SH20-9002-2

Customer Information Control System/Virtual Storage (CICS/VS) System/Application Design Guide

Program Product

Program Numbers 5740-XX1 (CICS/OS/VS)
5746-XX3 (CICS/DOS/VS)

IBM

Third Edition (July 1975)

This is a reprint of SH20-9002-1 incorporating changes released in the following Technical Newsletter:

SN20-9089 (dated May 22, 1975)

This edition applies to Version 1, Modification Level 1, Release 1, of the program product Customer Information Control System/Virtual Storage (CICS/VS), program numbers 5740-XX1 and 5746-XX3, and to all subsequent versions and modifications until otherwise indicated in new editions or technical newsletters.

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using this publication, consult the latest IBM System/360 and System/370 Bibliography, GA22-6822, and the technical newsletters that amend the bibliography, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

A form for readers' comments has been provided at the back of this publication. If the form has been removed, address comments to IBM United Kingdom Laboratories Ltd., Publications Department, Hursley Park, Winchester, Hampshire, SO21 2JN, England. Comments become the property of IBM.

PREFACE

This publication provides the system analyst and system administrator with guidelines which assist in the design of online applications to run under the control of CICS/DOS/VS or CICS/OS/VS, hereafter referred to as CICS/VS or simply CICS. It assumes that the reader is familar with the _CICS/VS General Information Manual_ (GIM). The GIM provides an introduction to the CICS/VS facilities, using several application examples to highlight the various facilities which these applications demand of CICS/VS. These applications are then developed further in this publication.

The publication is directed mainly towards the inexperienced CICS user, and assumes no prior CICS knowledge apart fram that presented in the _CICS/VS General Information Manual_.

It presents separate chapters, covering the following design topics:

- Introduction to Systems Design
- Program Design
- Data Communication Design
- Data Management Design
- Data Base Design
- Advanced Features
- Performance Considerations
- Recovery and Restart
- Testing and Integration
- Cutover and Follow-up
- Application Design

Each chapter is presented in a tutorial fashion, generally first with an outline of various CICS/VS facilities relevant to that chapter, followed by specific design techniques utilizing those facilities.

To enable experienced CICS users to concentrate only on these CICS/VS facilities which differ from previous versions of CICS, each chapter commences with a reading guide identifying only those topics which may need to be read by experienced users.

To enable the publication to be subsequently used for reference purposes, various CICS/VS facilities relevant to several areas of design are identified in those areas. However, whenever a CICS/VS facility is discussed, cross-reference is made to the section of the publication which describes that facility in more detail.

RELATED PUBLICATIONS

VTAM

Introduction to VTAM	GC27-6987

VSAM

DOS/VS Data Management Guide	GC33-5372
OS/VS VSAM Planning Guide	GC26-3799

CICS/VS

Advanced Communication Guide	SH20-9049
Application Programmer's Reference Manual	SH20-9003
System Programmer's Reference Manual	SH20-9004
Terminal Operator's Guide	SH20-9005
System Administrator's Guide	SH20-9006
Operations Guide (CICS/DOS/VS)	SH20-9012
Operations Guide (CICS/OS/VS)	SH20-9011
Subset User's Guide (CICS/DOS/VS)	SH12-5404
Reference Summary: Master Terminal Operator	SX26-3700
Reference Summary: Program Debugging	SX26-3701

DL/I DOS/VS

System/Application Design Guide	SH12-5413
General Information Manual	GH20-1246
Application Programming Reference Manual	SH12-5700
DL/I Bridge General Information Manual	GH12-5106

IMS/VS

System/Application Design Guide	SH20-9025
General Information Manual	GH20-1260
System Programming Reference Manual	SH20-9027
Application Programming Reference Manual	SH20-9026
Utilities Reference Manual	SH20-9029

DMS II

General Information Manual	GH20-1251

Video/370

General Information Manual	SC27-6960

CICS Productivity Aids

Installed User Programs

CICS Online Test/Debug Program Description Operations Manual	SH20-1258
CICS/COBOL Call Interface Program Description Operations Manual	SH20-1359

Field Developed Programs

CICS Dynamic Map Program Description Operations Manual	SB21-1075
CICS Performance Analyzer Program Description Operations Manual	SB21-1181
CICS/3270 Simulator Program Description Operations Manual	SB21-1036

CONTENTS

CHAPTER 1. INTRCDUCTION TC CNLINE SYSTEMS DESIGN

This chapter presents an cverview of CICS/VS system design, and introduces informaticn ccvered in more detail in later chapters of this publication.

SYSTEM DESIGN IN THE IMPLEMENTATION PHASE

The installation of an cnline system involves a number of activities. These include, but are nct limited to:

- Feasibility study
- System design of online applications
- Application prcgramming
- Program testing
- Documentation
- System testing
- Training
- Installation of equipment
- Cutover to online application
- Follcwup of system

The purpose of this publicaticn is to discuss only one activity of those detailed above, namely, Systems Design of Cnline Applications, because of the effect of system design on the overall success cr failure of the installation. Online system design is presented in the same sequence as would be covered in real-life. The varicus factors to consider during each step cf the design process are identified in terms of the application requirements. Some design factcrs and application requirements are satisfied by CICS/VS-provided support. These facilities are explicitly defined and identified.

Many applications will nct require additional user-developed support beyond that provided by CICS/VS. However, online applications may exhibit unique requirements, for example, high online system availability beyond the standard provided by CICS/VS. This publication presents these additional support requirements, cutlines suggested design solutions, and discusses some of the potential prcblem areas that should be ccnsidered by the user.

To utilize CICS/VS facilities efficiently and satisfy various design requirements, it is important that the system designer be aware of the manner in which CICS/VS implements these facilities. This information is presented at the conceptual level, assuming there is no prior knowledge beyond that ccvered in the CICS/VS General Information Manual. More detail can also be obtained, if necessary, by referring tc other CICS/VS documentation.

THE NEED FOR GOOD SYSTEM DESIGN

The design of any system, whether it be a batch processing system or an online system, is a complex and involved procedure. A "cookbook" approach to system design cannot be followed because of the variety of ways the same application may be implemented in different organizations. However, guidelines can be recommended which direct the designer to consider those functions or requirements which exist in the design of most online systems.

TURNAROUND OR RESPONSE TIME

The effect of poor system design in a batch processing environment increases the total processing time of applications, with consequent delays in turnaround time before results of that processing are available. With an infrequently run batch application, the effect of poor system design on the installation may not be great. However, with frequently run batch processing applications, poor system design and long run times may impact the ability of the installation to provide adequate turnaround for that and other applications. This will probably necessitate a change in the system design of the offending application.

In an online environment, the effect of poor system design is often immediately apparent, generally through the online system providing unacceptable response times for the particular applications concerned. The definition of an "acceptable response time" is generally very application-dependent. For example, in an online order entry application, where the terminal operator takes an order from a customer directly over the telephone, any response time that keeps that customer waiting unnecessarily can be regarded as unacceptable.

WITHOUT DOUBT, THE SINGLE MOST IMPORTANT FACTOR IN ONLINE PERFORMANCE IS THE SYSTEM DESIGN OF THE ONLINE APPLICATIONS.

USER ACCEPTANCE

A factor that can affect the acceptability of an online application is the way in which it meets the needs of the users of that application. It is pointless for the user to design a system that provides fast response time if the information provided cannot be used. In this regard, measured by the usability of the system, an unusable system is therefore a "poor performance" system.

RESOURCE UTILIZATION

A final factor to consider is the utilization of resources such as the CPU processing capability, CPU storage, and input/output devices. An online system which unnecessarily uses so much CPU processing capability, or storage, or so many input/output devices that it impacts the ability of the installation to carry out other processing in other partitions or regions may be a "poor performance" system.

Thus, poor system design can have a significant impact on:

- Customer service (because of poor response time)
- Application usability
- Installation processing capability

Once the system designer realizes that poor design can result in the opposite of the desired objectives being met, he is well on the way to producing a well designed system.

DESIGN STRATEGY

Generally, online systems cannot be designed in isolation. To ensure that the foregoing objectives are met, it is important that a design group comprise people with knowledge of:

- Application requirements
- CICS/VS facilities
- Installation requirements

Usually, the optimum size for the design group is three or four. Fewer than this number increases the probability that bad design decisions can slip through, while many more than four may affect the productivity of the design group as a whole.

The system design phase is an iterative process. Based on the decisions taken at one stage of the design, it may be necessary to change decisions which were made earlier in another area of the design. This change may in turn affect other decisions. Thus, the design group must be flexible in its approach and be prepared during the design phase to change its decisions if necessary. However, once the system design has been completed, it should be frozen at that point, and not changed unless serious errors or omissions are found which will affect the ability of the system to run effectively.

During implementation of the design, there is always the temptation to incorporate improvements from an application point of view. While each improvement may not represent a great deal of extra implementation effort, all of these improvements may affect the project completion date. Also, the effect of these improvements on the overall system performance must be evaluated. The danger is that this evaluation may not be carried out for those changes introduced after the system design phase has been completed.

These changes or enhancements must be controlled. The best way of achieving this control may be to incorporate all of these enhancements in a later version of the online application or system. These enhancements become a project in their own right, and must therefore go through the system design phase before implementation. In this way their effect on system performance can be readily evaluated.

A structured approach to system design is possible, and such a structured approach should direct the design group to consider all of those areas of the online system which may require decisions to be taken. This structured design approach is illustrated in Figure 1-1. This figure also illustrates some of the topics presented in this publication, and the description of each topic following the figure provides an overview of this publication.

STRUCTURED SYSTEM DESIGN

Application Design

The starting point for online system design is the application design. The initial application design steps require that the objectives to be achieved by an online application be defined and the requirements of the users of that application be identified. A broad system flow of the application is then developed as part of the initial design. This system flow and application design are an extremely important part of the overall design process, since they define the interface between the terminal user and the computer. Unless the online application meets the requirements of its users, it is destined to fail.

The online application should be designed initially to identify the broad input, processing, and output requirements of the application. The need for conversational and/or batch data transmission between the terminals and the CPU can be identified. The terminal output requirements of the application can be determined, after which the broad processing logic and data set accessing necessary to produce that output can be designed. At this stage, the input data required for that processing and output can also be defined.

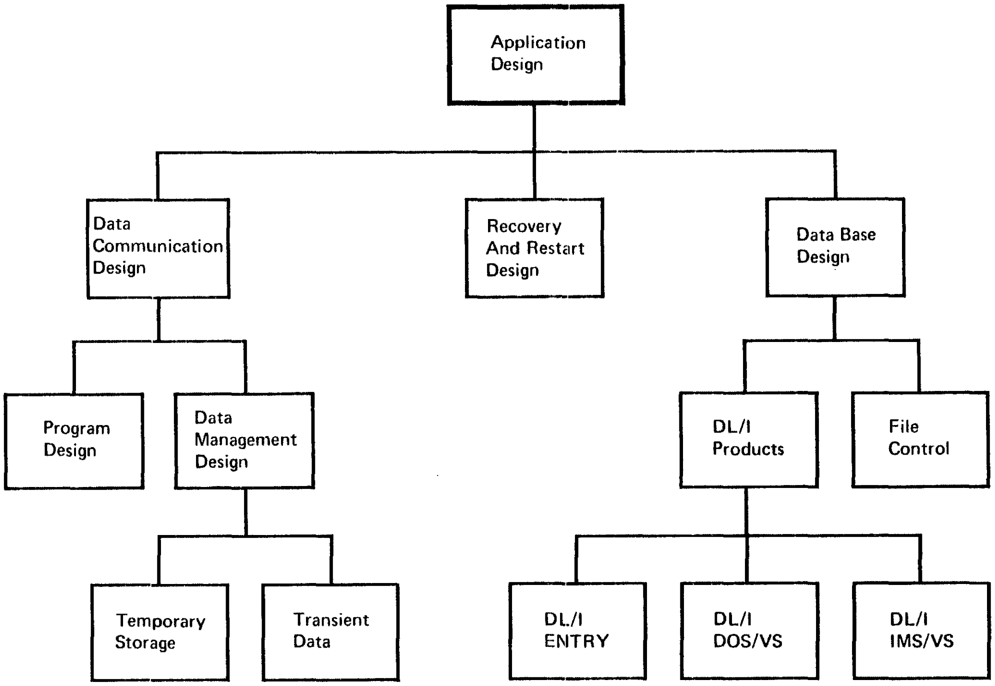


Figure 1-1. Structured Systems Design

The result of this application design phase is a broad system flowchart showing, in application terms only, the flow of information to and from terminals, the broad processing to be carried out by the CPU, and the file accessing necessary to allow that processing. Figures 1-2 and 1-3 illustrate two types of flowcharts, both representing the system flow of an Order Entry and Invoicing application in the Distribution industry.

Data Communication Design

With the broad application design mapped out, design of transactions to be initiated from terminals and the responses to be sent back to the terminals can be developed. Also, during this phase the editing and validation of input messages can be defined in more detail.

Consideration should be given to the design of security procedures and the handling of high priority transactions. The effect of unrecoverable terminal and line errors should be considered, together with approaches which may be used to provide a communications backup capability (if required) to enable the online applications to continue to function, if possible, in the event of a communications equipment malfunction.

Program Design

After determining system flow and broad processing to be carried out by the CPU, this processing should now be broken down into particular functions. For example, the initial function on receiving a terminal transaction would be that of editing or validation.

