address is passed if the routine is declared as ENTRY EXTERNAL. For structures or arrays, set a pointer variable to the beginning of the area and pass that for easy definition of the area in the called routine.

A sample program which combines the logic of the sample programs in Chapter 10, but uses PMIPL1 calls instead of direct calls, is given at the end of this Appendix. Note that this program is eligible for loading above the 16M line under Release 10 (all passed parameters in program's DSA except the routine-code-names). Note also that fullword-aligned areas passed as character string variables to Intercomm routines require use of a DEFINED statement to reference subfields in the string (see FHCW and MCW).

D.2 PMIPL1 PARAMETER LISTS

For programs using PMIPL1 under the Optimizing compiler, all parameters passed on calls to PMIPL1 (except the routine-code-name) must be non-arithmetic or pointer variables if the called routine is Assembler, that is, Locator addresses for the parameters are passed to PMIPL1. When executing under XA or ESA and Release 10, PMIPL1 checks that each parm passed to the called subroutine is a 24-Amode address. PL/1 subroutines are defined on the SUBMODS macro in the REENTSBS table with the TYPE=PL1 parameter. COBOL subroutines should not be called by PL/1 programs, but if used, they are passed Assembler parameter lists by PMIPL1. COBOL subroutines that may be called by Assembler or PL/1 programs must be defined on the SUBMODS macro with USAGE=REUSE or NONREUSE and may not call COBREENT.

Exceptions to the non-arithmetic parameters for PMIPL1 calls for Intercomm service routines are as follows:

MSGCOL - three parameters may be passed (instead of one parameter as described in Chapter 9) as follows:

```
          CALL  PMIPL1(MSGCOL,message,SPA,return-code);
```

where message is the same as in Chapter 9
SPA is the SPA entry parameter
return-code is declared as FIXED BIN(31) and may be the same field as the return-code entry parameter.

CONVERSE - the first parameter must be a fullword-aligned CHAR(4) field, and the second parameter must be a FIXED BIN(31) field as described in Chapter 8.

PAGE - the second parameter (page-return-code) must be declared as FIXED BIN(31), see Page Facility.

MAPIN - the six-parameter form of the call must be used, with the label of the input message area (structure), not the pointer, passed as the fourth parameter (see Message Mapping Utilities - MAPIN call formats).

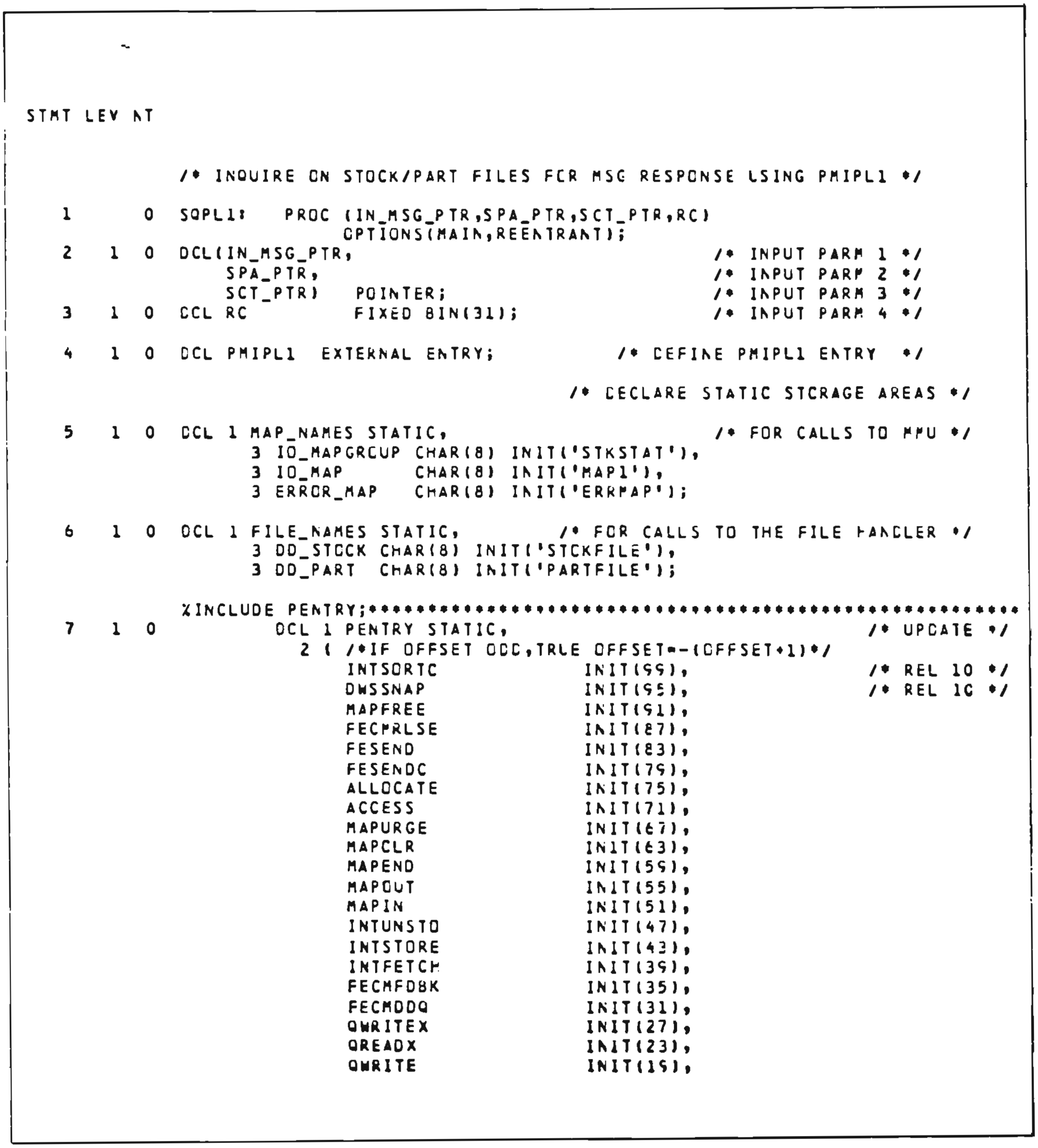
When using MMU, because the input message is mapped into the non-based symbolic map area in the program's DSA, a call to MAPFREE to free the area is not used. See Chapter 4 and compare sample programs in Chapter 10 and this Appendix.

Routines called via PMIPL1 (even if also via DYNLLOAD for dynamically loaded or PL1 subroutines) receive the caller's registers 2 through 13, thus preserving the PL/1 environment for called PL/1 subroutines. At entry to the called routine, register 14 points to a return address in PMIPL1 (or to return code in DYNLLOAD's save area if entry is made via DYNLLOAD), register 15 contains the entry point address and register 1 points to the new parameter list set up by PMIPL1 in an Intercomm storage area acquired by PMIPL1. The routine-code parameter is not passed to the called routine.

To set up the new parameter list, PMIPL1 acquires a 56-byte storage area for a 12-byte processing area followed by an 11-word parameter list area (a larger area is acquired if more than 11 parameters are to be passed). At entry to the called routine, register 1 points to the 13th byte (4th word) in this area. The first 12 bytes are used as follows:

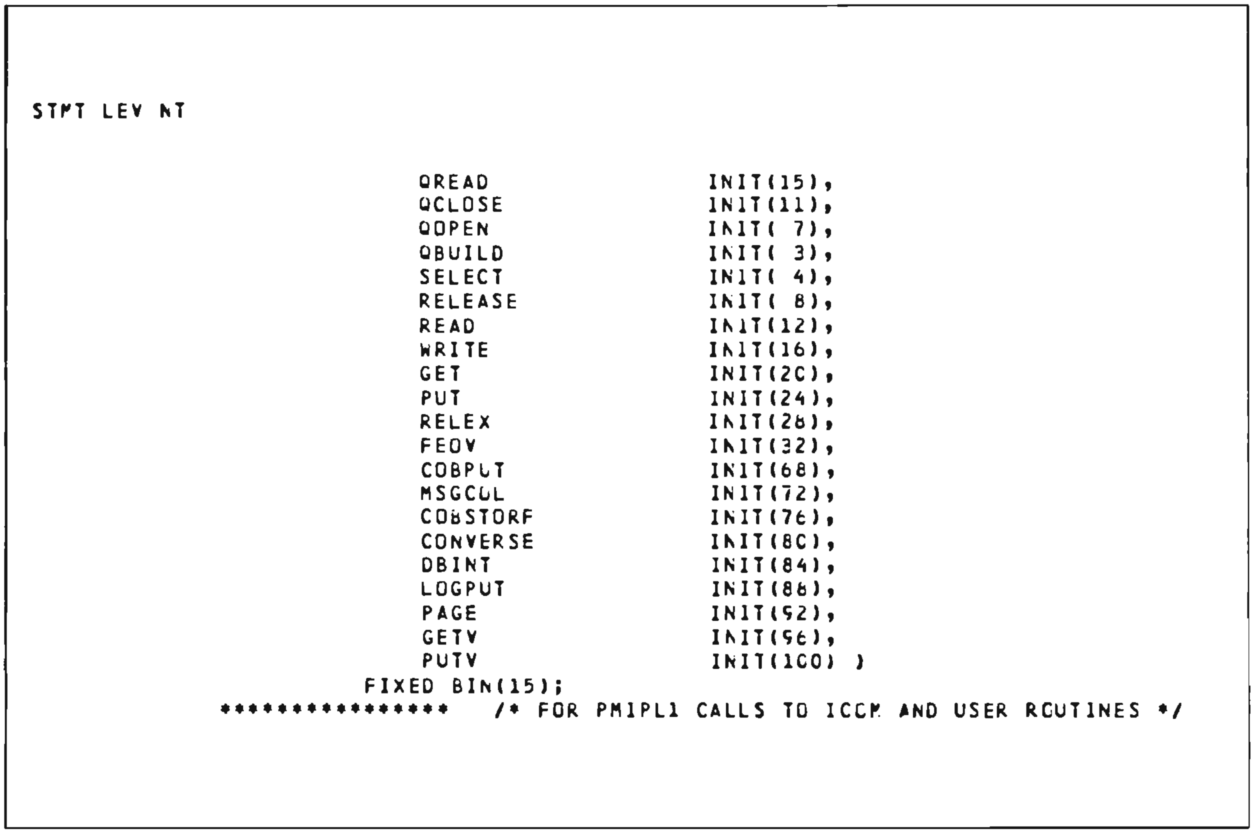
bytes 1-2 - halfword length (in binary) of area
3-4 - halfword routine-code value passed to PMIPL1
5-8 - fullword containing caller's original register 1 value (parameter list address)
9-12 - fullword containing caller's original return address (register 14 at entry to PMIPL1 with hi-order byte (bit if 31-Amode address) cleared to binary zeros).

On return to PMIPL1, the caller's original registers 1 and 14 are restored to the caller's save area before PMIPL1's parameter list storage area is freed. Register 15 in the caller's save area will contain a return code from the called Assembler routine if stored there by the called routine (or provided via RC parameter on RTNLINK macro), see Assembler Language Programmers Guide. On return from PMIPL1, the caller's registers 1 through 14 are restored and branch exit to its caller is effected via register 14 from PMIPL1.