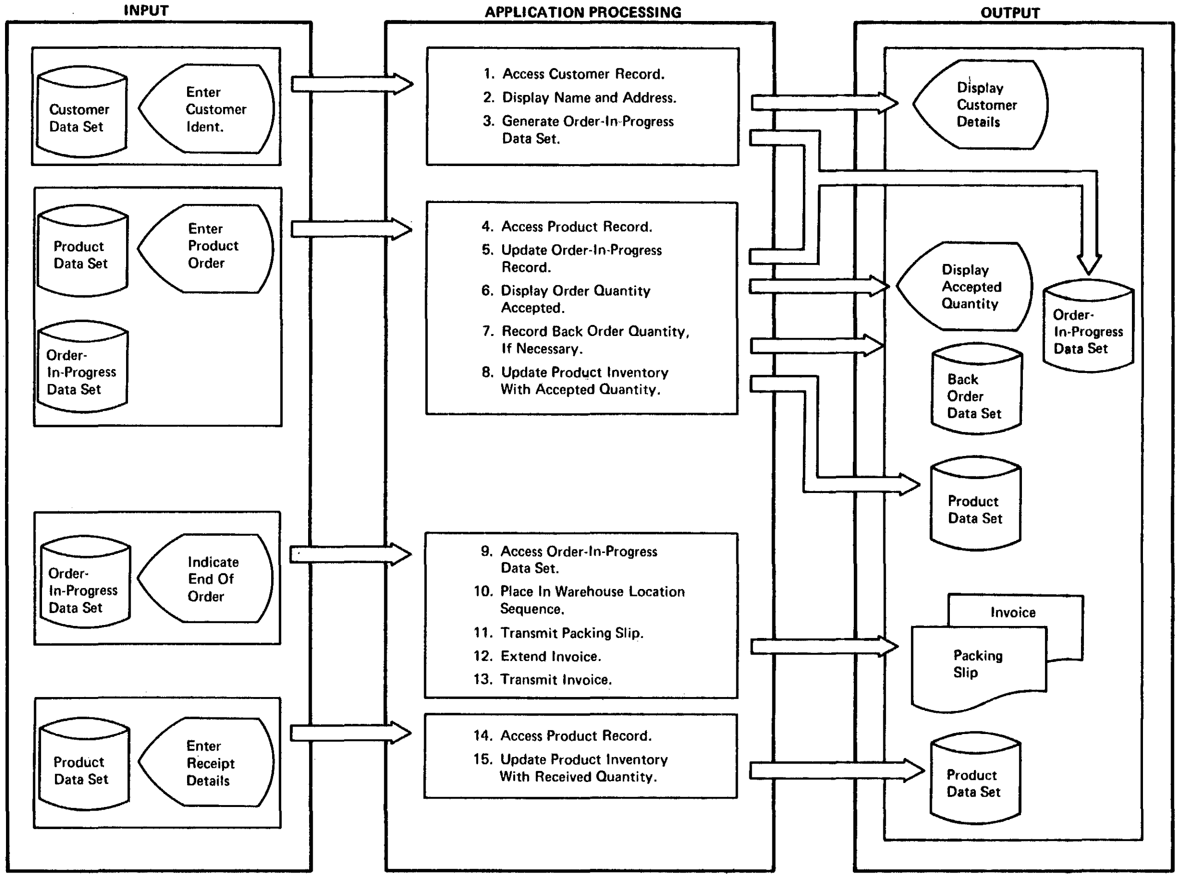

INPUT
APPLICATION PROCESSING
OUTPUT
Customer Data Set
Enter Customer Ident.
Product Data Set
Enter Product Order
Order-In-Progress Data Set
Order-In-Progress Data Set
Indicate End Of Order
Product Data Set
Enter Receipt Details
1. Access Customer Record.
2. Display Name and Address.
3. Generate Order-In-Progress Data Set.
4. Access Product Record.
5. Update Order-In-Progress Record.
6. Display Order Quantity Accepted.
7. Record Back Order Quantity, If Necessary.
8. Update Product Inventory With Accepted Quantity.
9. Access Order-In-Progress Data Set.
10. Place In Warehouse Location Sequence.
11. Transmit Packing Slip.
12. Extend Invoice.
13. Transmit Invoice.
14. Access Product Record.
15. Update Product Inventory With Received Quantity.
Display Customer Details
Display Accepted Quantity
Order-In-Progress Data Set
Back Order Data Set
Product Data Set
Invoice
Packing Slip
Product Data Set

Figure 1-2. Order Entry and Invoicing Function Diagram

This validation may require access tc various data sets. Follcwing validation, it may be necessary to retrieve information from other data sets fcr processing, follcwed by pcssible updating of those data sets. Finally, it would te necessary tc prepare a respcnse to te sent to the terminal.

The processing for each type cf transaction in the application should te broken down intc logical secticns in this manner. These logical sections may subsequently tecome separate CICS/VS application program modules, or can te incorpcrated into one module. Figure 1-4 illustrates the various modules in the program design for the Order Entry and Invoicing applicaticn shcwn in Figures 1-2 and 1-3.

Note that the separate prcgrams and broad processing required, developed in Figure 1-4 from the flcwchart in Figure 1-3, are described as part of the function diagram in Figure 1-2. In effect, the first three boxes in Figure 1-2 define the three separate programs in Figure 1-4.

A point to consider when defining program modules is the frequency of use of different modules. For example, excepticn routines cr error routines which are infrequently used should te separated from the more frequently used main processing modules. In this way program design and subsequent implementatior will be able tc take best advantage of the dynamic storage capatilities of CICS/VS and the virtual storage capatilities of DOS/VS, CS/VS1, or CS/VS2.

Application programs car te ccded in Assembler Language, ANS COBOL, or PL/I. The user can select the most appropriate language for each program. Prcgrams written in one language can pass control to programs written in any other language.

Data Management Design

Application requirements for the temporary stcrage of information and the queuing cf informaticn should te defined. CICS/VS Temporary Storage management provides a "scratchpad" capability and allows information to te stored temporarily in main storage or, alternatively, on secondary storage.

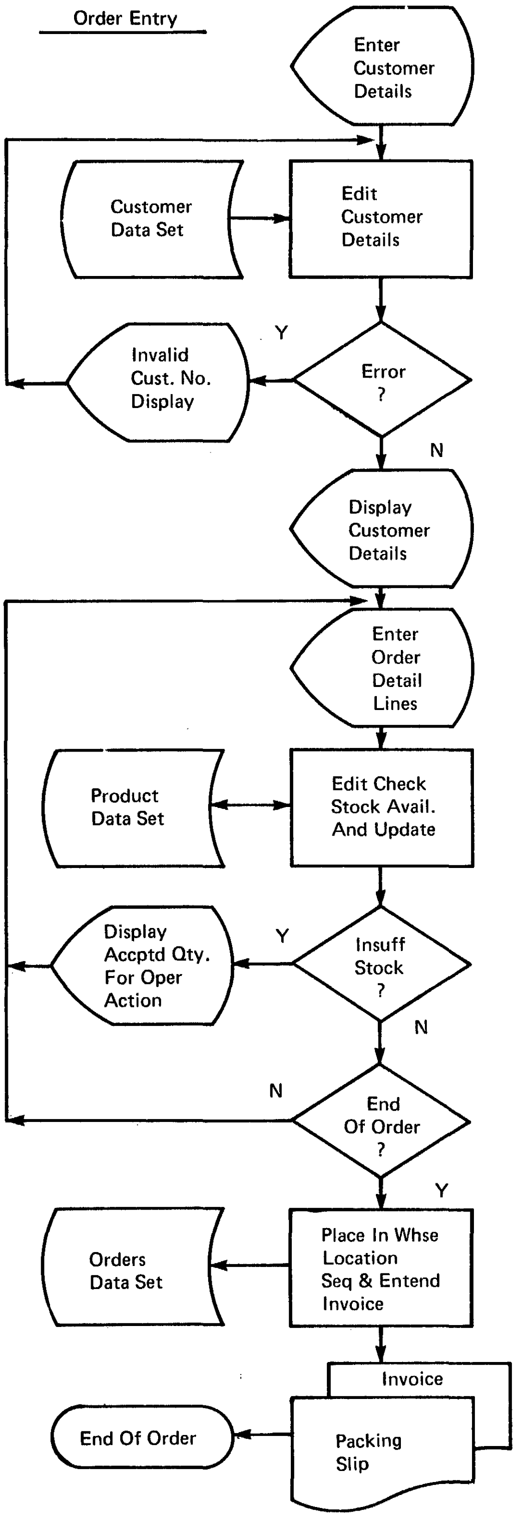
The queuing, or sequential data set requirements, of the application can te defined. The need to pass information thrcugh sequential files to and from the CICS/VS partition and cther batch partitions or regions using the CICS/VS Transient Data management facility can also te determined, together with trcad recovery procedures.

The need for the applicaticn prcgrams to pass small sequential queues of information tetween each cther in the CICS/VS partition using CICS/VS Transient Data can be determined.

Data Base Design

Particular application data base characteristics and requirements are considered when selecting the best data tase support. This can be based on CICS/VS File Control facilities or on one of the DL/I products.

SYSTEM FLOW



DESCRIPTION

Order Entry

Enter customer number and customer reference number.

Validate customer number and extract credit limit.

If an error is found, display error message back at terminal.

Display customer name, address, ship-to-address and credit limit.

Enter product number and quantity for each line item.

Validate product number against product data set. Determine current stock availability, and update product data set.

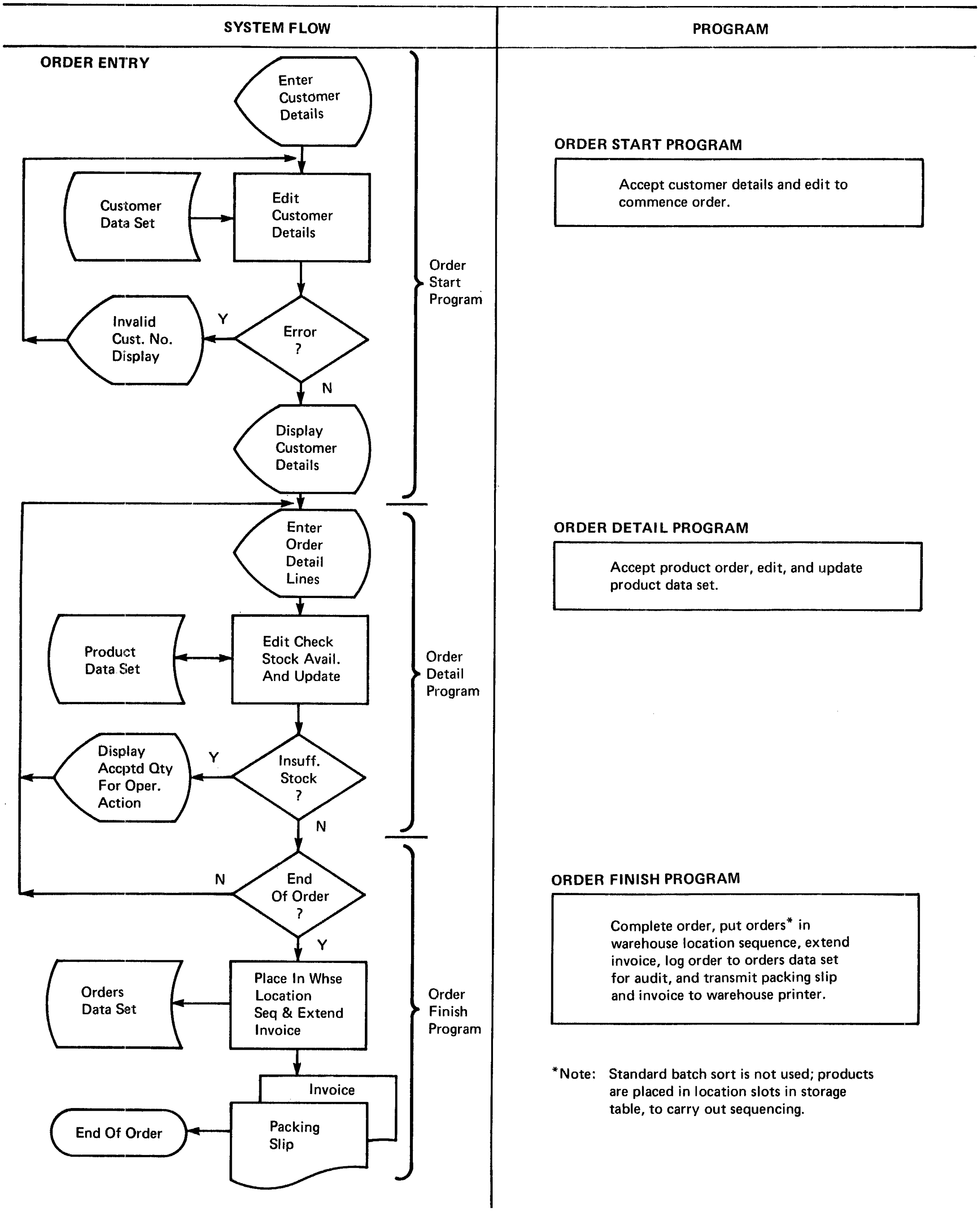
If insufficient stock, indicate quantity on-hand. Then allow operator to either order available quantity, cancel item, or cancel order.

If not end of order, read next line item from order terminal.

Sequence products in order to warehouse loacation sequence. Extend invoice. Write order to orders data set.

Transmit packing slip ans invoice to terminal in warehouse.

Figure 1-3. Order Entry and Invcicing Flowchart



ORDER START PROGRAM

Accept customer details and edit to commence order.

ORDER DETAIL PROGRAM

Accept product order, edit, and update product data set.

ORDER FINISH PROGRAM

Complete order, put orders* in warehouse location sequence, extend invoice, log order to orders data set for audit, and transmit packing slip and invoice to warehouse printer.

*Note: Standard batch sort is not used; products are placed in location slots in storage table, to carry out sequencing.

Figure 1-4. Order Entry ard Invcicing Program Design

Factors to be considered in this decision include the need to access the data base from both online application programs and batch processing programs, and the number of ways in which information is to be retrieved, such as by the use of different record keys (for example, part number or part name in an inventory control application). Further factors in this decision are the number of times certain information occurs in each record, and the amount of information which may be absent in some records, yet present in others.

After selecting the appropriate data base support, the structure of the data base is designed, and how that data can be retrieved from application programs is defined. Figure 1-5 shows the design of a DL/I logical structure for the Order Entry application discussed above. (This logical structure is discussed in "Distribution Industry" in Chapter 11.)

The effect of various error or system failure situations on the integrity of the data base is considered, and a data base recovery and backup approach (if required) is defined.

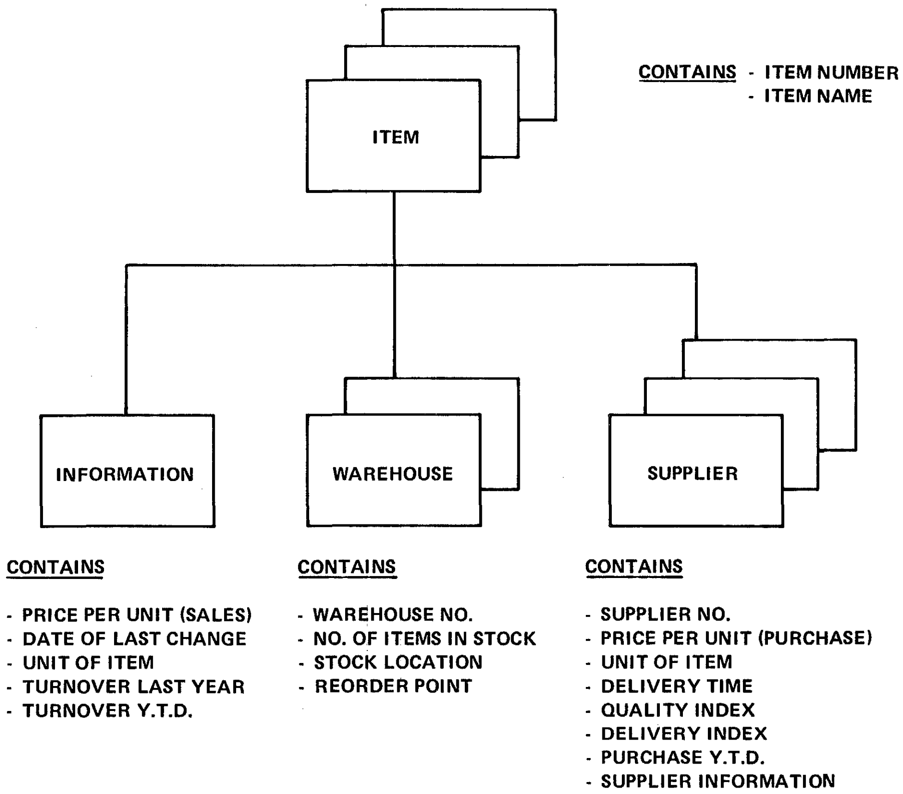


Figure 1-5. Order Entry Application Data Base Design

Performance Considerations

The designed system is starting to take shape. The extent to which the application objectives and requirements are met by the designed system must be evaluated. The potential performance of the system must be evaluated to identify areas where improvement can be made if necessary.

Performance evaluation may include one, or both, of:

- Simulation techniques
- Benchmark techniques

Based on this performance evaluaticn, changes in the system design may be considered, with possible iteration through the above steps.

Recovery and Restart

The success of an online system is dependent cn its availability. Procedures must te designed fcr tackup in the event of failure of varicus components of the system, and for recovery and restart following abnormal termination of the system.

Testing and Integration

The amount and type of testing tc be carried out should be broadly defined as part of system design, together with the way in which the varicus online applicaticns are tc be integrated.

Producticn Cutover and Followup

It is important to define the procedures to be followed fcr training of terminal operators, system administrators, and all personnel involved in the cutover and subsequent operation of the system. The procedures to be used for cutover must be fully defined to ensure smooth transition to the new online system.

APPLICATION DESIGN

The logical starting point for online system design is the broad system flow design of the applications to be implemented. In the normal design phase of an online system, the design team would commence with application design. However, in the presentation of topics in this publication, application design will be left until the various CICS/VS facilities and design techniques have been discussed. In this way, application design in a number of industries can te described more effectively, indicating hcw specific CICS/VS facilities can be utilized for different applications. Using these application design guidelines, the design team may wish tc use the various techniques described as a starting point for their own applications. The applications discussed in Chapter 11 of this publication are those introduced in the "Management Overview" section of the *CICS/VS General Information Manual* (GIM). The applications described in the GIM are:

- *Manufacturing Industry*
  - Production Order and Status Reporting System
- *Banking Industry*
  - Savings Bank and Mortgage Loan System
  - Customer Information System (often called Customer Information File)
- *Insurance Industry*
  - Policy Information System
  - New-Business Policy Entry System

- *Medical Industry*
  - Patient Informaticn System

- *Pharmaceutical Industry*
  - Pharmaceutical Crder Entry System

- *Law Enforcement Industry*
  - Police Information System

- *Distribution Industry*
  - Order Entry and Invoicing System

- *Utilities Industry*
  - Customer Infcrmaticn System

The reader may wish to refer to these application descriptions from time to time as he reads this *CICS/VS System/Application Design Guide*.

Deferring the applicaticn design now until Chapter 11 in this publication, the next design topic is that of Prcgram Design.

CHAPTER 2. CICS/VS PROGRAM DESIGN

Chapter 2 presents Program Design in a tutorial manner. Experienced CICS users may wish to omit most of this chapter. However, it is strongly recommended that such users still read the following topics:

- CICS/DOS/VS Subset Option
- Virtual Storage Environment
- Tabular Structures
- Structured Programming
- Application Built-in Functions
- Program Error Recovery

CICS/VS is a transaction-oriented DB/DC system which uses the techniques of:

- Multitasking
- Quasi-reentrant programming
- Dynamic storage allocation

These techniques are described in the CICS/VS General Information Manual. The design of applicaticn programs to take advantage of them for efficient online operation will now be discussed. In this discussion, the facilities available to the system design team are outlined first, followed by a discussion of the various program services provided by CICS/VS. The design facilities available for use are:

- CICS/DOS/VS Subset Option
- Modular programming
- High-level languages
- Tabular structures
- Structured programming
- Application functions

CICS/DOS/VS SUBSET OPTION

Facilities are provided to generate a subset of CICS/DOS/VS for new CICS/DOS/VS users. It is easy to install and is fully compatible with the complete CICS/DOS/VS system. No changes need be made to application programs when the user generates a complete CICS/DOS/VS system. The subset option identifies CICS/VS facilities which may be utilized by a CICS/VS user with limited CPU Storage. See "CICS/DOS/VS Subset Option" in Chapter 7 for additional information.

CICS/DOS/VS STARTER SYSTEM

A set of object modules, generated using the subset option of CICS/DOS/VS, is supplied with the CICS/DOS/VS system. The starter system includes precompiled sample application programs and predefined tables, and need only be link-edited into a DOS/VS core image library before use.

INSTALLATION AND USE

The user can install the pregenerated CICS/DOS/VS system as described above, and expand it as his needs dictate. CICS/VS facilities which are not part of the subset option can be generated when required to support advanced CICS/VS capabilities. These can be achieved by regenerating only those CICS/VS management modules and table options required to support the advanced capabilities.

The subset option is described in the Subset User's Guide (DOS) SH12-5404.

MODULAR PROGRAMMING

BATCH ENVIRONMENT

Modular programming techniques in a batch environment may involve the consolidation of similar program functions in one program module. For example, the main execution code used may be incorporated in one module, while exception routines may be in another module and error routines in other modules. In this way, modular programming enables sections of the program to be written by programmers at different times. Apart from the advantage of distributing the program workload across several people, another advantage of modular programming is that it generally makes the application program logically easier to follow for someone who is unfamiliar with it.

CICS/VS ONLINE ENVIRONMENT

CICS/VS is oriented around the concept of modular programming. Transactions received from terminals are analogous to transaction cards read from a card reader. A transaction code defines the format and processing required for an online transaction, in the same manner as a card code defines the format and processing of a card.

This transaction code identifies the CICS/VS application program that will process the transaction. The use of such modular programming techniques is an integral part of CICS/VS and enables large programs to be broken into smaller logical modules. However, program size and CICS/VS address space availability should be balanced with the additional overheads involved in passing control between many small modules.

When a transaction is received from a terminal, only that program code relevant to the processing of that transaction need be loaded into storage, if it is not already present. As modules tend to be smaller than complete programs, more application program modules may reside in a given address space than may full programs. This enables one copy of each of many different modules to be currently resident in the CICS/VS dynamic storage area. A high degree of multitasking may therefore be achieved within a limited storage size.

VIRTUAL STORAGE ENVIRONMENT

Using the modular programming techniques discussed above, a CICS/VS application program module should include code which is relevant to the processing of the specific transaction.

From the system design point of view, the design team should specify the various application programs which are to be written to implement the particular application. They should also identify those application functions (and hence program coding) which will be frequently used by transactions, and those which will be infrequently used. In this way, the design team is able to broadly specify the modular program structure of the application, and define the necessary application programs.

The various application programs executing concurrently in the CICS/VS partition, and the demands made by them for CICS/VS services and resources, contribute to the total "working set" of CICS/VS. This term is used in a virtual storage environment to describe that part of a program which is active over a specific period of time. The CICS/VS working set is influenced by the sizes of the various concurrently executing application programs, the online transaction load and its use of various application programs, and the degree of multitasking permitted by the CICS/VS master terminal operator. Techniques for varying the working set are discussed in "CICS/VS Working Set" in Chapter 7.

HIGH-LEVEL LANGUAGES

CICS/VS accepts application programs written in Assembler Language, American National Standard (ANS) COBOL, or PL/I Optimizing Compiler for DOS/VS, OS/VS1, or OS/VS2. In addition, application programs may be compiled with the PL/I F Compiler for OS/VS1 or OS/VS2.

TABULAR STRUCTURES

CICS/VS is basically table driven. Tables define the terminal network configuration, data set and data base specifications, online transaction details, application programs, and output message destination information. Since CICS/VS is written as a generalized program and is table-oriented, the unique requirements of an installation can be tailored by specifying those requirements in the various CICS/VS tables. CICS/VS uses the tabular information in a direct manner to complete the particular functions required. In the event of the installation characteristics changing (such as the addition of more terminals, or extra data sets for example), that change in the installation requirements can be incorporated into the system by modifying and reassembling the relevant tables.

The tabular structure of CICS/VS is one of the main factors which enables fast implementation and easy installation growth--two of the significant advantages of CICS/VS.

This same tabular structure concept can be extended to application programs. An example of a tabular structure application is a savings bank and mortgage loan system in the banking industry. Figure 11-7 illustrates a typical savings bank and mortgage loan system.

This application is characterized by a large number of transaction types with similar transaction formats, similar processing, and similar output formats. Certainly, unique modules may be written to accept each different transaction, process that information, and send back an output response. However, the overall logic in the separate modules is basically identical, with differences appearing only in the particular input and output formats. In some cases certain information is processed by additicn (a deposit transaction amount to be added to the current balance), and in cther cases by subtraction (a withdrawal transaction amount to be subtracted frcm the current balance). It is expensive in the initial applicaticn programming and subsequent maintenance to write separate programs for these various similar transactions.

In this banking application, a generalized application program may be written utilizing a tabular structure. A number of application tables wculd be required in this environment. These are illustrated in Figure 2-1, and are listed and discussed below:

- Input format table
- Processing requirements table
- Output format table

Alternatively, the information described in these tables may be consolidated intc one ccmpcsite table.

On receipt of a transacticn from a terminal, that transaction type (Bank Trancode) may be identified in the input fcrmat table. Switches in this table specify the locaticn of information within the transaction. This informaticn is used together with information obtained from the prccessing requirements table which is also accessed based on the transaction type (Bank Trancode). Fcr that transaction type, the processing requirements table entry may indicate certain fields of the transacticn are to be edited based upon specific editing criteria, and fields are tc be added to or subtracted from specific applicaticn counters.

Based on the particular transaction type, the relevant entry in the output format table may specify the exact locaticn in the cutput message into which certain fields are to be inserted. Responses may then be sent back to the terminal to update the customer's bank passbook based upon the particular input transacticn entered.

Use of tabular structures results in less programming effort. Only one generalized application program is written, determining the editing and processing required cf transacticns by means of various switches in the relevant tables.

However, the power of this prcgram design approach becomes more apparent when it is necessary to mcdify the application requirements. Typically such application modification may require considerable recoding and testing if a tabular structure is not used. In this environment, the relevant table entry may be quickly and easily changed to reflect a changed input fcrmat, changed processing requirements, or a changed output format. In many cases, no modification of the generalized application program is required.

The net effect is greater responsiveness to the application needs of user departments, as well as the needs of the ccmpany's customers.

The IBM 3600 Finance Communication System, a banking system, consists of an IBM 3601 programmable controller and several terminals. Some of the functions perfcrmed by the previcusly described tables may be executed in the 3601 contrcller. For example, the terminal input message may be ccnverted tc a standard format input message by the 3601 controller for transmission to CICS/VS with processing requirement switches inccrporated in the input message. Similarly, a standard format output response may be transmitted by CICS/VS back to the controller, which performs any unique formatting required by the response and transmits it to the criginating terminal.

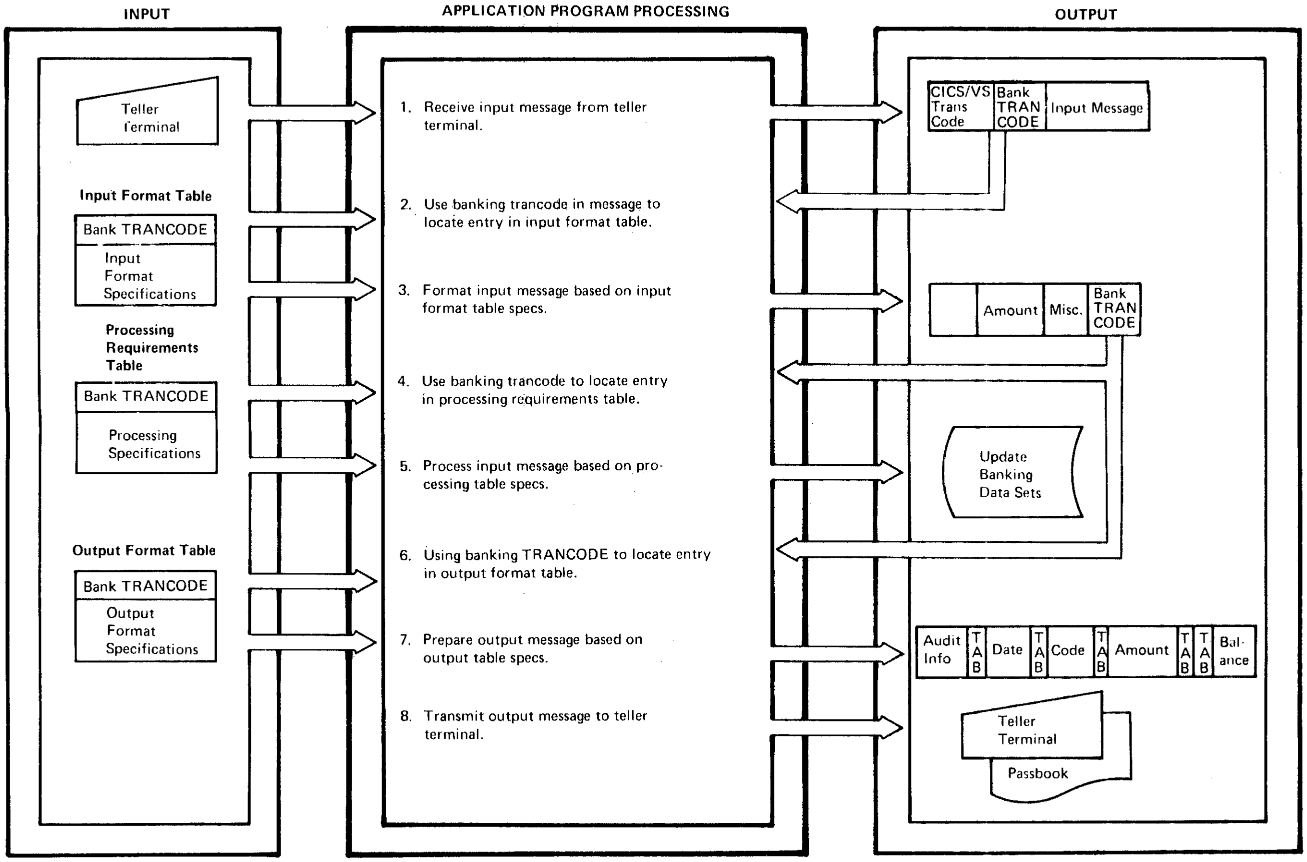

INPUT
APPLICATION PROGRAM PROCESSING
OUTPUT
Teller Terminal
Input Format Table
Bank TRANCODE
Input Format Specifications
Processing Requirements Table
Bank TRANCODE
Processing Specifications
Output Format Table
Bank TRANCODE
Output Format Specifications
1. Receive input message from teller terminal.
2. Use banking trancode in message to locate entry in input format table.
3. Format input message based on input format table specs.
4. Use banking trancode to locate entry in processing requirements table.
5. Process input message based on processing table specs.
6. Using banking TRANCODE to locate entry in output format table.
7. Prepare output message based on output table specs.
8. Transmit output message to teller terminal.
CICS/VS Trans Code
Bank TRAN CODE
Input Message
Amount
Misc.
Bank TRAN CODE
Update Banking Data Sets
Audit Info
TAB
Date
TAB
Code
TAB
Amount
TAB
TAB
Balance
Teller Terminal
Passbook

Figure 2-1. Tabular Program Structure in Banking

Using this apprcach, the tabular structure concept described in Figure 2-1 can also te applied tc the application programming performed in the 3601. The functicns cutlined in steps 1 through 4 and 6 through 8 in Figure 2-1 are then executed by the 3601 and only step 5 is executed by the CICS/VS application program. See "Virtual Telecommunications Access Method" in Chapter 3 fcr additional information regarding 3600/CICS/VS operation.