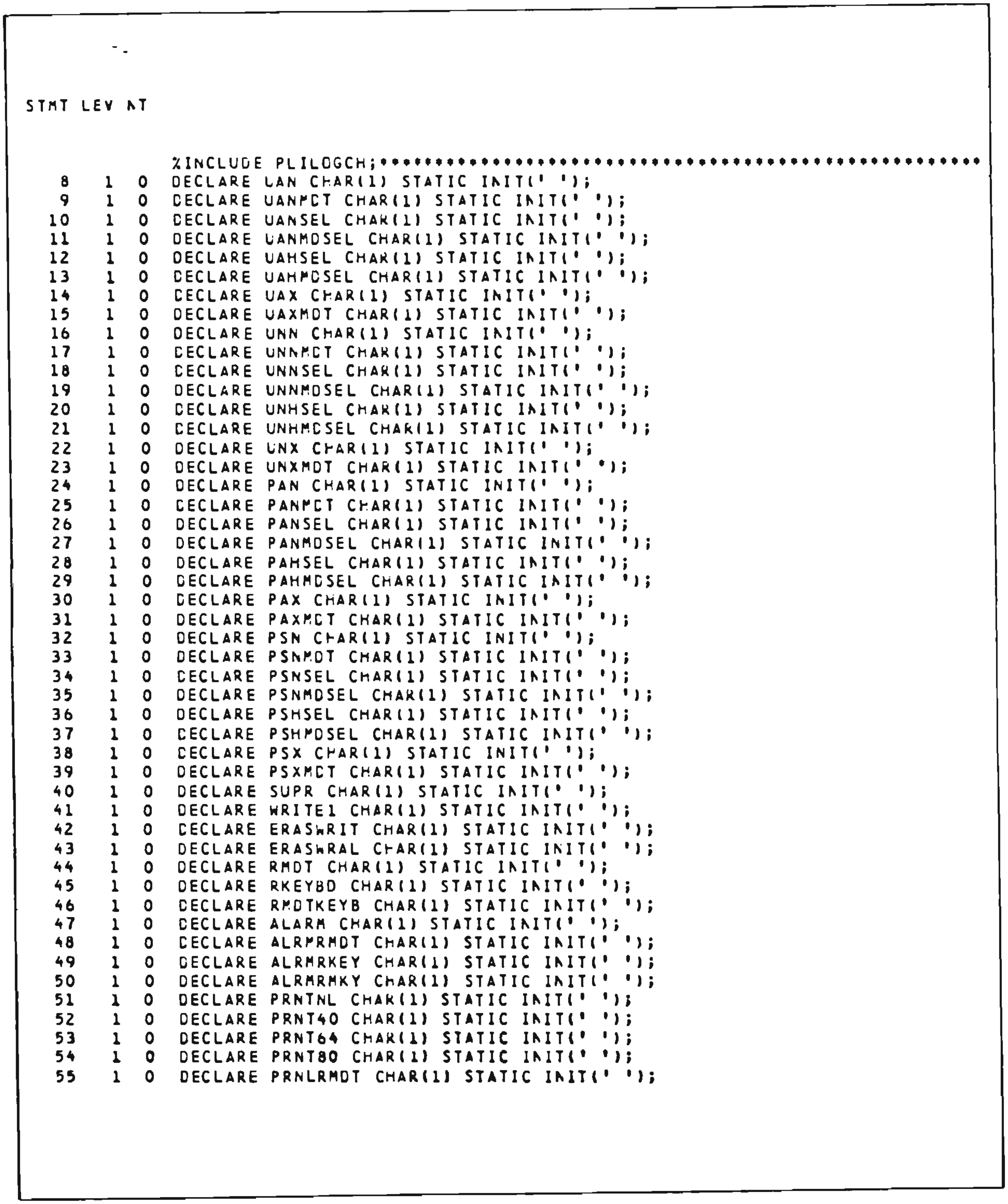
```
 STMT LEV NT

                /* INQUIRE ON STOCK/PART FILES FOR MSG RESPONSE USING PMIPL1 */

    1       0   SQPL1:    PROC (IN_MSG_PTR,SPA_PTR,SCT_PTR,RC)
                               OPTIONS(MAIN,REENTRANT);
    2    1  0   DCL(IN_MSG_PTR,                                   /* INPUT PARM 1 */
                     SPA_PTR,                                     /* INPUT PARM 2 */
                     SCT_PTR)    POINTER;                         /* INPUT PARM 3 */
    3    1  0   DCL RC          FIXED BIN(31);                    /* INPUT PARM 4 */

    4    1  0   DCL PMIPL1  EXTERNAL ENTRY;           /* DEFINE PMIPL1 ENTRY  */

                                                 /* DECLARE STATIC STORAGE AREAS */

    5    1  0   DCL 1 MAP_NAMES STATIC,                           /* FOR CALLS TO MPU */
                        3 IO_MAPGROUP CHAR(8) INIT('STKSTAT'),
                        3 IO_MAP      CHAR(8) INIT('MAP1'),
                        3 ERROR_MAP   CHAR(8) INIT('ERRMAP');

    6    1  0   DCL 1 FILE_NAMES STATIC,         /* FOR CALLS TO THE FILE HANDLER */
                        3 DD_STOCK CHAR(8) INIT('STCKFILE'),
                        3 DD_PART  CHAR(8) INIT('PARTFILE');

                %INCLUDE PENTRY;*****************************************************
    7    1  0           DCL 1 PENTRY STATIC,                              /* UPDATE */
                          2 ( /*IF OFFSET ODD,TRUE OFFSET=-(OFFSET+1)*/
                              INTSORTC               INIT(99),            /* REL 10 */
                              DWSSNAP                INIT(95),            /* REL 10 */
                              MAPFREE                INIT(91),
                              FECMRLSE               INIT(87),
                              FESEND                 INIT(83),
                              FESENDC                INIT(79),
                              ALLOCATE               INIT(75),
                              ACCESS                 INIT(71),
                              MAPURGE                INIT(67),
                              MAPCLR                 INIT(63),
                              MAPEND                 INIT(59),
                              MAPOUT                 INIT(55),
                              MAPIN                  INIT(51),
                              INTUNSTO               INIT(47),
                              INTSTORE               INIT(43),
                              INTFETCH               INIT(39),
                              FECMFDBK               INIT(35),
                              FECMDDQ                INIT(31),
                              QWRITEX                INIT(27),
                              QREADX                 INIT(23),
                              QWRITE                 INIT(19),
```

Figure D-1.   Sample PL/1 Program Calling PMIPL1 (Page 1 of 15)



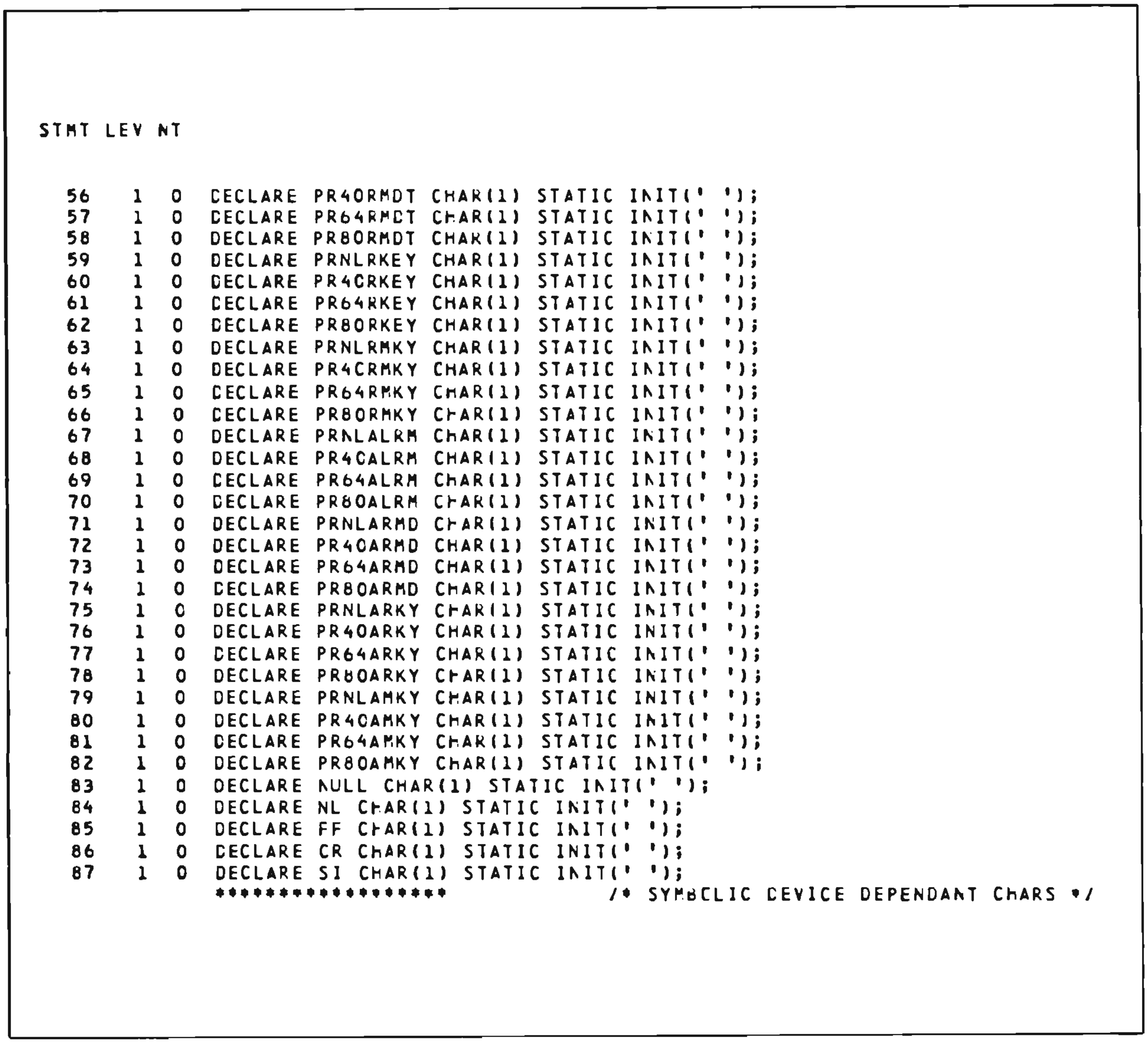
```
STMT LEV NT

                              QREAD                INIT(15),
                              QCLOSE               INIT(11),
                              QOPEN                INIT( 7),
                              QBUILD               INIT( 3),
                              SELECT               INIT( 4),
                              RELEASE              INIT( 8),
                              READ                 INIT(12),
                              WRITE                INIT(16),
                              GET                  INIT(20),
                              PUT                  INIT(24),
                              RELEX                INIT(28),
                              FEOV                 INIT(32),
                              COBPUT               INIT(68),
                              MSGCOL               INIT(72),
                              COBSTORF             INIT(76),
                              CONVERSE             INIT(80),
                              DBINT                INIT(84),
                              LOGPUT               INIT(88),
                              PAGE                 INIT(92),
                              GETV                 INIT(96),
                              PUTV                 INIT(100) )
                          FIXED BIN(15);
                 ****************   /* FOR PMIPL1 CALLS TO ICCF AND USER ROUTINES */
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 2 of 15)



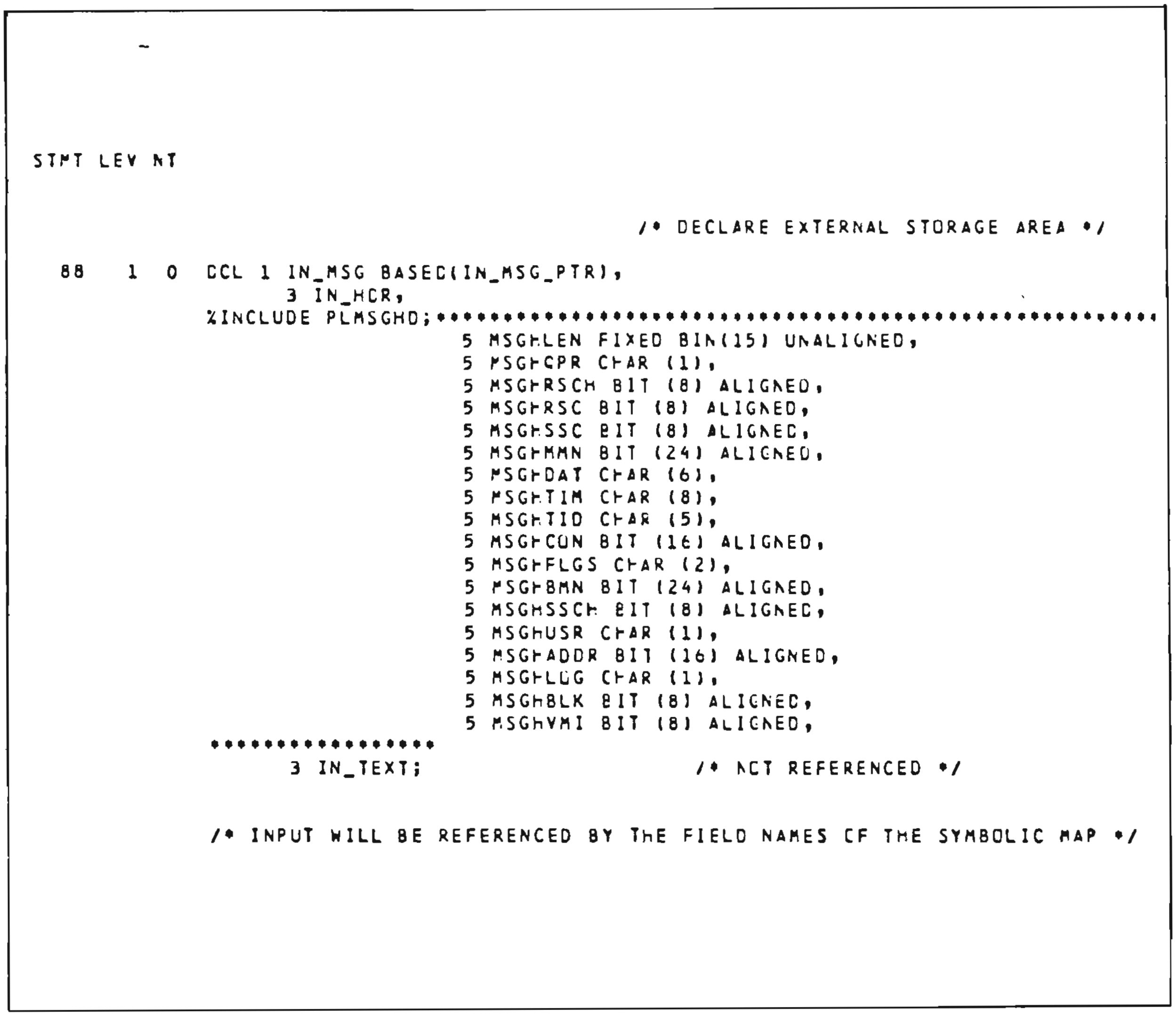
```
STMT LEV NT