STRUCTURED PROGRAMMING

Structured programming is a modular programming technique which has been developed tc permit easier integration cf mcdules into a working program. It is sometimes referred to as "top-down programming," and provides a useful tocl fcr ccntrcl and development of large programming projects. The fcllowing remarks serve to introduce the concept of top-down programming.

TRADITIONAL PROGRAM DEVELOPMENT

Traditionally, some programs have been developed from the bottom up, as illustrated in Figure 2-2. That is to say, each routine or module has been designed and written, then these modules have been combined, or integrated, to produce a working program. Programs at the lowest level are combined by integrating them with a program at a higher level, which calls them.

Thus a large program is built up from separate modules, with the lowest level of modules combined first, and then the successively higher levels of modules until eventually the entire program, with all of its modules, has been integrated. If the system design has been well done, and all of the linkages and interfaces have been fully designed, documented, and completely adhered to by all programmers, a working program results.

However, as is often the case, each programmer's understanding of the way in which his modules fit into the total project may be slightly different. These differences are often reflected in errors in the module interfaces. These errors are not determined until integration or system test, and may involve considerable modification to enable the entire program to be built up.

A further problem that arises with the traditional "bottom up" development of modules is that of testing. To test a lower level module, a test driver invariably has to be developed. The function of this driver is to present to the module to be tested the same interface which will be presented by the higher level module which will eventually call that lower level module. Thus, the testing of these lower level modules can require considerable additional work on the part of the programmer in developing test vehicles.

In addition, in testing higher level modules, changes may have to be made to lower level modules because of errors identified or interface

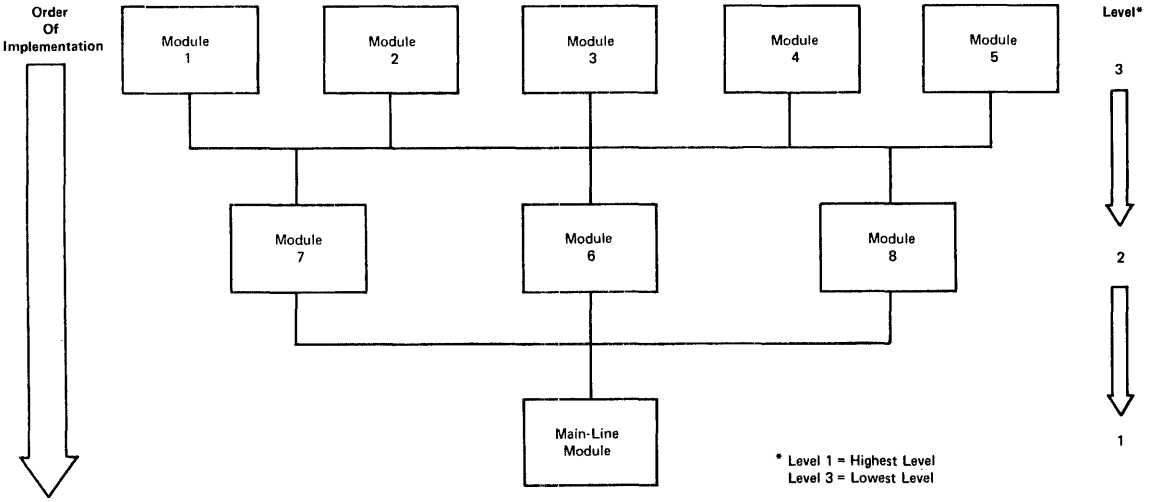


Figure 2-2. Traditional Program Development

changes. Consequently, the lower level changes must be fully tested before testing can continue with the higher level module.

TOP-DOWN PROGRAMMING

Top-down programming approaches the problem of program development in a different way. The highest level module is defined and coded first, including the necessary linkages to lower level modules. However, these lower level modules are not developed at this time. Instead, a general "dummy" test module is used in place of the lower level modules. The high level module links to this test module in place of lower level modules which are yet to be written. The dummy test module notes the fact that control was passed to this module from the higher level module (perhaps by a test output message, or a dump, for example), and then returns control to the higher level module.

It is not until the high level module has been tested that coding commences on the next lower modules. At this time, the interface between the higher level and lower level modules has been completely defined, coded, and tested. Furthermore, the higher level module now becomes a test driver for the lower level modules.

As each low level module is coded, it replaces the common dummy test module which was used in the higher module testing. When the higher module passes control to this low level test module, the only functions which have to be tested are the functions represented by that low level module.

This testing continues, with progressively lower level modules being integrated into the total program structure in this way, until the entire program has been developed and tested. Top-down programming is illustrated in Figure 2-3.

Advantages of Top-Down Programming

The advantages offered by this technique over traditional "bottom-up" programming are:

- Chief Programmer Operation

  Top-down programming lends itself to the chief programmer method of program development. This method involves an experienced (or "chief") programmer, who defines the overall logic flow of the program by developing the highest level modules, and leaving the lower level modules to less-experienced programmers. In this way, the controlling high-level modules benefit from the skill of the chief programmer, resulting in better overall control of the total program development to ensure that processing objectives and performance requirements are met.

- Module Development Independence

  Modules can be coded and tested from the highest level down to the lowest level without consideration for the progress of other modules at the same level. For example, developing a program from the bottom up generally requires all of the lower level modules to be coded and available before integration with the higher modules can be achieved.

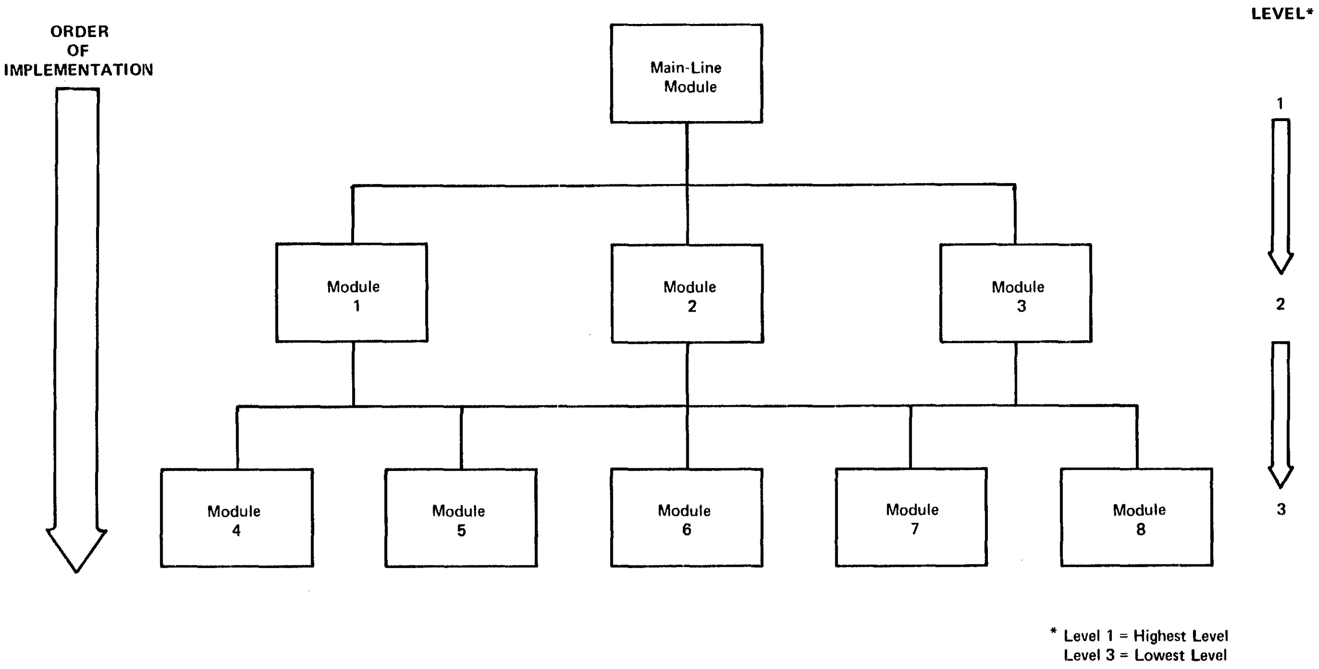


Figure 2-3. Tcp-Dcwn Prcgram Development

- Module Testing Independence

  Because higher level modules are coded and tested befcre lower level modules, the testing of these lower level modules is not dependent upon code which may not have been written at that time. The familiar problem cf the testing of the lower level module being held up because the higher level module which called it (or alternatively a test driver in its place) was not ready, is now not significant.

- Easier Evaluation of Testing Progress

  The progress of testing can be mcre readily evaluated using the top-down programming approach. As testing descends to lower level modules, the probability of errors detected in these lower level modules affecting already tested higher level modules is much less. However, with bottcm-up programming, a problem area encountered during integration from lower level modules up to higher level modules may require considerable mcdification of the lower level modules to ensure that the interface requirements are met.

  Furthermore, the amount of testing necessary fcr integration or system test is variable and difficult to predict. All too often, communication between members cf a project is such that each programmer has a slightly different view of the function and integration of his module into the overall program. This misunderstanding dces not generally become apparent until testing and integration are well advanced. With top-dcwn programming, all interfaces are fully defined, coded, and tested before coding starts on lower level modules.

- Programmer Resource Flexibility

  With the top-down technique, if a particular program falls behind schedule, other programming resources can support the coding of lower level modules while the original programmer continues with his higher level module.

The system design technique described in this publication is also a top-down design technique. As shown in Figure 1-1, systems design occurs at the highest level first by broadly defining the application requirements in terms of the input, general processing required, and the output. This highest level application design then allows the design team to descend to a lower level data communication design to define input and output, program design to define processing, and data base design. Furthermore, within these functions are additional levels, each level taking the design to a greater depth of detail.

STRUCTURED PROGRAMMING WITH CICS/VS

The CICS/VS program structure accommodates the top-down programming technique. A terminal transaction initiates an application program, which can be regarded as being at the highest level. This program can utilize a module at a lower level to carry out the necessary editing, another module to carry out the processing required by the transaction, and still another to prepare the output response.

Each of these lower level modules can be written separately, but does not have to be available before testing starts. For example, the editing module can be tested without the processing or output module being ready. Instead, dummy modules can be used in their places to indicate that control did pass from the higher level module to the processing and output modules. In the editing module, control can pass to lower level modules in the event of errors. Again, these error modules are not coded until the editing module has been tested.

In this way, development of the CICS/VS application program proceeds from the top down. Top-down programming definitely offers the most advantages for complex programs. In this case, the number of functions to be coded and tested may be sufficiently complex that the development of the application program is a major task.

Lower level modules need not necessarily be separate CICS/VS application programs. Instead, the technique of top-down programming can be used in a single CICS/VS application program, with the main-line of the program being the highest level. This may call various subroutines. These subroutines may be replaced during testing with a dummy subroutine to implement the top-down approach. An example would be the use of the PERFORM verb in COBOL to execute a number of "dummy" paragraphs. The actual paragraphs may be incorporated in the program at a later time, when the main logic flow of the program has been fully tested. This technique allows easier initial testing, without having to test out all program logic right from the start of testing.

The use of CICS/VS as the DB/DC system represents the start of a top-down programming technique. CICS/VS is the main controlling routine, which in turn calls a number of modules at a lower level--that is, the application programs.

APPLICATION FUNCTIONS

In designing an online application to execute under control of CICS/VS, the design team should be aware of those application functions offered by CICS/VS. Some of these are referred to as built-in CICS/VS functions, while other functions (such as message routing, terminal paging, and device independence) use CICS/VS basic mapping support (BMS). Consequently, these functions can only be used with terminals supported by BMS. These application functions provide the facilities listed in Figure 2-4. They are summarized here, and described in detail in Chapter 3 unless stated otherwise.

MESSAGE ROUTING

CICS/VS provides an optional message routing and broadcasting capability. This enables any terminal to transmit messages to other BMS-supported terminals in the system, either immediately, or at some future time, provided that all affected terminals are of a type supported by basic mapping support. While message routing may have relevance in specific application design, it is particularly important in enabling the master terminal operator, who has the overall responsibility for control of the online applications, to communicate with each terminal operator.

TERMINAL PAGING

The terminal paging facility of CICS/VS enables application programs to develop information to be presented to BMS-supported terminals as a series of pages. However, the sequence of these pages requested by the terminal operator is not important to the application program. CICS/VS-provided terminal operator commands enable the operator to request the display of pages in any sequence desired.

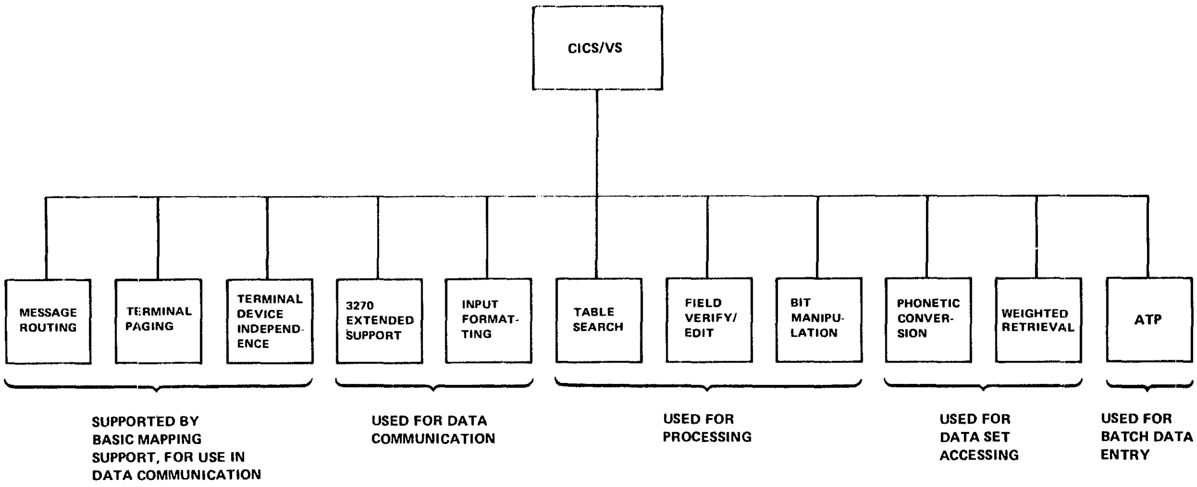


Figure 2-4. CICS/VS Application Built-in Functions

TERMINAL DEVICE INDEPENDENCE

CICS/VS terminal device independence enables transactions to be entered from any BMS-supported terminal type, and presents those transactions to the application program in a standard form. The output response developed by the application program can be presented in standard form to CICS/VS, which then prepares it for transmission to the relevant BMS-supported terminal. The application program is relieved of most considerations regarding device-dependent requirements for terminals. It can accept input from any BMS-supported terminal in the network and prepare terminal output as a series of lines regardless of the particular terminal type to be used.

EXTENDED 3270 SUPPORT

This added support enables the design of input transactions to take advantage of the 3270 Program Attention (PA) or Program Function (PF) keys to initiate transactions. This enables frequently used transactions to be initiated by one key depression, instead of the use of a transaction code which normally is from one to four characters long.

In addition, a specific PA or PF key can be defined as a 3270 PRINT key. Depression of this key enables the contents of the 3270 screen to be printed on the first available printer identified for this purpose. The 3270 selector light pen can also be used to initiate transactions.

INPUT FORMATTING

This CICS/VS built-in function enables input transactions to be entered in a variety of formats. They are converted to a standard format for presentation to application programs for processing. This can enable application programs to be developed relatively independent of the way in which specific transactions are entered at a terminal.

TABLE SEARCH

This built-in function enables a table of information to be readily searched to extract the appropriate value from that table based upon a search argument. The table search can be either a sequential or a binary search.

FIELD VERIFY/EDIT

Editing macro instructions are provided by this CICS/VS built-in function to enable the contents of a field to be examined for:

- All numeric (0 to 9)
- All alphabetic (A to Z, or blanks)
- All packed decimal (COMPUTATIONAL-3 in American National Standard (ANS) COBOL or FIXED DECIMAL in PL/I)

User routines can be executed in the event that characters other than those specified for a field are present.

A macro instruction is also provided to edit nonnumeric information from a field (for example, part number 119-445/B) and present the remaining numeric characters in EBCDIC.

BIT MANIPULATION

The ability to test the status of individual bits, and to turn bits on or off, is provided through the use of CICS/VS macro instructions for Assembler, PL/I, and COBOL. This built-in function is especially useful in COBOL, which does not have a standard bit manipulation capability.

PHONETIC CONVERSION

This built-in function enables misspelled names to be used as keys to access data sets. The name is converted to a standard key based upon the phonetic sound of the name. For example, the names Smith, Smyth, Smythe, and Smiths result in the phonetic code S530.

This is particularly useful for identification of names, such as in a police information system, customer information systems in banking, insurance and medical applications, or product names in order entry applications.

A phonetic conversion subroutine is also provided for use by batch programs executing in partitions other than the one containing CICS/VS. This subroutine can be used for batch programs executed under the control of DOS/VS, OS/VS1, or OS/VS2.

Refer to "Record Identification" in Chapter 5 for a more detailed discussion of phonetic conversion.

WEIGHTED RETRIEVAL

This powerful built-in function provides CICS/VS application programs with the ability to search part, or all, of a specified VSAM data set, and retrieve information from that data set based upon user-specified selection criteria. Furthermore, records satisfying the criteria are indicated as relevant only if they fall between user-specified limits. The records that fall between the specified limits are then presented to the application program, with those records best satisfying the criteria presented first, followed by records satisfying the criteria least.

This function is useful for the design and development of query applications. Queries can be designed based upon the selection of information meeting fixed criteria specified in a program. Alternatively, the design team can define user transactions and programs which permit terminal operators to specify the relevant selection criteria, or selection limits, to permit "ad-hoc" queries to be entered by terminal operators. Refer to "Weighted Retrieval" in Chapter 5 for more detail and for specific application examples of the use of this function.

ASYNCHRONOUS TRANSACTION PROCESSING (ATP)

This is not a built-in function, but is supplied by CICS/VS to provide a batch data collection capability, oriented to high-volume data transmission from remote batch terminals. Specifically, this function enables batches of data to be entered from remote terminals and queued by CICS/VS for processing. On receipt of the entire batch, CICS/VS initiates the processing of that batch while the terminal is able to transmit further batches to the system.

Messages describing any errors detected during application-program processing of the batch are queued by CICS/VS. These error messages are transmitted back to the remote terminal on request, to permit batch error correction and resubmission of corrections if required. Refer to "Asynchronous Transaction Processing" in Chapter 3 for more detail.

QUASI-REENTRANT PROGRAMMING

For efficient utilization of storage, CICS/VS ensures (unless requested otherwise) that only one copy of a program will reside in the dynamic storage area. All tasks requiring the use of that program are able to execute that program concurrently.

In order to achieve a high degree of multitasking, CICS/VS supports quasi-reentrancy. This allows several tasks to utilize the same section of code over the same period of time. However, it differs from fully reentrant programming in that control is only passed from one task to another when the active task issues a CICS/VS macro instruction. Control will not pass from one task to another on an I/O interrupt, for example, as is the case in a DOS/VS or OS/VS multitasking environment. CICS/VS provides a quasi-reentrant capability for Assembler, ANS COBOL, and PL/I.

Information unique to the processing of a transaction (such as the terminal input area, file I/O areas, or work areas) is separated from the body of the application program. Instead of these areas residing within the program, they are allocated from dynamic storage. The execution of each separate transaction in a multitasking environment is controlled by a task control area (TCA) that contains address pointers and other vital information for that particular transaction (task). Because the information unique to a task is separated from the main body of a program, the program can be used concurrently by several tasks. The access methods are incorporated within the CICS/VS nucleus, and exception or error routines are included in other CICS/VS application programs. Figure 2-5 shows the concept of quasi-reentrant programming. This figure is discussed in the *CICS/VS General Information Manual*.

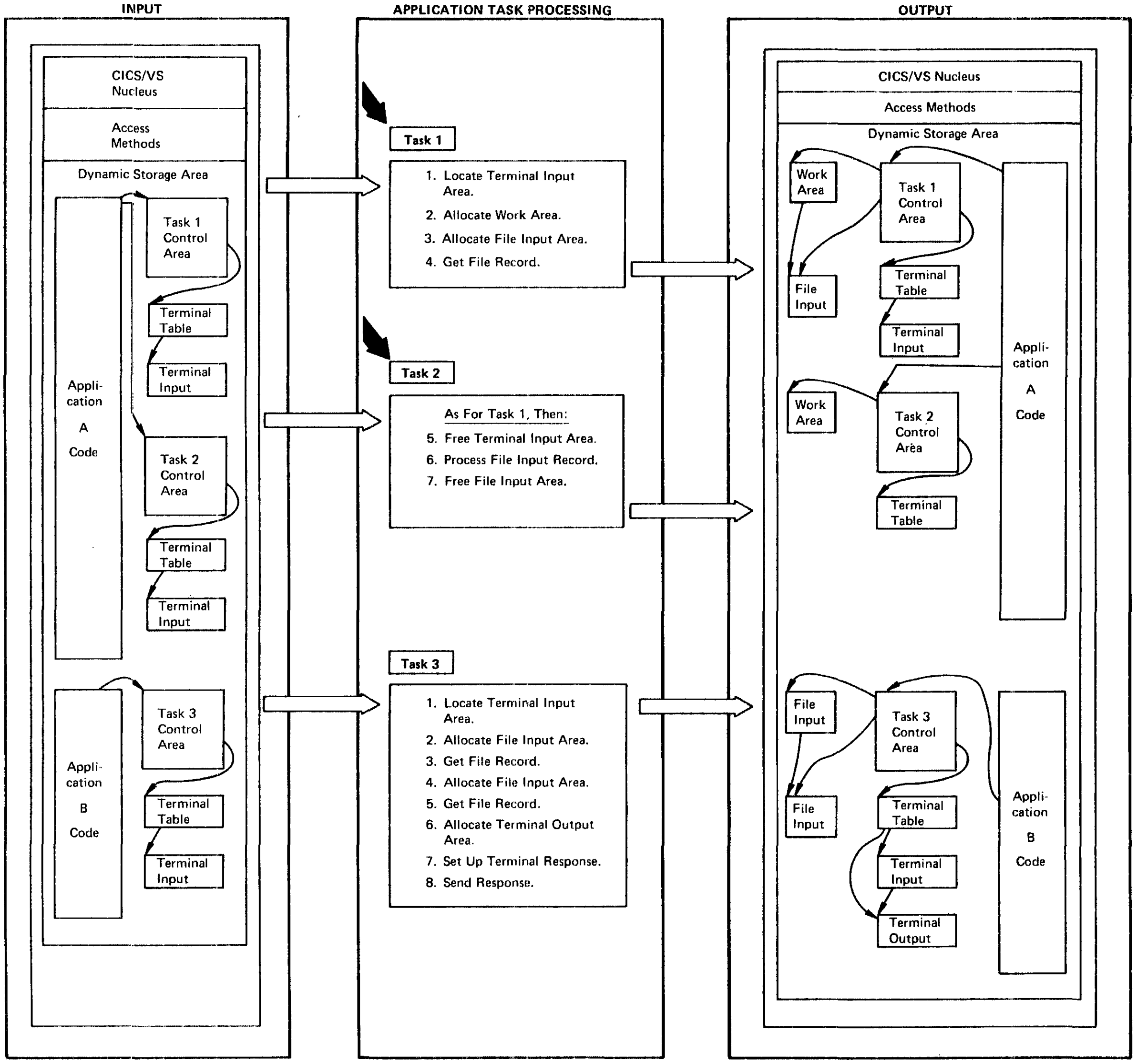


Figure 2-5. Quasi-Reentrant Programming and Multitasking

TASK INITIATION

There are several methods utilized by CICS/VS tc initiate tasks. These are briefly outlined here, but discussed in more detail in later sections of this publication.

TRANSACTION CODES

CICS/VS examines a transaction code received as part of a terminal message, to identify the particular transaction involved. This transaction code must occupy the first one to four characters of the transaction invocation message. An input message is considered a transaction invocation when it occurs and no task is active on the terminal. This transaction code is validated by CICS/VS against a program control table (PCT). If the specific transaction code exists in that table, the transaction is assumed to be a valid one, and the transaction is passed to that program identified in the relevant PCT entry, for processing (see "Task Initiation" in Chapter 3 for more detail).

The 3270 enables transactions to be initiated by the use of a Program Attention (PA) key, Program Function (PF) key, or selector light pen.

AUTOMATIC TASK INITIATION

Automatic task initiation involves the queuing of transactions on disk using CICS/VS transient data management. A number of transactions may be queued based upon a specific trigger level. When the number of transactions queued reaches this trigger level, CICS/VS automatically utilizes a specified transaction code for that queue to initiate a task and allow those queued transactions to be processed by a specific program. (See "Intrapartition Queue Usage" in Chapter 4.)

INTERVAL CONTROL

CICS/VS enables a task to be initiated using a specified transaction code at some future time, based upon time of day or on elapsed time. Data may be passed to that future task for use in processing when it has been initiated. (See Chapter 6.)

TASK CONTROL

A task can be explicitly initiated from a CICS/VS application program by the use of the CICS/VS task control ATTACH macro instruction. This macro instruction specifies the transaction code to be used to identify the program which will process the transaction. (See Chapter 6.)

PROGRAM CONTROL

A program may pass control to other programs in a variety of ways. These are illustrated in Figure 2-6 and described briefly below. See the *CICS/VS Application Programmer's Reference Manual*, SH20-9003, for additional information.

TRANSFER CONTROL TO PROGRAM (XCTL)

This macro instruction (XCTL) enables one application program (referred to as the calling program) to pass control to another application program (referred to as the called program). Control is not returned to the calling program on completion of execution of the called program.

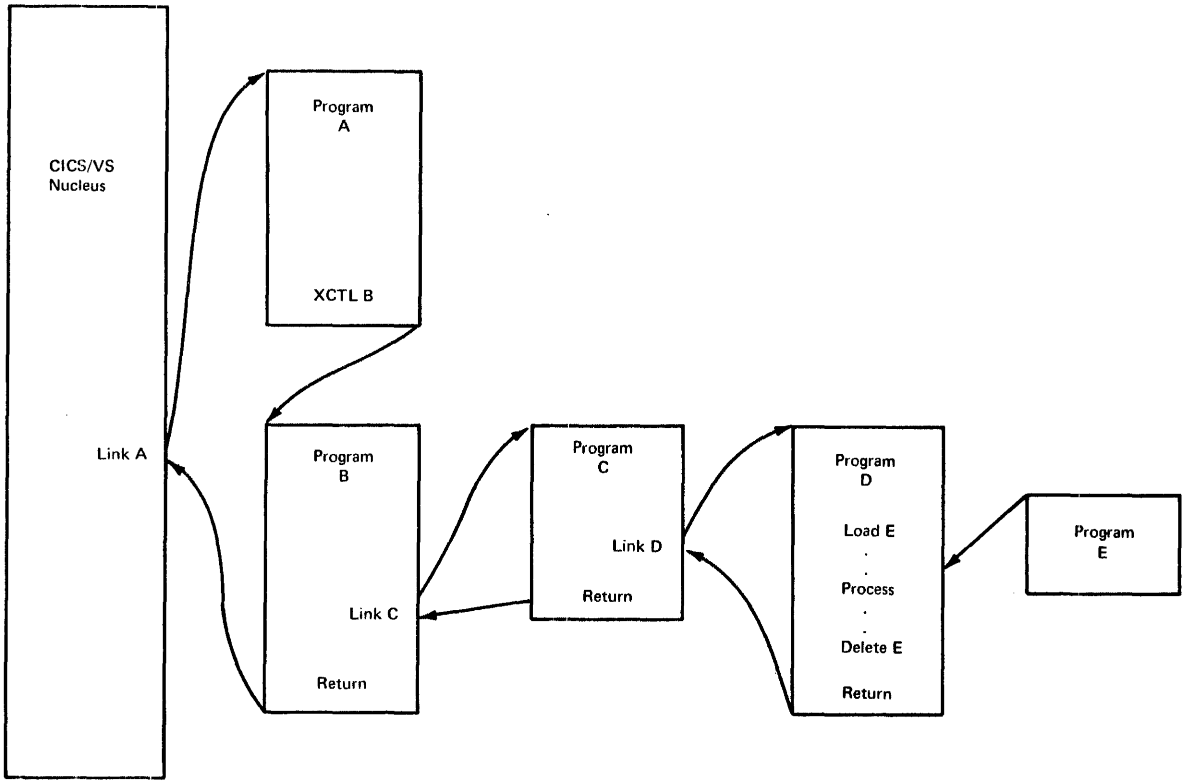


Figure 2-6. CICS/VS Program Control Facilities

LINK TO PROGRAM (LINK)

This macro instruction specifies the name of an application program to be executed. The calling program passes control to the called program. On completion of execution, control is returned to the calling program, to the statement following the LINK.