              %INCLUDE PLILOGCH;**********************************************************
   8   1  0   DECLARE UAN CHAR(1) STATIC INIT(' ');
   9   1  0   DECLARE UANMDT CHAR(1) STATIC INIT(' ');
  10   1  0   DECLARE UANSEL CHAR(1) STATIC INIT(' ');
  11   1  0   DECLARE UANMDSEL CHAR(1) STATIC INIT(' ');
  12   1  0   DECLARE UAHSEL CHAR(1) STATIC INIT(' ');
  13   1  0   DECLARE UAHMDSEL CHAR(1) STATIC INIT(' ');
  14   1  0   DECLARE UAX CHAR(1) STATIC INIT(' ');
  15   1  0   DECLARE UAXMDT CHAR(1) STATIC INIT(' ');
  16   1  0   DECLARE UNN CHAR(1) STATIC INIT(' ');
  17   1  0   DECLARE UNNMDT CHAR(1) STATIC INIT(' ');
  18   1  0   DECLARE UNNSEL CHAR(1) STATIC INIT(' ');
  19   1  0   DECLARE UNNMDSEL CHAR(1) STATIC INIT(' ');
  20   1  0   DECLARE UNHSEL CHAR(1) STATIC INIT(' ');
  21   1  0   DECLARE UNHMDSEL CHAR(1) STATIC INIT(' ');
  22   1  0   DECLARE UNX CHAR(1) STATIC INIT(' ');
  23   1  0   DECLARE UNXMDT CHAR(1) STATIC INIT(' ');
  24   1  0   DECLARE PAN CHAR(1) STATIC INIT(' ');
  25   1  0   DECLARE PANMDT CHAR(1) STATIC INIT(' ');
  26   1  0   DECLARE PANSEL CHAR(1) STATIC INIT(' ');
  27   1  0   DECLARE PANMDSEL CHAR(1) STATIC INIT(' ');
  28   1  0   DECLARE PAHSEL CHAR(1) STATIC INIT(' ');
  29   1  0   DECLARE PAHMDSEL CHAR(1) STATIC INIT(' ');
  30   1  0   DECLARE PAX CHAR(1) STATIC INIT(' ');
  31   1  0   DECLARE PAXMDT CHAR(1) STATIC INIT(' ');
  32   1  0   DECLARE PSN CHAR(1) STATIC INIT(' ');
  33   1  0   DECLARE PSNMDT CHAR(1) STATIC INIT(' ');
  34   1  0   DECLARE PSNSEL CHAR(1) STATIC INIT(' ');
  35   1  0   DECLARE PSNMDSEL CHAR(1) STATIC INIT(' ');
  36   1  0   DECLARE PSHSEL CHAR(1) STATIC INIT(' ');
  37   1  0   DECLARE PSHMDSEL CHAR(1) STATIC INIT(' ');
  38   1  0   DECLARE PSX CHAR(1) STATIC INIT(' ');
  39   1  0   DECLARE PSXMDT CHAR(1) STATIC INIT(' ');
  40   1  0   DECLARE SUPR CHAR(1) STATIC INIT(' ');
  41   1  0   DECLARE WRITE1 CHAR(1) STATIC INIT(' ');
  42   1  0   DECLARE ERASWRIT CHAR(1) STATIC INIT(' ');
  43   1  0   DECLARE ERASWRAL CHAR(1) STATIC INIT(' ');
  44   1  0   DECLARE RMDT CHAR(1) STATIC INIT(' ');
  45   1  0   DECLARE RKEYBD CHAR(1) STATIC INIT(' ');
  46   1  0   DECLARE RMDTKEYB CHAR(1) STATIC INIT(' ');
  47   1  0   DECLARE ALARM CHAR(1) STATIC INIT(' ');
  48   1  0   DECLARE ALRMRMDT CHAR(1) STATIC INIT(' ');
  49   1  0   DECLARE ALRMRKEY CHAR(1) STATIC INIT(' ');
  50   1  0   DECLARE ALRMRMKY CHAR(1) STATIC INIT(' ');
  51   1  0   DECLARE PRNTNL CHAR(1) STATIC INIT(' ');
  52   1  0   DECLARE PRNT40 CHAR(1) STATIC INIT(' ');
  53   1  0   DECLARE PRNT64 CHAR(1) STATIC INIT(' ');
  54   1  0   DECLARE PRNT80 CHAR(1) STATIC INIT(' ');
  55   1  0   DECLARE PRNLRMDT CHAR(1) STATIC INIT(' ');
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 3 of 15)



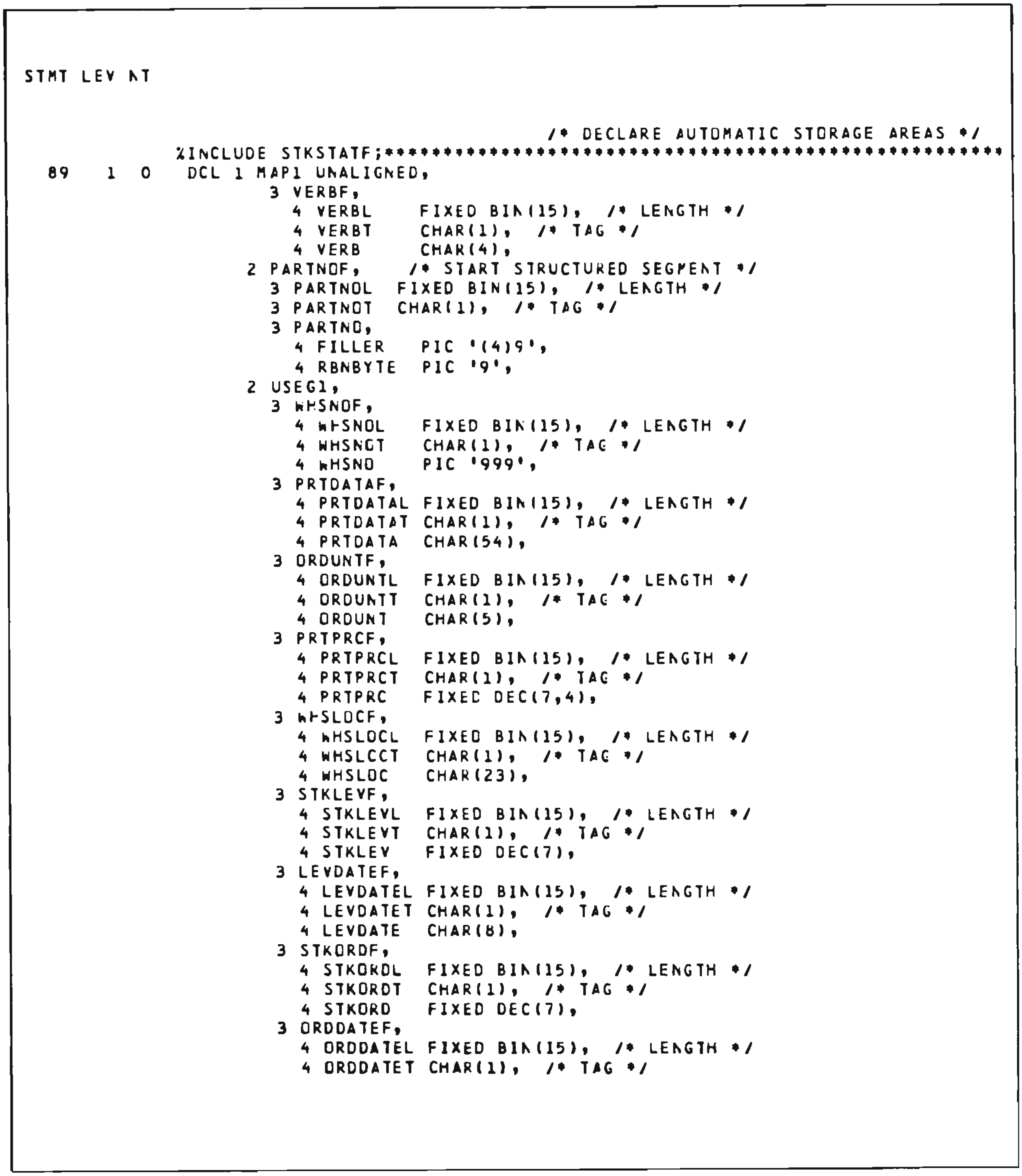
```
STMT LEV NT

  56    1  0  DECLARE PR40RMDT CHAR(1) STATIC INIT(' ');
  57    1  0  DECLARE PR64RMDT CHAR(1) STATIC INIT(' ');
  58    1  0  DECLARE PR80RMDT CHAR(1) STATIC INIT(' ');
  59    1  0  DECLARE PRNLRKEY CHAR(1) STATIC INIT(' ');
  60    1  0  DECLARE PR40RKEY CHAR(1) STATIC INIT(' ');
  61    1  0  DECLARE PR64RKEY CHAR(1) STATIC INIT(' ');
  62    1  0  DECLARE PR80RKEY CHAR(1) STATIC INIT(' ');
  63    1  0  DECLARE PRNLRMKY CHAR(1) STATIC INIT(' ');
  64    1  0  DECLARE PR40RMKY CHAR(1) STATIC INIT(' ');
  65    1  0  DECLARE PR64RMKY CHAR(1) STATIC INIT(' ');
  66    1  0  DECLARE PR80RMKY CHAR(1) STATIC INIT(' ');
  67    1  0  DECLARE PRNLALRM CHAR(1) STATIC INIT(' ');
  68    1  0  DECLARE PR40ALRM CHAR(1) STATIC INIT(' ');
  69    1  0  DECLARE PR64ALRM CHAR(1) STATIC INIT(' ');
  70    1  0  DECLARE PR80ALRM CHAR(1) STATIC INIT(' ');
  71    1  0  DECLARE PRNLARMD CHAR(1) STATIC INIT(' ');
  72    1  0  DECLARE PR40ARMD CHAR(1) STATIC INIT(' ');
  73    1  0  DECLARE PR64ARMD CHAR(1) STATIC INIT(' ');
  74    1  0  DECLARE PR80ARMD CHAR(1) STATIC INIT(' ');
  75    1  0  DECLARE PRNLARKY CHAR(1) STATIC INIT(' ');
  76    1  0  DECLARE PR40ARKY CHAR(1) STATIC INIT(' ');
  77    1  0  DECLARE PR64ARKY CHAR(1) STATIC INIT(' ');
  78    1  0  DECLARE PR80ARKY CHAR(1) STATIC INIT(' ');
  79    1  0  DECLARE PRNLAMKY CHAR(1) STATIC INIT(' ');
  80    1  0  DECLARE PR40AMKY CHAR(1) STATIC INIT(' ');
  81    1  0  DECLARE PR64AMKY CHAR(1) STATIC INIT(' ');
  82    1  0  DECLARE PR80AMKY CHAR(1) STATIC INIT(' ');
  83    1  0  DECLARE NULL CHAR(1) STATIC INIT(' ');
  84    1  0  DECLARE NL CHAR(1) STATIC INIT(' ');
  85    1  0  DECLARE FF CHAR(1) STATIC INIT(' ');
  86    1  0  DECLARE CR CHAR(1) STATIC INIT(' ');
  87    1  0  DECLARE SI CHAR(1) STATIC INIT(' ');
              ******************                /* SYMBOLIC DEVICE DEPENDANT CHARS */
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 4 of 15)



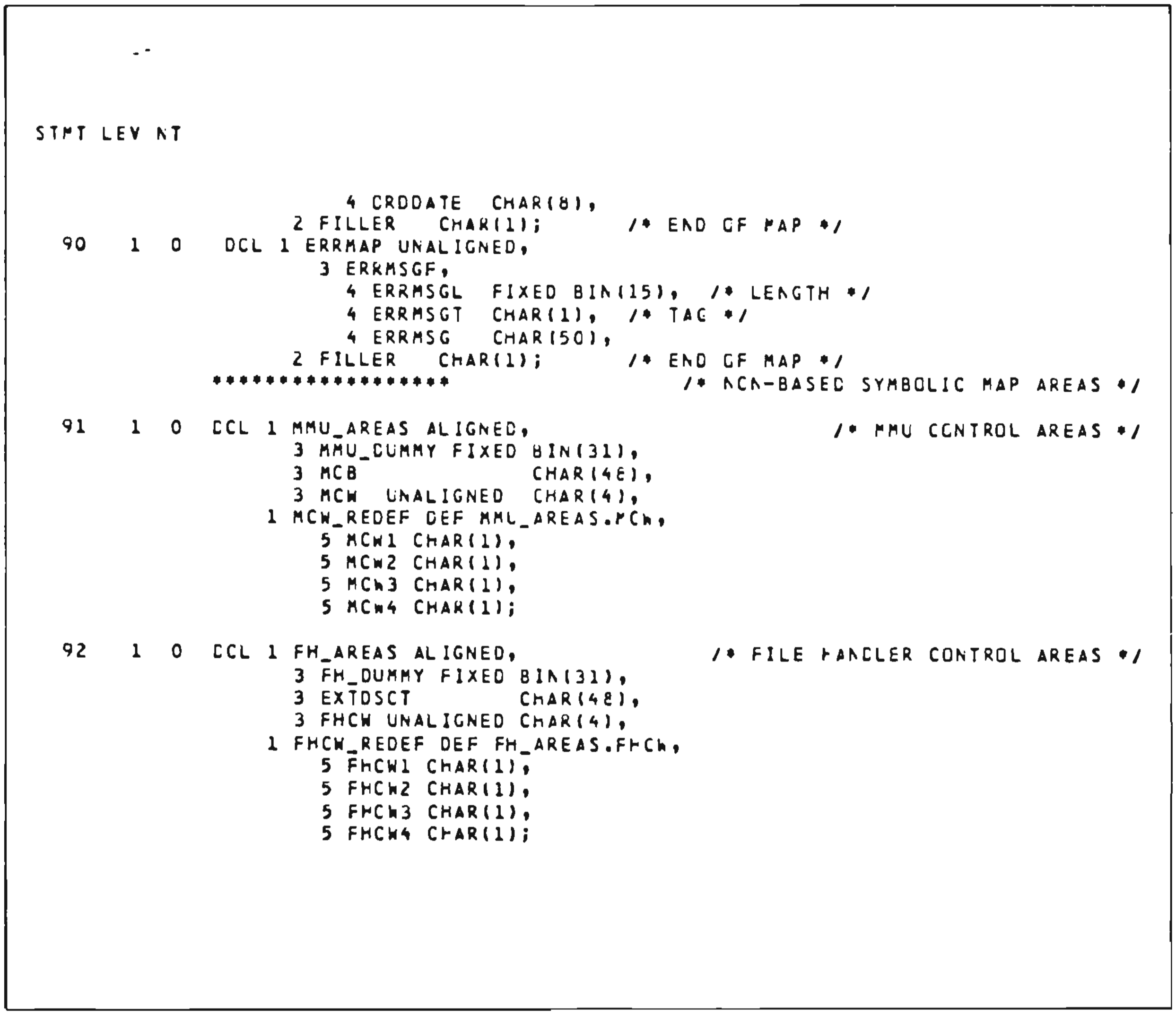
```
STMT LEV NT

                                              /* DECLARE EXTERNAL STORAGE AREA */

  88    1   0  DCL 1 IN_MSG BASED(IN_MSG_PTR),
                     3 IN_HDR,
               %INCLUDE PLMSGHD;**********************************************************
                                 5 MSGHLEN FIXED BIN(15) UNALIGNED,
                                 5 MSGHCPR CHAR (1),
                                 5 MSGHRSCH BIT (8) ALIGNED,
                                 5 MSGHRSC BIT (8) ALIGNED,
                                 5 MSGHSSC BIT (8) ALIGNED,
                                 5 MSGHMMN BIT (24) ALIGNED,
                                 5 MSGHDAT CHAR (6),
                                 5 MSGHTIM CHAR (8),
                                 5 MSGHTID CHAR (5),
                                 5 MSGHCON BIT (16) ALIGNED,
                                 5 MSGHFLGS CHAR (2),
                                 5 MSGHBMN BIT (24) ALIGNED,
                                 5 MSGHSSCH BIT (8) ALIGNED,
                                 5 MSGHUSR CHAR (1),
                                 5 MSGHADDR BIT (16) ALIGNED,
                                 5 MSGHLOG CHAR (1),
                                 5 MSGHBLK BIT (8) ALIGNED,
                                 5 MSGHVMI BIT (8) ALIGNED,
               ******************
                     3 IN_TEXT;                    /* NOT REFERENCED */

               /* INPUT WILL BE REFERENCED BY THE FIELD NAMES OF THE SYMBOLIC MAP */
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 5 of 15)



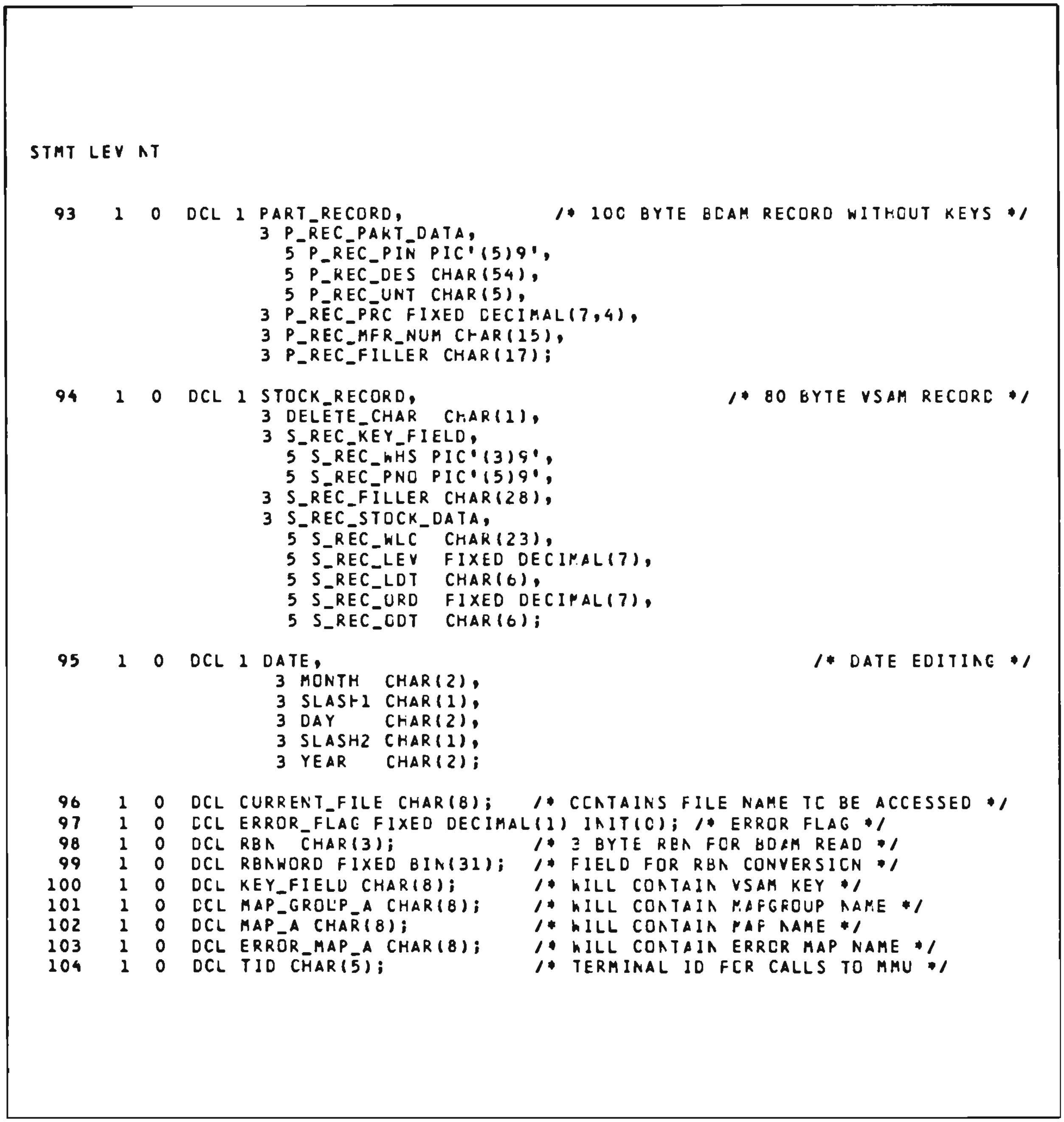
```
STMT LEV NT

                                            /* DECLARE AUTOMATIC STORAGE AREAS */
             %INCLUDE STKSTATF;*******************************************************
  89   1  0   DCL 1 MAP1 UNALIGNED,
                     3 VERBF,
                       4 VERBL    FIXED BIN(15),  /* LENGTH */
                       4 VERBT    CHAR(1),  /* TAG */
                       4 VERB     CHAR(4),
                   2 PARTNOF,    /* START STRUCTURED SEGMENT */
                     3 PARTNOL  FIXED BIN(15),  /* LENGTH */
                     3 PARTNOT  CHAR(1),  /* TAG */
                     3 PARTNO,
                       4 FILLER   PIC '(4)9',
                       4 RBNBYTE  PIC '9',
                   2 USEG1,
                     3 WHSNOF,
                       4 WHSNOL   FIXED BIN(15),  /* LENGTH */
                       4 WHSNOT   CHAR(1),  /* TAG */
                       4 WHSNO    PIC '999',
                     3 PRTDATAF,
                       4 PRTDATAL FIXED BIN(15),  /* LENGTH */
                       4 PRTDATAT CHAR(1),  /* TAG */
                       4 PRTDATA  CHAR(54),
                     3 ORDUNTF,
                       4 ORDUNTL  FIXED BIN(15),  /* LENGTH */
                       4 ORDUNTT  CHAR(1),  /* TAG */
                       4 ORDUNT   CHAR(5),
                     3 PRTPRCF,
                       4 PRTPRCL  FIXED BIN(15),  /* LENGTH */
                       4 PRTPRCT  CHAR(1),  /* TAG */
                       4 PRTPRC   FIXED DEC(7,4),
                     3 WHSLOCF,
                       4 WHSLOCL  FIXED BIN(15),  /* LENGTH */
                       4 WHSLCCT  CHAR(1),  /* TAG */
                       4 WHSLOC   CHAR(23),
                     3 STKLEVF,
                       4 STKLEVL  FIXED BIN(15),  /* LENGTH */
                       4 STKLEVT  CHAR(1),  /* TAG */
                       4 STKLEV   FIXED DEC(7),
                     3 LEVDATEF,
                       4 LEVDATEL FIXED BIN(15),  /* LENGTH */
                       4 LEVDATET CHAR(1),  /* TAG */
                       4 LEVDATE  CHAR(8),
                     3 STKORDF,
                       4 STKORDL  FIXED BIN(15),  /* LENGTH */
                       4 STKORDT  CHAR(1),  /* TAG */
                       4 STKORD   FIXED DEC(7),
                     3 ORDDATEF,
                       4 ORDDATEL FIXED BIN(15),  /* LENGTH */
                       4 ORDDATET CHAR(1),  /* TAG */
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 6 of 15)



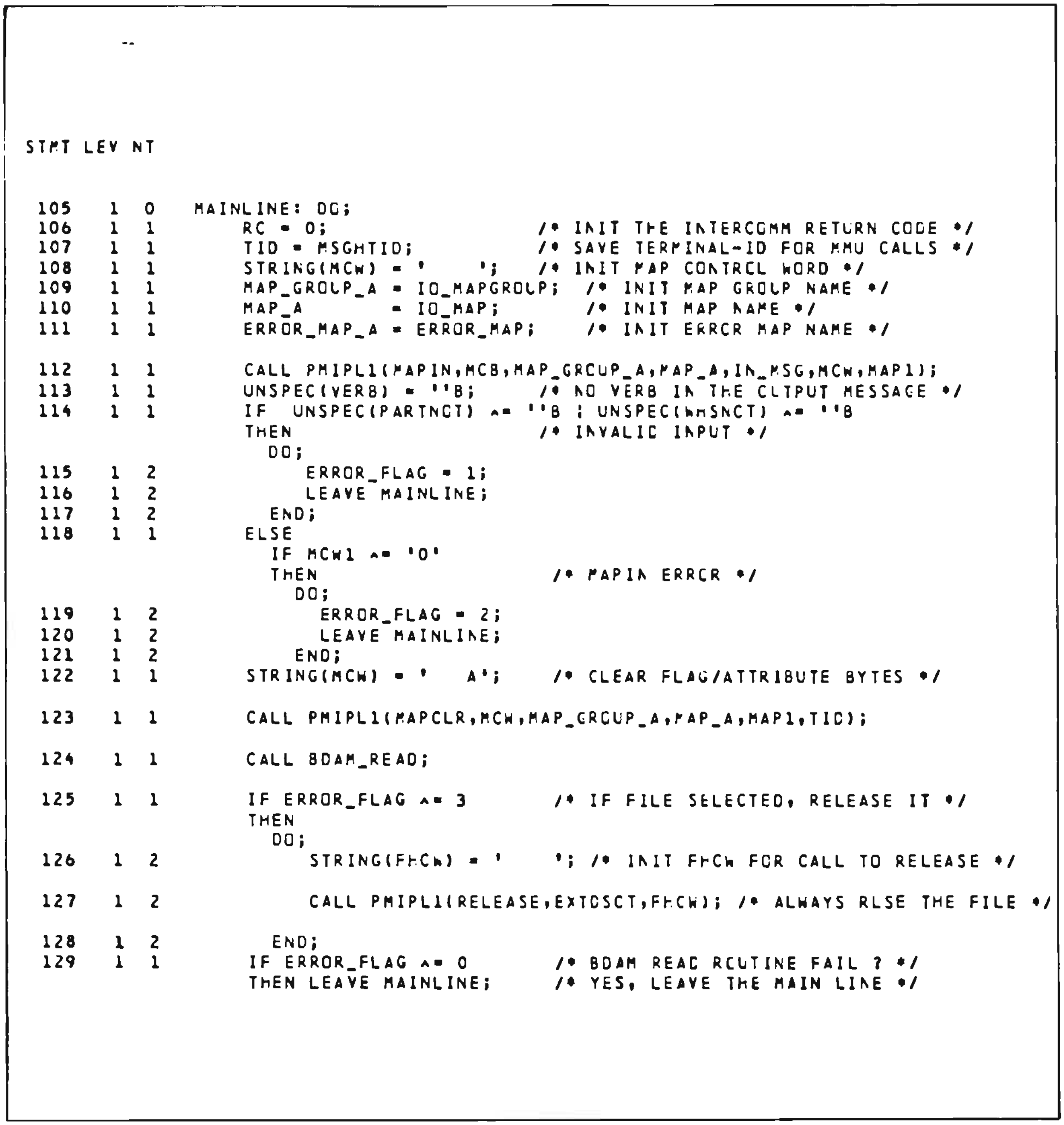
```
STMT LEV NT