LOAD PROGRAM (LOAD)

This macro instruction enables a program, identified by name, to be loaded into storage. However, control is not passed to that program for execution, but is returned to the statement following the LOAD macro instruction. The LOAD macro instruction can be used to load application programs, which may subsequently be linked to or transferred to. Alternatively, this macro instruction may be used to load tables of information.

DELETE PROGRAM (DELETE)

Normally, on completion of execution of a task, all storage utilized by that task is automatically freed and made available for use in processing other transactions. Active programs being used by tasks will continue to reside in storage. If, however, the storage occupied by an inactive program is required for some other executing task, that storage will be freed. When a program is loaded, its storage can be automatically freed if it is currently inactive, allowing another task to use that storage.

Alternatively, the LOAD macro instruction can specify that the storage is not to be freed, but that the program is to remain resident in storage even though inactive, for performance reasons.

The DELETE macro instruction is used to delete such a resident program at a point which will enable its storage to be utilized for other functions.

RETURN FROM PROGRAM (RETURN)

The Program Control RETURN macro instruction enables a program to return control to a higher level program, that program can be either another application program or CICS/VS if the RETURN is issued by the application program at the highest level. The RETURN indicates that the relevant program has now completed processing. At task completion, CICS/VS frees all of the storage associated with the task, such as terminal I/O areas, file I/O areas, and file work areas, and eventually also frees the storage occupied by the task control area. Optionally, a transaction code may be specified in the RETURN macro instruction. The transaction code is used with the next input message from the same terminal that originated this completed task.

ABNORMALLY TERMINATE PROGRAM (ABEND)

This macro instruction enables a program to immediately terminate execution of a task, with an optional dump if required. In conjunction with the optional operator signon facility (see Chapter 3), the ABEND macro instruction can be used to develop operator error statistics.

ABNORMAL TERMINATION EXIT (SETXIT)

This macro instruction enables a task to activate, deactivate, or reestablish a program-provided exit routine to be executed in event of abnormal termination of the task. This exit routine can be utilized either on CICS/VS abnormal task termination, or by termination through the use of the ABEND macro instruction by the task.

An abnormal termination exit routine is used to complete urgent processing by a task for recovery purposes, or it may attempt recovery of the particular error condition itself. Refer to "Program Error Recovery" in the following section for more detail.

PROGRAM ERROR RECOVERY

CICS/VS features facilities for detection of program error situations, and protection of the online system from the effect of these errors. If a program-check error is detected in an application program, CICS/VS will attempt to abnormally terminate that task while still permitting other tasks to continue processing.

PROGRAM ERROR PROCESSING

CICS/VS enables an application program to indicate, through the use of the program control SETXIT macro instruction, that control is to be passed to a specified routine in the program in the event of a program error, or to another program. This routine (or program) may attempt recovery of the error situation, record certain critical information necessary to the application before the error program is abnormally terminated, or ignore the error situation and continue program processing.

PROGRAM ERROR PROGRAM

In the event that a program error requires abnormal termination of a task, the terminal operator who invoked the transaction to be abnormally terminated will be notified of this fact by CICS/VS, if appropriate. CICS/VS passes control to a program error program (PEP), after all SETXIT processing has been completed for the task and the decision has been made to permit the abnormal termination to continue. This PEP is a user-written routine, given control through a LINK from the CICS/VS abnormal condition program (ACP), which enables the user to perform installation-level abnormal termination action.

SETXIT program processing and PEP are described further in "System Recovery Program" in Chapter 8. These facilities can be used to provide program backup capability for online applications.

CHAPTER 3. CICS/VS DATA COMMUNICATIONS DESIGN

This chapter presents Data Communications Design in a tutorial fashion. Experienced CICS users may wish to omit reading much of this chapter. It is suggested, however, that such users should read the following topics:

- Terminal device independence
- Terminal paging
- Message routing
- Virtual Telecommunications Access Method (VTAM)
- Task initiation
- Input transaction design
- Field verify/edit
- Table search

These topics describe changed, or new, facilities which were not available in previous versions of CICS.

The Data Communications design has a significant effect on the success or failure of the overall project. It is in this area that the interface between the user and the computer is defined. This definition should be oriented toward satisfying the requirements of the user and the application, while still presenting information to the computer in a suitable form for processing.

Before discussing various Data Communications design approaches, it is important that the reader understand the following CICS/VS support and telecommunications access method facilities which will aid him in his design:

- Basic mapping support
- Terminal device independence
- Terminal paging
- Message routing
- Virtual Telecommunications Access Method (VTAM)

With these features available to the system designer, various approaches for conversational CICS/VS applications and batch CICS/VS applications can be considered.

BASIC MAPPING SUPPORT

The basic mapping support function (BMS) enables the application program to have access to input data, and prepare output data for transmission to terminals, without regard to the physical location of the data in the terminal message. Additional information regarding basic mapping support can be found in the *CICS/VS Application Programmer's Reference Manual*, SH20-9003.

BMS uses "maps" to describe the input format of data received from terminals, and (if necessary) to describe the format of output data to be transmitted to terminals. These maps are defined by the user (generally the system programmer) and are separately assembled and cataloged into the CICS/VS program library for retrieval when needed by application programs.

The application program accesses data from input messages, and prepares data for output responses, by field rather than by location of that information in the terminal message. Consequently, the application becomes less dependent on the actual message format. This format independence is one of the most significant advantages of BMS. Changes to message formats to meet various application requirements can be readily applied, by modifying only the BMS maps describing those affected formats, reassembling them, and cataloging the changed maps to the CICS/VS program library. All programs using these maps reflect the changed formats without modification of the programs. However, recompilation of the programs might be necessary. In this manner, the installation will be more responsive to application needs. The use of BMS by application programs is illustrated in Figure 3-1.

BMS MAPS

An input map can specify data in an input message which is relevant to a particular program and ignore other data in the input. Several programs can then operate on the same input message format, using a unique map for each program. In addition, constant (or descriptive) information, if desired, can be defined in an output map and be automatically incorporated by BMS in an output message.

Basic mapping support provides the following services:

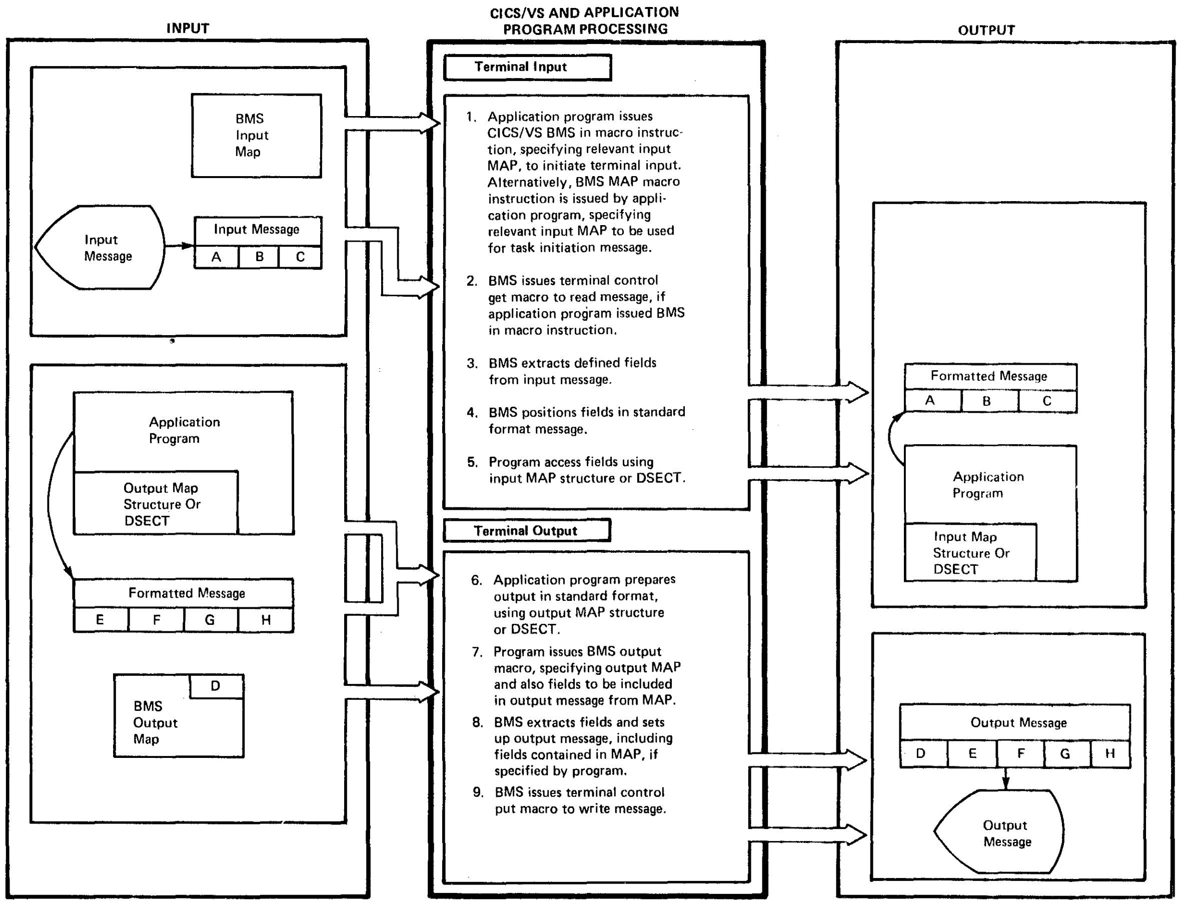
- Terminal device independence
- Terminal paging
- Message routing

TERMINAL DEVICE INDEPENDENCE

Because Basic Mapping Support removes the need for the application programmer to code most terminal device-dependent support in his programs, programs can be written without regard to the input or output device used for transmission of messages to those terminals supported by BMS. BMS accepts input messages and transmits output messages to and from the devices below (see Figure 3-2).

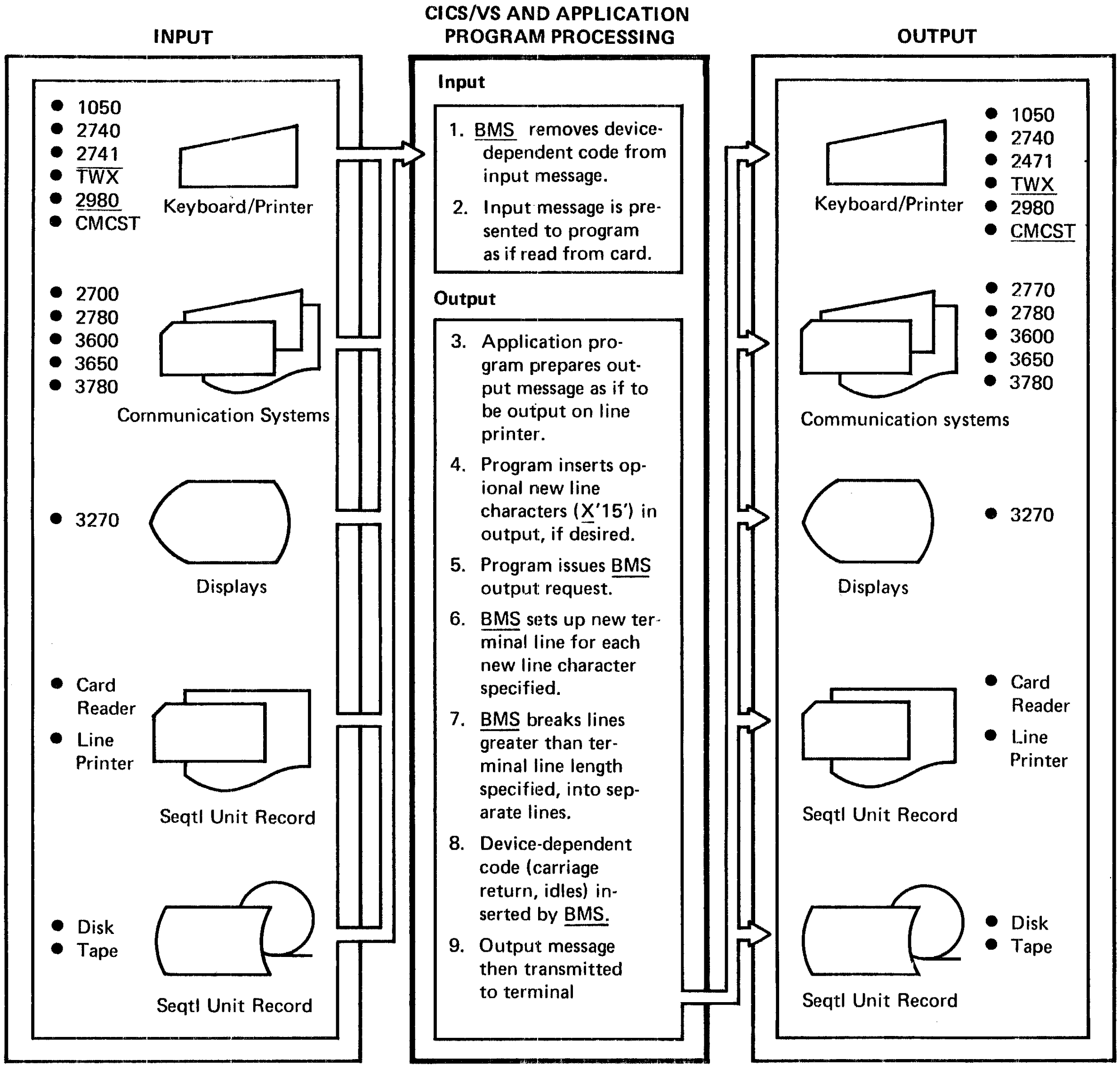
- 1050 Data Communication System
- 2740 Communications Terminal, Models 1 and 2
- 2741 Communications Terminal
- 2770 Data Communication System
- 2780 Data Transmission Terminal
- 2980 General Banking System (keyboard and printer only)

- 3270 Information Display System
- Teletype Terminals
- Communicating Magnetic Card SELECTRIC ® Typewriter (CMCST) Model 6610, used as a 2741



1. Application program issues CICS/VS BMS in macro instruction, specifying relevant input MAP, to initiate terminal input. Alternatively, BMS MAP macro instruction is issued by application program, specifying relevant input MAP to be used for task initiation message.
2. BMS issues terminal control get macro to read message, if application program issued BMS in macro instruction.
3. BMS extracts defined fields from input message.
4. BMS positions fields in standard format message.
5. Program access fields using input MAP structure or DSECT.
6. Application program prepares output in standard format, using output MAP structure or DSECT.
7. Program issues BMS output macro, specifying output MAP and also fields to be included in output message from MAP.
8. BMS extracts fields and sets up output message, including fields contained in MAP, if specified by program.
9. BMS issues terminal control put macro to write message.

Figure 3-1. CICS/VS Basic Mapping Support (BMS)



Input

1. BMS removes device-dependent code from input message.
2. Input message is presented to program as if read from card.

Output

3. Application program prepares output message as if to be output on line printer.
4. Program inserts opional new line characters (X'15') in output, if desired.
5. Program issues BMS output request.
6. BMS sets up new terminal line for each new line character specified.
7. BMS breaks lines greater than terminal line length specified, into separate lines.
8. Device-dependent code (carriage return, idles) inserted by BMS.
9. Output message then transmitted to terminal

Extended Description
Device-dependent code may be removed by programmable controllers, such as the 3600, before transmission to CICS/VS. For the 3600 and 3650, input mapping requests are ignored.

Figure 3-2. CICS/VS Terminal Device Independence

- 3780 Communication System
- 3600 Finance Communication System (BMS input requests are ignored)
- 3650 Retail Store System

Furthermore, the following sequential devices can be used to simulate online terminals, and transmit simulated terminal messages to and from the system:

- Card reader/line printer

- Tape drives

- Disk drives

INPUT MESSAGES

CICS/VS accepts input messages from any of the above devices and, using the input map specified by the application program, converts the input message into a fixed format message, as specified by that map. Device-dependent characteristics in the input message are removed, and the appropriate fields are selected from the message and inserted in fixed locations in the mapped message.

In the case of the 3600, device-dependent characteristics in the input message are removed by the 3601 Controller and the input message is formatted for CICS/VS application program processing before transmission to CICS/VS. Consequently, BMS input mapping requests associated with 3600 input messages are ignored, and the data received from the 3600 is passed to the CICS/VS application program without change. See "Basic Mapping Support Using VTAM" for additional information.

OUTPUT MESSAGES

Output messages for transmission to terminals can be prepared without the control characters required for field positioning, or line separation. Output messages can be presented to CICS/VS as a data stream. Optionally, the application program can insert new line (X'15') characters in the output data stream if required.

CICS/VS device-independent support divides the data stream into lines no longer than those defined for the particular terminal. If new-line characters appear occasionally in the data stream to further define line lengths, they are honored. CICS/VS inserts the appropriate leading characters, carriage returns, and idle characters, and truncates trailing blanks from each line.

Terminal device independence permits an application program to be independent of the terminal type (or types) in the installation, and can provide for support of mixed terminal types by the same program. This allows the use of backup terminals of a different type, or changeover of hard-copy terminals to display terminals when transaction volumes warrant the change, with little or no additional programming effort. It also reduces the amount of program maintenance necessary when changes are made in the terminal devices used by online programs, and permits increased growth flexibility in the installation.

Terminal device independence is the only BMS support available with the subset option of CICS/DOS/VS. Terminal paging and message routing are not supported. (See "CICS/DOS/VS Subset Option" in Chapter 7.)

TERMINAL PAGING

Terminal paging is an additional feature that extends the capabilities of terminal device independence. The application programmer can prepare more output than can be conveniently or physically displayed at the receiving terminal. That output can be presented by CICS/VS as a series of pages. CICS/VS identifies and saves each page of information prepared by the application program.

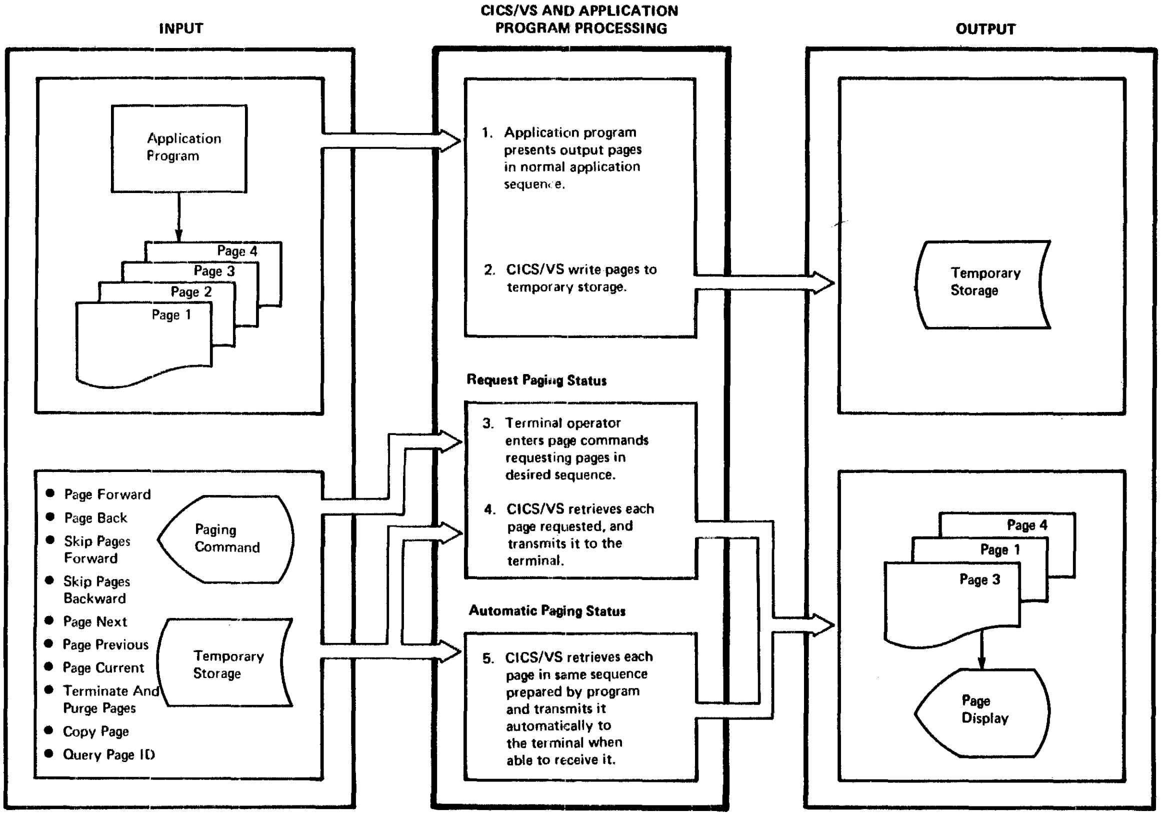

INPUT
CICS/VS AND APPLICATION PROGRAM PROCESSING
OUTPUT
Application Program
Page 4
Page 3
Page 2
Page 1
1. Application program presents output pages in normal application sequence.
2. CICS/VS write pages to temporary storage.
Temporary Storage
Request Paging Status
3. Terminal operator enters page commands requesting pages in desired sequence.
4. CICS/VS retrieves each page requested, and transmits it to the terminal.
Automatic Paging Status
5. CICS/VS retrieves each page in same sequence prepared by program and transmits it automatically to the terminal when able to receive it.
● Page Forward
● Page Back
● Skip Pages Forward
● Skip Pages Backward
● Page Next
● Page Previous
● Page Current
● Terminate And Purge Pages
● Copy Page
● Query Page ID
Paging Command
Temporary Storage
Page 4
Page 1
Page 3
Page Display

Figure 3-3. CICS/VS Terminal Paging

The terminal operator can then retrieve this output as a number of pages in any order; that is, in the crder they were prepared, or by skipping forward or backward in the cutput page sequence.

CICS/VS provides a series of paging commands which can be used by the terminal operatcr to select pages for display in whichever sequence he desires (see Figure 3-3).

Terminal paging also provides the ability to combine several small sections of data intc cne page which is then sent to the terminal. This is referred to as "page building" and enables the application programmer to prepare his output independent of the physical output capability of the terminal.

Terminal paging further relieves the application programmer of the need to concern himself with the presentation of information in a form suitable for display at the appropriate terminal, or with presentation of that information to the terminal in the sequence requested by the terminal operator. The application programmer can now prepare a series of pages of information, on the assumption that the terminal operator may wish to examine all of that information, and then present those pages directly to CICS/VS. No further programming is necessary to handle the selection of pages for display at the terminal. Page selection is made by the terminal operator, using the CICS/VS paging commands.

This will simplify program development of conversational applications and consequently increase programmer productivity and decrease the amount of future program maintenance necessary.

It is important that the system designer recognize that terminal pages are saved by CICS/VS in temporary storage. Temporary storage may be supported in main storage alone, or on auxiliary storage using VSAM; both will increase the demands for real storage during execution. Using VSAM on a CPU with limited real storage available for virtual storage paging may increase paging and therefore influence online performance. Refer to Chapter 7 for recommended minimum CPU sizes when VSAM is used. Terminal paging is not supported by the subset option of CICS/DOS/VS. (See "CICS/DOS/VS Subset Option.")

TERMINAL PAGING STATUS

Terminal paging is particularly oriented toward display terminals. However, it can also be used for hard-copy terminals. A terminal can be defined as having a "request paging" status or an "automatic paging" status.

Display terminals must use a request paging status, while hard-copy terminals can use either request paging or automatic paging status. Request paging status enables pages to be displayed at the terminal on request by the terminal operator, who can specify the sequence of pages to be displayed based upon his requirements.

Automatic paging status, such as normally used for a hard-copy terminal, causes CICS/VS to automatically output the next page of information on completion of a previous page. In this way, all information is presented to the hard-copy terminal in a continuous output stream. If required, an automatic paging terminal may be changed to request paging status by either the terminal operator or the application program, enabling only those pages to be printed which are of significance to the terminal operator. Similarly, the terminal operator or the application program can change request paging status to automatic paging for all terminals except display terminals.

Other terminal status specifications can also be used to indicate whether a terminal automatically receives messages sent from the CPU or from other terminals. This is discussed under "Terminal Status" in Chapter 4. Additional information on terminal paging can be found in the *CICS/VS Application Programmer's Reference Manual*, SH20-9003.

MESSAGE ROUTING

Message routing directs messages to one or more terminals in the system, either by use of the message switching transaction, CMSG, supplied by CICS/VS, or by the BMS ROUTE macro instruction. In this context, the term "message switching" refers to the use of CMSG. The term "message routing" refers to the use of the BMS ROUTE macro instruction. (The CMSG transaction itself uses the services of the BMS ROUTE macro instruction.)

The CMSG transaction is entered by a terminal operator together with a message to be directed tc ancther terminal, or to several terminals identified by the terminal operator. (This is discussed further in "Message Switching Transacticn (CMSG)" later in this chapter.)

The message rcuting macro instructicn permits an application program to send messages to one cr mcre terminals not in direct ccntrol of the transaction. Message routing uses BMS, and saves messages in temporary storage to be automatically sent to the specified destination terminals if the status of those terminals allcws for reception of the messages (refer to "Terminal Status" in Chapter 4). If a terminal is not immediately eligible to receive the message, CICS/VS preserves it in tempoary storage until such time as a change in terminal status allows it to be sent, cr a user-specified period of time elapses, whichever occurs first (see Figure 3-4). The message to be delivered is separated into a message for each terminal type that will receive it. Each separate terminal-type message is saved in tempoary storage, together with a destination terminal list fcr that particular terminal type.

In addition, an applicaticn program can prepare pages of information to be transmitted to terminals, using BMS and the terminal paging facilities as described abcve. These pages can be routed to one or more terminals or operators, through the use of the BMS ROUTE macro instruction.

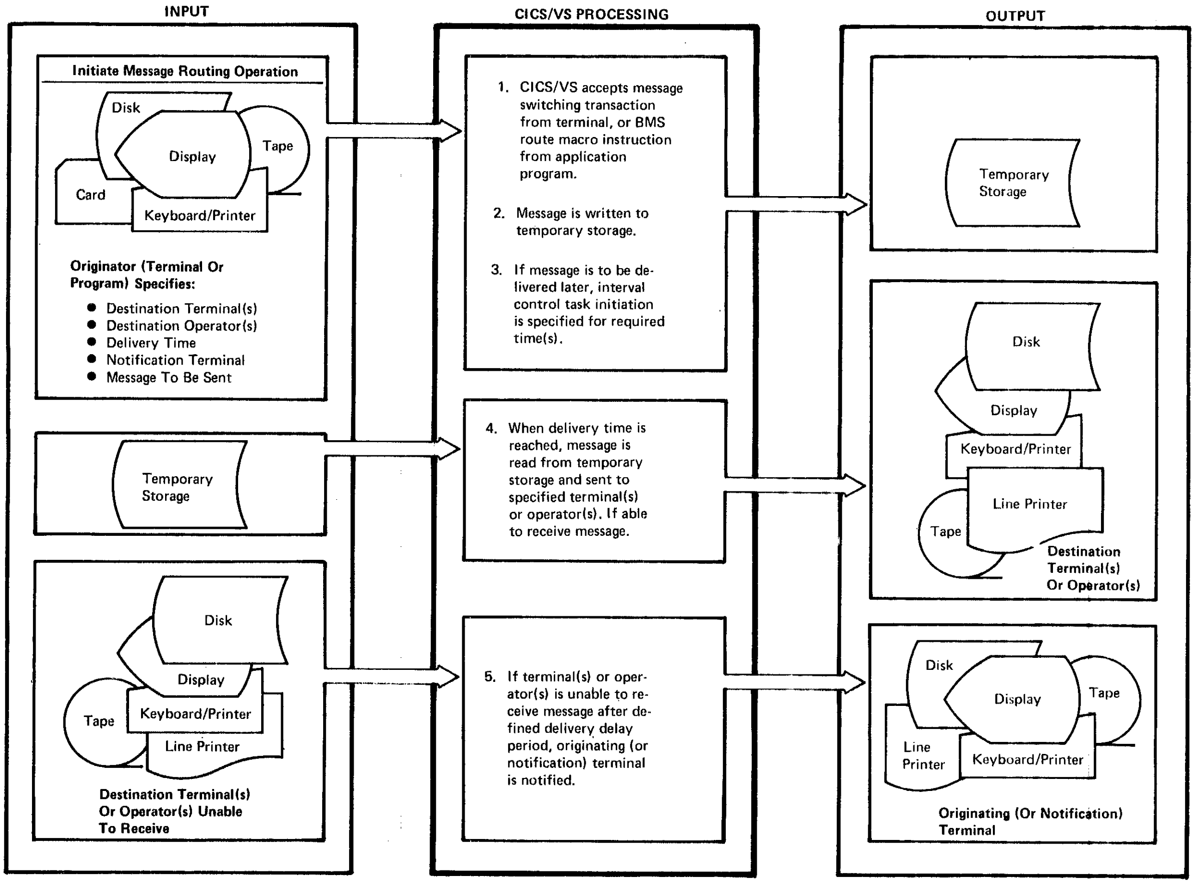
MESSAGE DELIVERY

The application programmer specifies the identification of the terminal (or terminals) tc receive the message, and, opticnally, can also specify a time when the message is to be delivered. If the message cannot be delivered either immediately or at the specified future time, CICS/VS retains the message fcr a user-specified period of time. If it still cannot be delivered, CICS/VS notifies the originating terminal or an alternative terminal specified when the original message was entered.