                    4 CRDDATE  CHAR(8),
                2 FILLER   CHAR(1);       /* END OF MAP */
  90   1  0   DCL 1 ERRMAP UNALIGNED,
                  3 ERRMSGF,
                    4 ERRMSGL  FIXED BIN(15),  /* LENGTH */
                    4 ERRMSGT  CHAR(1),  /* TAG */
                    4 ERRMSG   CHAR(50),
                2 FILLER   CHAR(1);       /* END OF MAP */
             ******************                /* NON-BASED SYMBOLIC MAP AREAS */

  91   1  0  DCL 1 MMU_AREAS ALIGNED,                       /* MMU CONTROL AREAS */
                3 MMU_DUMMY FIXED BIN(31),
                3 MCB              CHAR(48),
                3 MCW  UNALIGNED  CHAR(4),
              1 MCW_REDEF DEF MMU_AREAS.MCW,
                  5 MCW1 CHAR(1),
                  5 MCW2 CHAR(1),
                  5 MCW3 CHAR(1),
                  5 MCW4 CHAR(1);

  92   1  0  DCL 1 FH_AREAS ALIGNED,               /* FILE HANDLER CONTROL AREAS */
                3 FH_DUMMY FIXED BIN(31),
                3 EXTDSCT        CHAR(48),
                3 FHCW UNALIGNED CHAR(4),
              1 FHCW_REDEF DEF FH_AREAS.FHCW,
                  5 FHCW1 CHAR(1),
                  5 FHCW2 CHAR(1),
                  5 FHCW3 CHAR(1),
                  5 FHCW4 CHAR(1);
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 7 of 15)



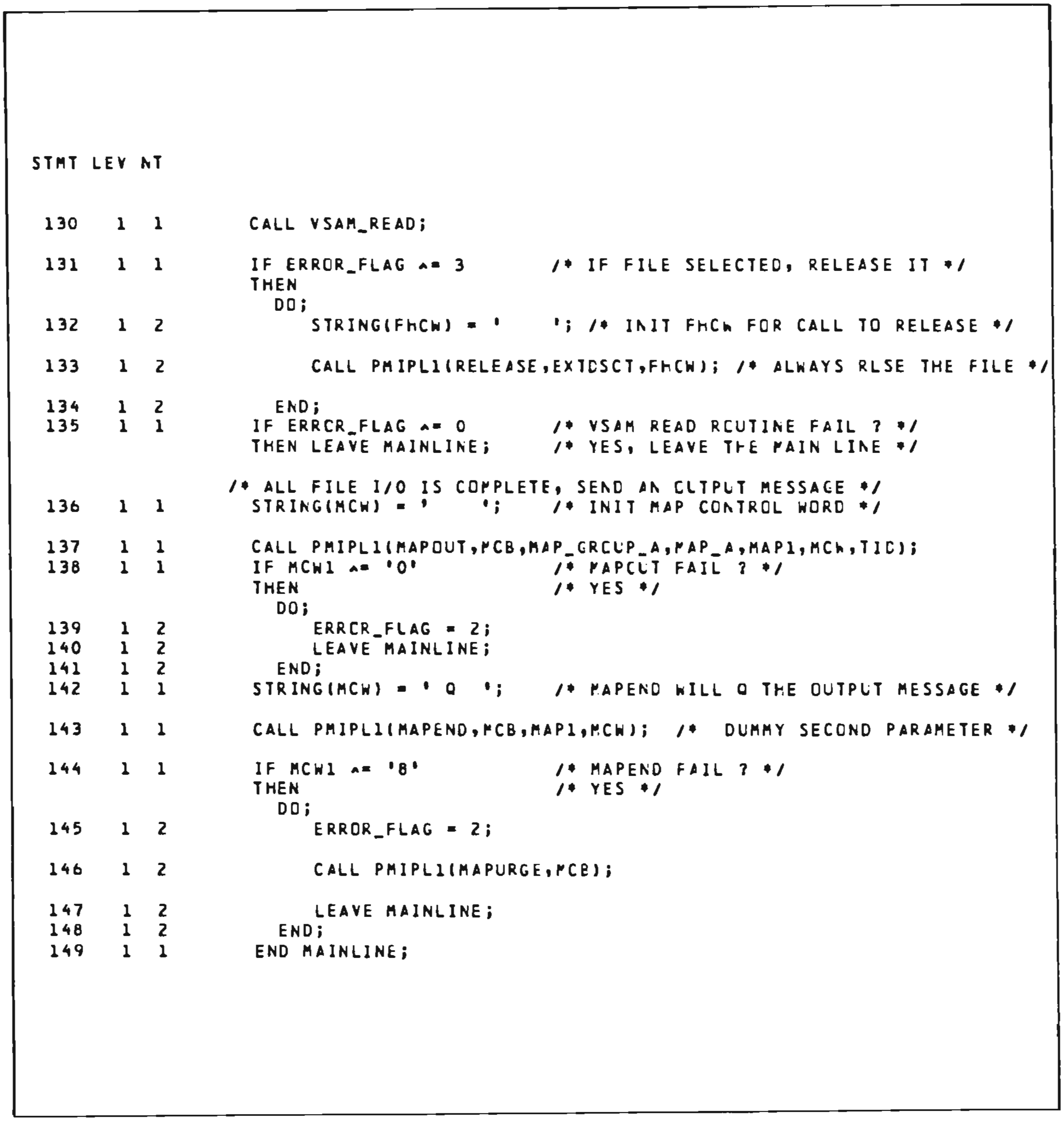
```
STMT LEV NT


  93   1  0  DCL 1 PART_RECORD,            /* 100 BYTE BDAM RECORD WITHOUT KEYS */
                   3 P_REC_PART_DATA,
                     5 P_REC_PIN PIC'(5)9',
                     5 P_REC_DES CHAR(54),
                     5 P_REC_UNT CHAR(5),
                   3 P_REC_PRC FIXED DECIMAL(7,4),
                   3 P_REC_MFR_NUM CHAR(15),
                   3 P_REC_FILLER CHAR(17);

  94   1  0  DCL 1 STOCK_RECORD,                          /* 80 BYTE VSAM RECORD */
                   3 DELETE_CHAR  CHAR(1),
                   3 S_REC_KEY_FIELD,
                     5 S_REC_WHS PIC'(3)9',
                     5 S_REC_PNO PIC'(5)9',
                   3 S_REC_FILLER CHAR(28),
                   3 S_REC_STOCK_DATA,
                     5 S_REC_WLC  CHAR(23),
                     5 S_REC_LEV  FIXED DECIMAL(7),
                     5 S_REC_LDT  CHAR(6),
                     5 S_REC_ORD  FIXED DECIMAL(7),
                     5 S_REC_ODT  CHAR(6);

  95   1  0  DCL 1 DATE,                                         /* DATE EDITING */
                    3 MONTH  CHAR(2),
                    3 SLASH1 CHAR(1),
                    3 DAY    CHAR(2),
                    3 SLASH2 CHAR(1),
                    3 YEAR   CHAR(2);

  96   1  0  DCL CURRENT_FILE CHAR(8);    /* CONTAINS FILE NAME TO BE ACCESSED */
  97   1  0  DCL ERROR_FLAG FIXED DECIMAL(1) INIT(0); /* ERROR FLAG */
  98   1  0  DCL RBN  CHAR(3);            /* 3 BYTE RBN FOR BDAM READ */
  99   1  0  DCL RBNWORD FIXED BIN(31);   /* FIELD FOR RBN CONVERSION */
 100   1  0  DCL KEY_FIELD CHAR(8);       /* WILL CONTAIN VSAM KEY */
 101   1  0  DCL MAP_GROUP_A CHAR(8);     /* WILL CONTAIN MAPGROUP NAME */
 102   1  0  DCL MAP_A CHAR(8);           /* WILL CONTAIN MAP NAME */
 103   1  0  DCL ERROR_MAP_A CHAR(8);     /* WILL CONTAIN ERROR MAP NAME */
 104   1  0  DCL TID CHAR(5);             /* TERMINAL ID FOR CALLS TO MMU */
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 8 of 15)



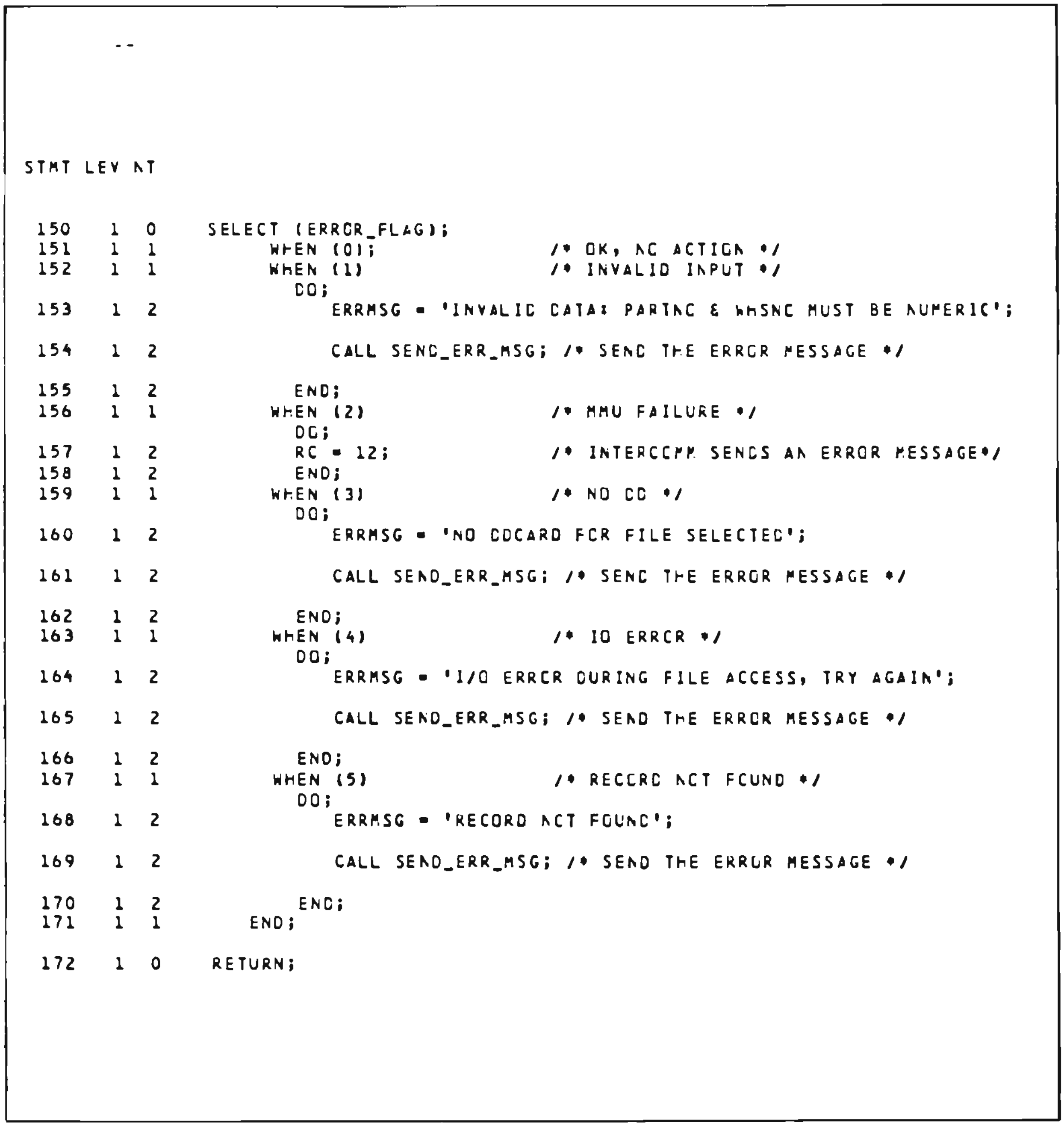
```
        --




STMT LEV NT


 105   1  0   MAINLINE: DO;
 106   1  1       RC = 0;                     /* INIT THE INTERCOMM RETURN CODE */
 107   1  1       TID = MSGHTID;              /* SAVE TERMINAL-ID FOR MMU CALLS */
 108   1  1       STRING(MCW) = '    ';       /* INIT MAP CONTROL WORD */
 109   1  1       MAP_GROUP_A = IO_MAPGROUP;   /* INIT MAP GROUP NAME */
 110   1  1       MAP_A       = IO_MAP;        /* INIT MAP NAME */
 111   1  1       ERROR_MAP_A = ERROR_MAP;     /* INIT ERROR MAP NAME */

 112   1  1       CALL PMIPL1(MAPIN,MCB,MAP_GROUP_A,MAP_A,IN_MSG,MCW,MAP1);
 113   1  1       UNSPEC(VERB) = ''B;         /* NO VERB IN THE OUTPUT MESSAGE */
 114   1  1       IF  UNSPEC(PARTNCT) ^= ''B | UNSPEC(WMSNCT) ^= ''B
                  THEN                        /* INVALID INPUT */
                    DO;
 115   1  2             ERROR_FLAG = 1;
 116   1  2             LEAVE MAINLINE;
 117   1  2         END;
 118   1  1       ELSE
                    IF MCW1 ^= '0'
                    THEN                        /* MAPIN ERROR */
                      DO;
 119   1  2             ERROR_FLAG = 2;
 120   1  2             LEAVE MAINLINE;
 121   1  2           END;
 122   1  1       STRING(MCW) = '   A';         /* CLEAR FLAG/ATTRIBUTE BYTES */

 123   1  1       CALL PMIPL1(MAPCLR,MCW,MAP_GROUP_A,MAP_A,MAP1,TID);

 124   1  1       CALL BDAM_READ;

 125   1  1       IF ERROR_FLAG ^= 3          /* IF FILE SELECTED, RELEASE IT */
                  THEN
                    DO;
 126   1  2           STRING(FHCW) = '    '; /* INIT FHCW FOR CALL TO RELEASE */

 127   1  2           CALL PMIPL1(RELEASE,EXTDSCT,FHCW); /* ALWAYS RLSE THE FILE */

 128   1  2         END;
 129   1  1       IF ERROR_FLAG ^= 0          /* BDAM READ ROUTINE FAIL ? */
                  THEN LEAVE MAINLINE;        /* YES, LEAVE THE MAIN LINE */
```

Figure D-1.   Sample PL/1 Program Calling PMIPL1 (Page 9 of 15)



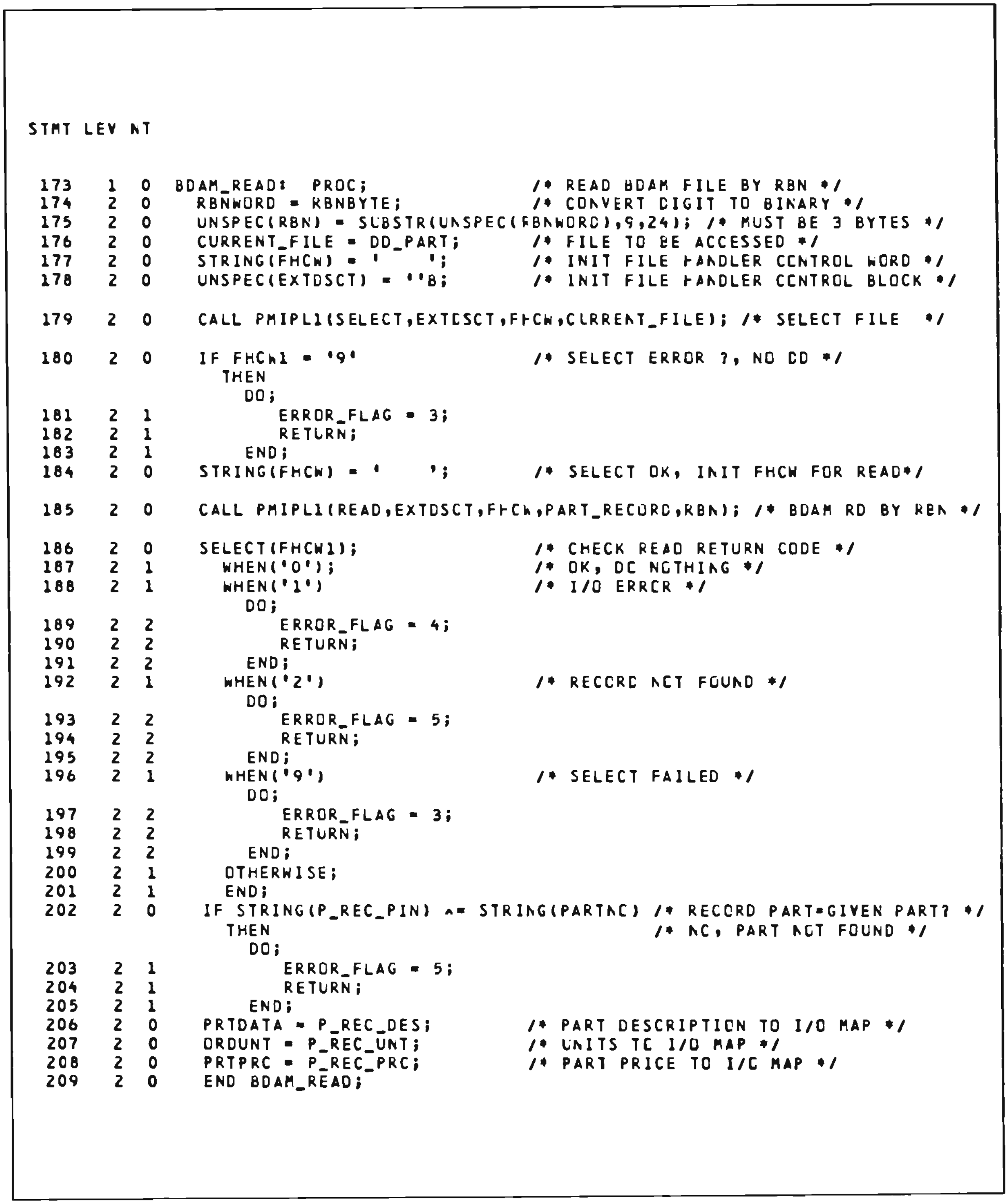
```
STMT LEV NT


 130   1  1       CALL VSAM_READ;

 131   1  1       IF ERROR_FLAG ^= 3        /* IF FILE SELECTED, RELEASE IT */
                  THEN
                    DO;
 132   1  2           STRING(FHCW) = '    '; /* INIT FHCW FOR CALL TO RELEASE */

 133   1  2           CALL PMIPL1(RELEASE,EXTDSCT,FHCW); /* ALWAYS RLSE THE FILE */

 134   1  2         END;
 135   1  1       IF ERROR_FLAG ^= 0        /* VSAM READ ROUTINE FAIL ? */
                  THEN LEAVE MAINLINE;      /* YES, LEAVE THE MAIN LINE */

                /* ALL FILE I/O IS COMPLETE, SEND AN OUTPUT MESSAGE */
 136   1  1       STRING(MCW) = '    ';     /* INIT MAP CONTROL WORD */

 137   1  1       CALL PMIPL1(MAPOUT,MCB,MAP_GROUP_A,MAP_A,MAP1,MCW,TIC);
 138   1  1       IF MCW1 ^= '0'            /* MAPOUT FAIL ? */
                  THEN                      /* YES */
                    DO;
 139   1  2           ERROR_FLAG = 2;
 140   1  2           LEAVE MAINLINE;
 141   1  2         END;
 142   1  1       STRING(MCW) = ' Q  ';     /* MAPEND WILL Q THE OUTPUT MESSAGE */

 143   1  1       CALL PMIPL1(MAPEND,MCB,MAP1,MCW);  /*  DUMMY SECOND PARAMETER */

 144   1  1       IF MCW1 ^= '8'            /* MAPEND FAIL ? */
                  THEN                      /* YES */
                    DO;
 145   1  2           ERROR_FLAG = 2;

 146   1  2           CALL PMIPL1(MAPURGE,MCB);

 147   1  2           LEAVE MAINLINE;
 148   1  2         END;
 149   1  1       END MAINLINE;
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 10 of 15)



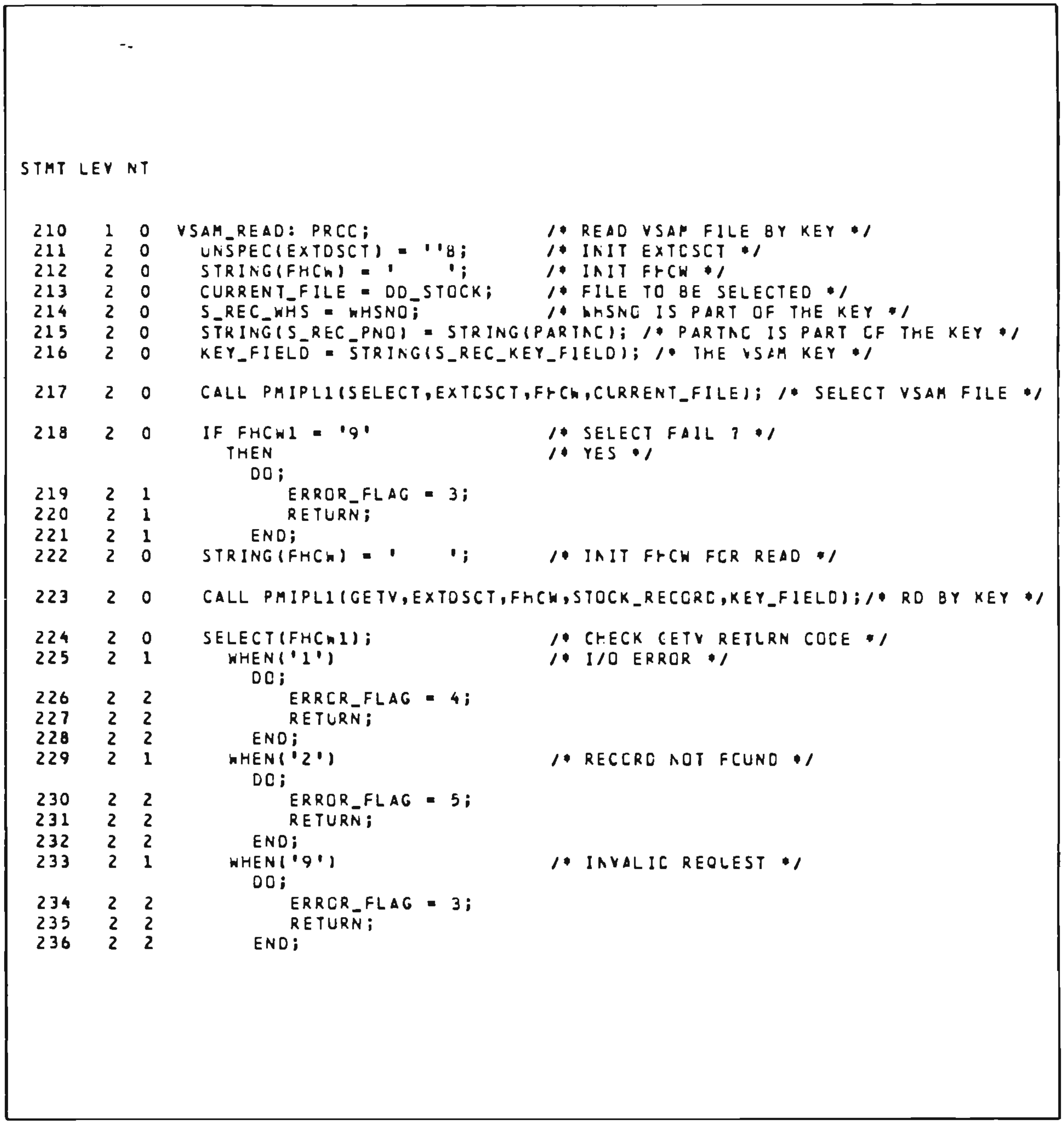
```
      --




STMT LEV NT


 150   1  0    SELECT (ERROR_FLAG);
 151   1  1         WHEN (0);                /* OK, NO ACTION */
 152   1  1         WHEN (1)                 /* INVALID INPUT */
                      DO;
 153   1  2             ERRMSG = 'INVALID DATA: PARTNO & WHSNO MUST BE NUMERIC';

 154   1  2             CALL SEND_ERR_MSG; /* SEND THE ERROR MESSAGE */

 155   1  2          END;
 156   1  1         WHEN (2)                 /* MMU FAILURE */
                      DO;
 157   1  2          RC = 12;                /* INTERCOMM SENDS AN ERROR MESSAGE*/
 158   1  2          END;
 159   1  1         WHEN (3)                 /* NO DD */
                      DO;
 160   1  2             ERRMSG = 'NO DDCARD FOR FILE SELECTED';

 161   1  2             CALL SEND_ERR_MSG; /* SEND THE ERROR MESSAGE */

 162   1  2          END;
 163   1  1         WHEN (4)                 /* IO ERROR */
                      DO;
 164   1  2             ERRMSG = 'I/O ERROR DURING FILE ACCESS, TRY AGAIN';

 165   1  2             CALL SEND_ERR_MSG; /* SEND THE ERROR MESSAGE */

 166   1  2          END;
 167   1  1         WHEN (5)                 /* RECORD NOT FOUND */
                      DO;
 168   1  2             ERRMSG = 'RECORD NOT FOUND';

 169   1  2             CALL SEND_ERR_MSG; /* SEND THE ERROR MESSAGE */

 170   1  2          END;
 171   1  1       END;

 172   1  0    RETURN;
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 11 of 15)



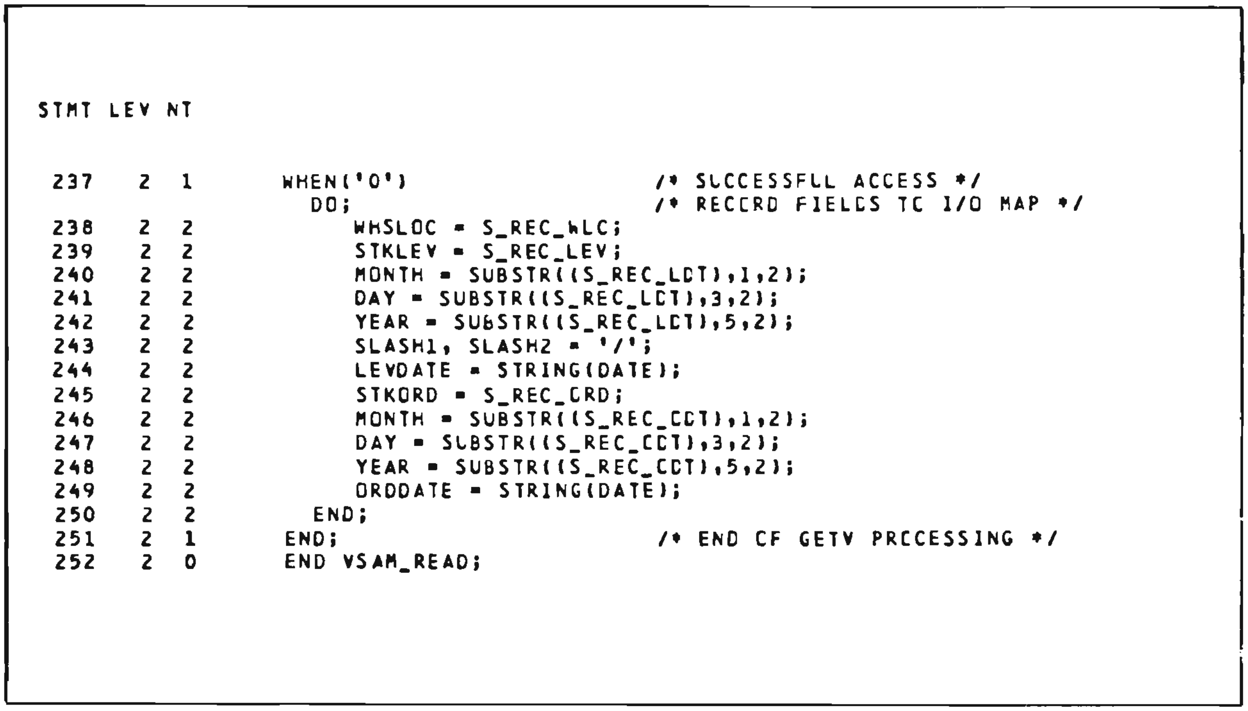
```
STMT LEV NT


 173   1  0  BDAM_READ:  PROC;                /* READ BDAM FILE BY RBN */
 174   2  0    RBNWORD = RBNBYTE;              /* CONVERT DIGIT TO BINARY */
 175   2  0    UNSPEC(RBN) = SUBSTR(UNSPEC(RBNWORD),9,24); /* MUST BE 3 BYTES */
 176   2  0    CURRENT_FILE = DD_PART;         /* FILE TO BE ACCESSED */
 177   2  0    STRING(FHCW) = '    ';          /* INIT FILE HANDLER CONTROL WORD */
 178   2  0    UNSPEC(EXTDSCT) = ''B;          /* INIT FILE HANDLER CONTROL BLOCK */

 179   2  0    CALL PMIPL1(SELECT,EXTDSCT,FHCW,CURRENT_FILE); /* SELECT FILE   */

 180   2  0    IF FHCW1 = '9'                  /* SELECT ERROR ?, NO DD */
                 THEN
                   DO;
 181   2  1          ERROR_FLAG = 3;
 182   2  1          RETURN;
 183   2  1        END;
 184   2  0    STRING(FHCW) = '    ';          /* SELECT OK, INIT FHCW FOR READ*/

 185   2  0    CALL PMIPL1(READ,EXTDSCT,FHCW,PART_RECORD,RBN); /* BDAM RD BY RBN */

 186   2  0    SELECT(FHCW1);                  /* CHECK READ RETURN CODE */
 187   2  1      WHEN('0');                    /* OK, DO NOTHING */
 188   2  1      WHEN('1')                     /* I/O ERROR */
                   DO;
 189   2  2          ERROR_FLAG = 4;
 190   2  2          RETURN;
 191   2  2        END;
 192   2  1      WHEN('2')                     /* RECORD NOT FOUND */
                   DO;
 193   2  2          ERROR_FLAG = 5;
 194   2  2          RETURN;
 195   2  2        END;
 196   2  1      WHEN('9')                     /* SELECT FAILED */
                   DO;
 197   2  2          ERROR_FLAG = 3;
 198   2  2          RETURN;
 199   2  2        END;
 200   2  1      OTHERWISE;
 201   2  1      END;
 202   2  0    IF STRING(P_REC_PIN) ¬= STRING(PARTNO) /* RECORD PART=GIVEN PART? */
                 THEN                                  /* NO, PART NOT FOUND */
                   DO;
 203   2  1          ERROR_FLAG = 5;
 204   2  1          RETURN;
 205   2  1        END;
 206   2  0    PRTDATA = P_REC_DES;          /* PART DESCRIPTION TO I/O MAP */
 207   2  0    ORDUNT = P_REC_UNT;           /* UNITS TO I/O MAP */
 208   2  0    PRTPRC = P_REC_PRC;           /* PART PRICE TO I/O MAP */
 209   2  0    END BDAM_READ;
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 12 of 15)



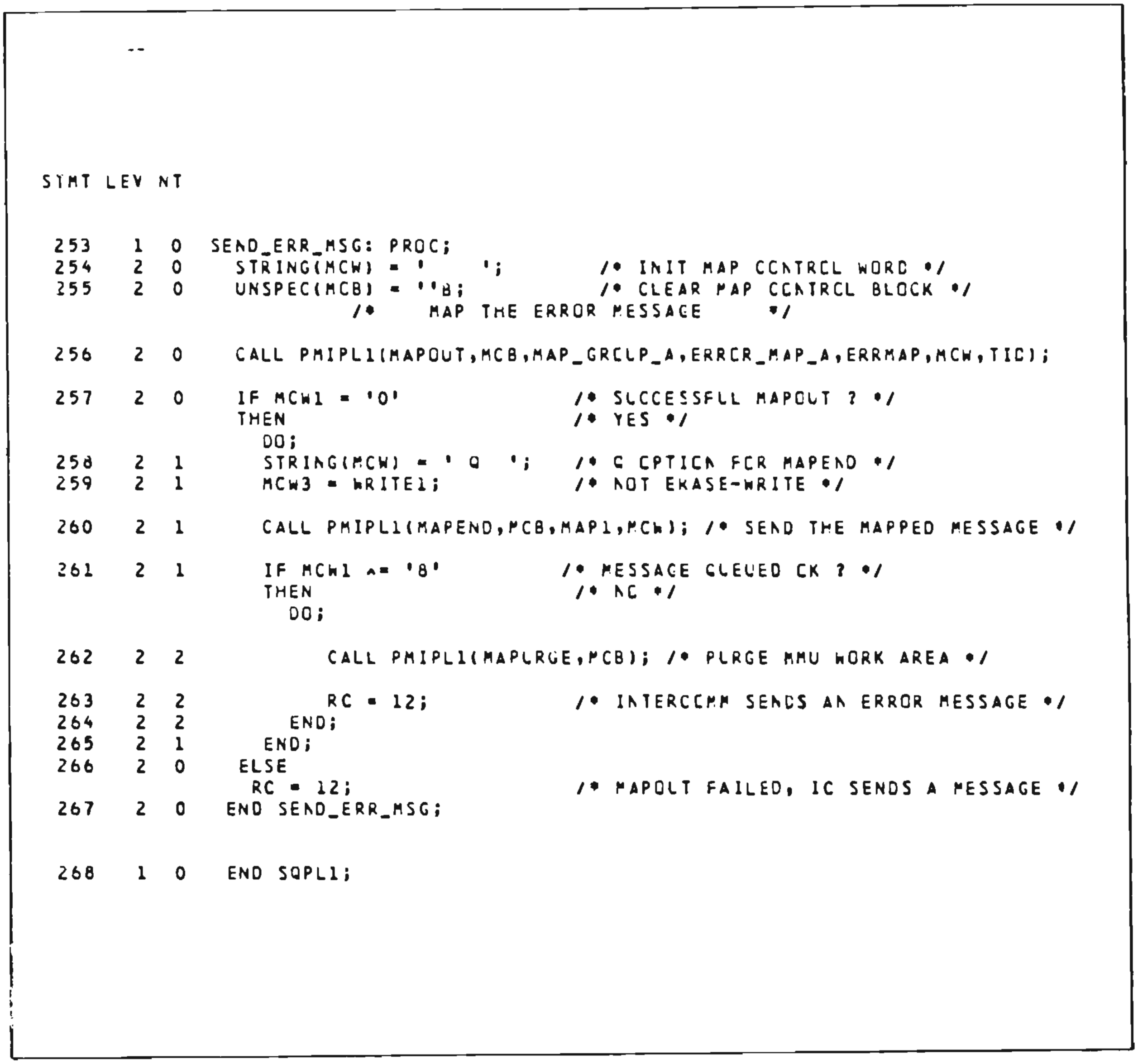
```
STMT LEV NT


 210   1  0  VSAM_READ: PRCC;                /* READ VSAM FILE BY KEY */
 211   2  0    UNSPEC(EXTDSCT) = ''B;        /* INIT EXTCSCT */
 212   2  0    STRING(FHCW) = '    ';        /* INIT FFCW */
 213   2  0    CURRENT_FILE = DD_STOCK;      /* FILE TO BE SELECTED */
 214   2  0    S_REC_WHS = WHSNO;            /* WHSNG IS PART OF THE KEY */
 215   2  0    STRING(S_REC_PNO) = STRING(PARTNC); /* PARTNC IS PART CF THE KEY */
 216   2  0    KEY_FIELD = STRING(S_REC_KEY_FIELD); /* THE VSAM KEY */

 217   2  0    CALL PMIPL1(SELECT,EXTCSCT,FFCW,CLRRENT_FILE); /* SELECT VSAM FILE */

 218   2  0    IF FHCW1 = '9'                /* SELECT FAIL ? */
                 THEN                        /* YES */
                   DO;
 219   2  1           ERROR_FLAG = 3;
 220   2  1           RETURN;
 221   2  1        END;
 222   2  0    STRING(FHCW) = '    ';        /* INIT FFCW FCR READ */

 223   2  0    CALL PMIPL1(GETV,EXTDSCT,FHCW,STOCK_RECCRC,KEY_FIELD);/* RD BY KEY */

 224   2  0    SELECT(FHCW1);                /* CHECK CETV RETLRN COCE */
 225   2  1      WHEN('1')                   /* I/O ERROR */
                   DO;
 226   2  2           ERRCR_FLAG = 4;
 227   2  2           RETURN;
 228   2  2        END;
 229   2  1      WHEN('2')                   /* RECCRD NOT FCUND */
                   DO;
 230   2  2           ERROR_FLAG = 5;
 231   2  2           RETURN;
 232   2  2        END;
 233   2  1      WHEN('9')                   /* INVALIC REQLEST */
                   DO;
 234   2  2           ERRCR_FLAG = 3;
 235   2  2           RETURN;
 236   2  2        END;
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 13 of 15)



```
STMT LEV NT


 237   2  1       WHEN('0')                     /* SUCCESSFUL ACCESS */
                    DO;                         /* RECORD FIELDS TO I/O MAP */
 238   2  2           WHSLOC = S_REC_WLC;
 239   2  2           STKLEV = S_REC_LEV;
 240   2  2           MONTH = SUBSTR((S_REC_LDT),1,2);
 241   2  2           DAY = SUBSTR((S_REC_LDT),3,2);
 242   2  2           YEAR = SUBSTR((S_REC_LDT),5,2);
 243   2  2           SLASH1, SLASH2 = '/';
 244   2  2           LEVDATE = STRING(DATE);
 245   2  2           STKORD = S_REC_ORD;
 246   2  2           MONTH = SUBSTR((S_REC_CDT),1,2);
 247   2  2           DAY = SUBSTR((S_REC_CDT),3,2);
 248   2  2           YEAR = SUBSTR((S_REC_CDT),5,2);
 249   2  2           ORDDATE = STRING(DATE);
 250   2  2         END;
 251   2  1       END;                          /* END OF GETV PROCESSING */
 252   2  0       END VSAM_READ;
```

Figure D-1.   Sample PL/1 Program Calling PMIPL1 (Page 14 of 15)



```
      --




STMT LEV NT


 253   1  0  SEND_ERR_MSG: PROC;
 254   2  0    STRING(MCW) = '     ';        /* INIT MAP CONTROL WORD */
 255   2  0    UNSPEC(MCB) = ''B;            /* CLEAR MAP CONTROL BLOCK */
                         /*    MAP THE ERROR MESSAGE      */

 256   2  0    CALL PMIPL1(MAPOUT,MCB,MAP_GROUP_A,ERROR_MAP_A,ERRMAP,MCW,TIO);

 257   2  0    IF MCW1 = '0'                /* SUCCESSFUL MAPOUT ? */
               THEN                         /* YES */
                 DO;
 258   2  1      STRING(MCW) = ' Q  ';      /* Q OPTION FOR MAPEND */
 259   2  1      MCW3 = WRITE1;             /* NOT ERASE-WRITE */

 260   2  1      CALL PMIPL1(MAPEND,MCB,MAP1,MCW); /* SEND THE MAPPED MESSAGE */

 261   2  1      IF MCW1 ¬= '8'            /* MESSAGE QUEUED OK ? */
                 THEN                       /* NO */
                   DO;

 262   2  2          CALL PMIPL1(MAPURGE,MCB); /* PURGE MMU WORK AREA */

 263   2  2          RC = 12;               /* INTERCOMM SENDS AN ERROR MESSAGE */
 264   2  2        END;
 265   2  1      END;
 266   2  0    ELSE
                RC = 12;                    /* MAPOUT FAILED, IC SENDS A MESSAGE */
 267   2  0   END SEND_ERR_MSG;


 268   1  0   END SQPL1;
```

Figure D-1. Sample PL/1 Program Calling PMIPL1 (Page 15 of 15)

APPENDIX E

SAMPLE PL/1 SUBROUTINE INTERFACE PROGRAM

E.1 INTRODUCTION

The routine listed below can be used to interface a PL/1 program to a user subroutine when either is dynamically loadable. Declare it in the calling PL/1 program as ENTRY EXTERNAL for calling PL/1 subroutines. The called subroutine must be defined to the REENTSBS table via a SUBMODS macro coded with the LNAME parameter in USRSUBS. On the PL/1 program call to BINTFAC, the first parameter must be the label of the 8-character (low-order blank padding, if needed) name (same as for LNAME parameter) of the desired subroutine (name in DSA if caller is loaded), the other parms (if any) are passed on to the subroutine. To use this routine, include it in the Intercomm linkedit for resident callers, link it with caller when calling program is loaded. Note that this routine preserves the PL/1 environment for the called subroutine by passing the caller's registers (except 0, 1 and 15). Return from the subroutine is directly to the PL/1 caller (via the 31-Amode interface when needed).

```
BINTFAC  TITLE 'BINTFAC - PL/1 SUBROUTINE INTERFACE'
BINTFAC2 CSECT
         ENTRY BINTFAC
         DC    C'BINTFAC',AL1(7)     ID FOR PL/1
BINTFAC  DS    0H
         REGS
         SAVE  (14,12),,*           SAVE CALLER'S REGISTERS
         LR    R2,R15               ESTABLISH BASE REGISTER
         USING BINTFAC,R2
         SLR   R15,R15              CLEAR
         ICM   R15,7,1(R1)          LOAD NAME ADDRESS
         TM    0(R15),X'C0'         POINTING TO ALPHA CHARACTER?
         BNZ   NAMEADOK             YES - BINTFAC DCL OPTIONS(ASM)
         L     R15,0(,R15)          LOAD NAME ADDRESS FROM LOCATOR
NAMEADOK DS    0H
         LR    R0,R15               PUT NAME ADDR IN R0 FOR MODCNTRL
         TM    0(R1),X'80'          ONLY ONE PARM PASSED?
         BZ    MOREPARM             NO
         SLR   R1,R1                CLEAR - NO OTHER PARMS
         B     PARMSOK
MOREPARM DS    0H
         LA    R1,4(,R1)            BUMP PARM LIST POINTER
PARMSOK  DS    0H
         MODCNTRL  ACTION=LINK,MODNAME=(0)  SET UP FOR CALL
         ORG   *-2                  OVERLAY BALR INSTRUCTION
         L     R14,12(,R13)         RELOAD CALLER'S RETURN ADDRESS
         LM    R2,R12,28(R13)       RELOAD REST OF CALLER'S REGS
         BR    R15                  NOW GO TO DYNLLOAD
         LTORG
         END
```

Figure E-1. Sample PL/1 Subroutine Interface

INDEX