CICS/VS allows messages tc be directed, not only to specific terminals, but alsc to specific operators or operator classes. In this way, sensitive security informaticn will only be delivered to those operators authorized to receive it. It is retained in temporary storage until the specified operatcrs sign on to the specified terminals, and only then will relevant messages be delivered.

If a message is to be sent to a specified operator without identifying a terminal, that operator must already be signed on when the message is first presented tc CICS/VS to establish the terminal identificaticn tc be used. If a message is sent to a specific operator and terminal, and that operator can never use that terminal (because of geographic location, fcr example), the message will be accepted by CICS/VS but may never be delivered. This is noted by CICS/VS upon expiraticn of the specified time within which the message must be delivered.

Terminals in the IBM 3600 Finance Communicaticn System are identified by a lcgical device code (LDC). Messages from CICS/VS are received by the appropriate 3601 application program which represents the specified terminal ID and controls the devices attached to the 3601. The message from CICS/VS identifies the LDC (specified by the application programmer) that is to receive the message; it is the responsibility of the 3601 application prcgram to ensure that the message is delivered to the device indicated by the LDC. Logical device codes are described in detail in "Basic Mapping Suppcrt Using VTAM" in this chapter.



1. CICS/VS accepts message switching transaction from terminal, or BMS route macro instruction from application program.
2. Message is written to temporary storage.
3. If message is to be delivered later, interval control task initiation is specified for required time(s).
4. When delivery time is reached, message is read from temporary storage and sent to specified terminal(s) or operator(s). If able to receive message.
5. If terminal(s) or operator(s) is unable to receive message after defined delivery delay period, originating (or notification) terminal is notified.

Figure 3-4. CICS/VS Message Routing

MESSAGE SWITCHING TRANSACTION (CMSG)

CICS/VS provides the message switching transaction CMSG for transmission of information between terminals. Figure 3-4 shows the use of this transaction.

The availability of the message routing feature in CICS/VS provides a valuable capability for communication of information, not only between terminals to satisfy application requirements, but also for better control of the online applications by the master terminal operator or supervisory terminal operators. These operators may broadcast messages to all terminals under their control informing them of certain significant information.

CICS/VS message routine utilizes temporary storage, which will use VSAM if auxiliary storage residence of messages is desired. Message routing and message switching are not supported by the subset option of CICS/DOS/VS. (See "CICS/DOS/VS Subset Option" in Chapter 7.)

Additional information about basic mapping support can be found in the CICS/VS Application Programmer's Reference Manual, SH20-9003. 3600 logical device codes are described in more detail in the CICS/VS Advanced Communications Guide, SH20-9049.

CICS/VS TERMINAL CONTROL AND BMS

CICS/VS application programs can communicate directly with terminals, using:

- Basic mapping support (BMS) macro instructions

- Terminal control macro instructions

BMS enables application programs to request terminal input or output, using the DFHBMS macro instruction:

- For input (TYPE=IN or MAP)

- For output (TYPE=OUT)

- For terminal paging (TYPE=TEXTBLD, PAGEBLD or PAGEOUT)

- For message routing (TYPE=ROUTE)

Terminal control enables programs to request additional functions by using application programmer DFHTC macro instructions to:

- Write data to a terminal (TYPE=WRITE or TYPE=PUT)

- Read data from a terminal (TYPE=READ or TYPE=GET)

- Synchronize terminal I/O with processing (TYPE=WAIT)

- Transmit a page of data and read reply (TYPE=PAGE)

- Transmit to the buffer of a banking terminal (TYPE=CBUFF)

- Test for the presence of a banking passbook (TYPE=PASSBK)

- Reset a line (TYPE=RESET)

- Disconnect a switched line (TYPE=DISCONNECT)

- Erase and write data to a visual display (TYPE=(ERASE,WRITE))

- Specify the last output message to a VTAM-supported terminal (TYPE=LAST)

In addition, the system programmer may use the following DFHTC macro instructions to:

- Change the status of a terminal (CTYPE=STATUS)

- Locate a terminal entry in the TCT (CTYPE=LOCATE)

- Check the results of a previous STATUS or LOCATE request (CTYPE=CHECK)

These macro instructions are described briefly in "Terminal Control Using VTAM" and in more detail in the *CICS/VS System Programmer's Reference Manual*, SH20-9004.

The particular communication macro instructions used in programming are not significant to the following discussion of data communication design. However, if BMS is not used, the following facilities cannot be provided by CICS/VS:

- Terminal format independence
- Terminal device independence
- Terminal paging
- Message routing

3270 BMS can be specified during system generation to provide compatibility with previous versions of CICS. Terminal device independence, terminal paging, and message routing need not be specified. However, if they are specified, they require that temporary storage (and also VSAM) be used.

BMS and terminal control macro instructions can be used in the same application program, if required. Refer to the CICS/VS Application Programmer's Reference Manual for further information.

CPU CONSOLE AS A CICS/DOS/VS TERMINAL

CICS/DOS/VS allows the CPU printer keyboard or display console to be used as a CICS/DOS/VS terminal. Users with only remote terminals may enter master terminal operator, system administration, and CICS/VS application transactions at the CPU, thereby isolating these activities from any network considerations. See the Subset User's Guide (DOS), SH12-5404 for additional information about this feature.

BASIC TELECOMMUNICATIONS ACCESS METHOD (BTAM)

The Basic Telecommunications Access Method (BTAM) is used by CICS/VS to support a wide variety of terminals. These include the following:

- 1050 Data Communication System
- 2260 Display Station
- 2265 Display Station
- 2740 Communication Terminal
- 2741 Communication Terminal
- Communicating Magnetic Card SELECTRIC ® Typewriter (CMCST) Model 6610, used as a 2741
- 7770 Audio Response Unit
- TWX Common Carrier Teletypewriter Exchange Terminal Station (Model 33/35)
- 2770 Data Communication System
- 2780 Data Transmission Terminal
- 2980 General Banking Terminal System
- 3270 Information Display System
- 3735 Programmable Buffered Terminal
- 3740 Data Entry System
- 3780 Data Communications Terminal

- System/3
- System/7
- System/370
- 3767 Communications Terminal (using 2740/2741 Compatibility Feature)
- 3770 Data Communications System (using 2770 Compatibility Feature)

See the CICS/VS System Programmer's Reference Manual for additional information concerning components supported and features required for these terminals.

The subset option of CICS/DOS/VS uses BTAM to support the following terminals:

- 3270 Information Display System (local and remote)
- 2740 Communication Terminal
- 2741 Communication Terminal

See the Subset User's Guide (DOS) for additional information.

BASIC GRAPHICS ACCESS METHOD (BGAM)

The Basic Graphics Access Method (BGAM) is used by CICS/OS/VS only. It supports the local connection of the 2260 Display Station.

TELECOMMUNICATIONS ACCESS METHOD (TCAM)

CICS/OS/VS provides an interface to the Telecommunications Access Method (TCAM). This interface enables CICS/OS/VS to run with other TCAM applications under a common TCAM environment, or under TCAM through a VTAM-controlled terminal environment.

CICS/VS accepts data streams from TCAM-supported terminals which can be edited in the message handler portion of the TCAM message control program (MCP) to appear as EBCDIC or 3270 data streams.

With the exception of 7770 and 3270/2260 compatibility support, the TCAM support provided in CICS/OS Standard Version 2.3 (Program Number 5736-XX7) is upward compatible to the CICS/OS/VS TCAM interface. See the CICS General Information Manual, GH20-1028 (for information regarding CICS/OS STANDARD Version 2.3, Program Number 5734-XX7) and the OS TCAM Programmer's Guide and Reference Manual, GC30-2024, for information regarding the terminals supported.

Other TCAM-supported terminals, which were not supported by CICS/OS STANDARD Version 2.3, are supported as data streams which are terminal-type transparent to CICS/OS/VS.

TCAM is an optional access method which may be used alone, or in combination with other access methods supported by CICS/OS/VS. When communicating with terminals attached to 3704 or 3705 Communications Controllers in NCP mode, TCAM must communicate through VTAM.

VIRTUAL TELECOMMUNICATIONS ACCESS METHOD (VTAM)

The Virtual Telecommunications Access Method (VTAM) is used by CICS/VS to support a number of programmable terminal systems. These terminal systems enable programming tasks (such as transaction editing, validation, and formatting) relevant to a remote location to be carried out in programmable control units at the remote location. The programmable control units can operate either online to a CPU, or offline in a standalone mode. Operating offline, many of the control units can permit terminal operation to continue independent of availability of the main CPU or associated communication links. Terminal transactions can be recorded on disk storage (which is part of the programmable control unit) for later transmission when CPU communication is available. Disk storage also enables selected application data sets to be stored in the programmable control unit for direct reference by application programs executed in the control unit on behalf of terminals.

This concept of "distributed function" enhances performance and permits cost efficiencies to be realized in the areas of data transmission, system availability and configuration, and application flexibility.

The Network Control System (NCS), which may be used by CICS/DOS/VS, provides a subset of the support available with VTAM. Refer to "Related Publications" in the Preface for NCS publications.

SYNCHRONOUS DATA LINK CONTROL (SDLC)

Further efficiency in data transmission and data integrity is realized through the use of synchronous data link control (SDLC) data transmission. SDLC allows data to be transmitted in full-duplex mode (transmitted simultaneously in both directions along a communications line). VTAM supports both SDLC and full-duplex transmission.

A significant feature offered by SDLC is data integrity. Both VTAM and the control unit can check for error-free transmission of data and can request retransmission if an error is detected. Each transmission between VTAM and a programmable control unit is assigned a sequence number. Messages lost, because of component or communication failures, are easily detected and the lost data recovered. If recovery is not possible at the time of detection, the end component (VTAM or the control unit) requesting recovery can switch to an alternate processing mode.

If communication is lost between the CPU and the control unit, the control unit may switch to offline mode. Transactions entered from the attached terminals are processed by referencing data sets stored on the disk storage in the controller. The transactions can be stored on the disk for later batch transmission to the CPU when communications are restored.

Message recovery and resynchronization are handled automatically by VTAM and the controller. Either end of the link can transmit a command to test the sequence numbers of the last transmitted messages and to set the sequence numbers to reflect the status of the CPU and control unit. If this operation determines that messages were lost, resynchronization is accomplished by retransmitting the lost messages from copies stored on the disk storage.

See *IBM Synchronous Data Link Control: General Information*, GA27-3093, for additional information concerning SDLC.

CICS/VS uses VTAM and SDLC to support the following programmable control unit terminal systems:

- IBM 3600 Finance Communication System
- IBM 3650 Retail Store System
- IBM 3790 Communications System

See "CICS/VS Session Types" for additional information about these systems, and see _CICS/VS System Programmer's Reference Manual_ SH20-9004, for features supported or required by CICS/VS.

VTAM NETWORK

The VTAM network consists of the following components:

- communications controllers
- communication lines
- programmable terminal systems

These components are controlled by the following programs:

- CPU program (VTAM application program)
- Network Control Program (NCP)--This program resides in the communication controller.
- Function program or application program block (AP)--This program resides in the programmable control unit.

CICS/VS support of BTAM, EGAM, TCAM, and VTAM can co-reside in the CICS/VS partition/region.

Communications Controllers

VTAM uses the IBM 3704/3705 Communications Controllers to enable part of the telecommunications processing to be moved out of the central computer and into the network. The 3604/3705 control the flow of information between VTAM and terminals through use of a network control program (NCP). The 3704/3705 and its NCP support a variety of remote terminals. An NCP can be generated to handle lines in either network control mode (for VTAM-supported terminals), in emulation mode (for BTAM-supported terminals), or in both modes. An NCP generated with both functions is called an NCP with partitioned emulation programming (PEP) extension. This permits both VTAM- and BTAM-supported terminals to communicate with application programs (such as CICS/VS) through one 3704/3705.

Functions provided by the 3704/3705 include:

- line control
- dynamic buffering
- deleting and inserting line control characters
- translating character codes
- handling recoverable errors

- detecting permanent line errcrs
- gathering line statistics
- activating and deactivating lines and closing down the network

By performing these functions, the 3704/3705 and NCP conserve central computing resources. See *Introduction to the* IBM *3704 and 3705 Communications Controller*, GA27-3051, for additional information.

Shared Resources

VTAM permits its network resources to be shared among the various VTAM application programs being executed in separate partitions/regions in the CPU. One VTAM application program may be CICS/VS, which uses VTAM to establish communication between CICS/VS application programs and terminals, and another program could be IMS/VS operating in a different partition/regicn.

VTAM controls the use of paths through the 3704/3705, communication lines, and programmable controllers so that several applications may communicate with different terminals on a single line. Also, the terminals on the same line may communicate with any of the application programs using VTAM. Thus, one terminal on the line may be communicating with CICS/VS, while another terminal on the same line is communicating with IMS/VS. However, once a terminal begins to communicate with a VTAM application program, that terminal cannot communicate with another VTAM application program until the first program breaks its logical connection and releases the terminal. While connected to CICS/VS, the terminal may, of course, enter transactions to initiate different CICS/VS application programs.

For further information on VTAM, refer to *Introduction to VTAM*, GC27-6987, and *VTAM Concepts and Planning*, GC27-6998.

USE OF VTAM BY CICS/VS

CICS/VS uses VTAM to communicate with the following terminal systems:

- 3600 Finance Communication System
- 3650 Retail Store System
- 3790 Communication System

These terminal systems have "intelligent" capabilities and use a programmable contrcller tc direct the operation cf a number of attached terminals. The controller communicates with VTAM, which directs traffic between CICS/VS and the contrcller. The 3601, 3651, and 3791 are the controllers for the 3600, 3650, and 3790 systems respectively.

A logical connection is established between CICS/VS and a controller program. The contrcller program and its associated terminals, is called a logical unit. The VTAM connection with a logical unit is regarded by CICS/VS as similar to a physical connection with BTAM-supported terminals. The CICS/VS terminal control table (TCT) is used to define the status of each VTAM logical connection. A 4-character terminal identification is used to identify each particular TCT terminal entry, and hence identifies the logical unit.

CICS/VS SESSION TYPES

CICS/VS supports several different logical connections (called sessions) with the 3600, 3650, and 3790.

3600 Sessions

A 3600 session is established between CICS/VS and a 3600 application program block (AP) in the 3601 controller. The AP is regarded by CICS/VS as a logical unit, referred to by its terminal identification in the TCT. The AP, in turn, controls the operation of one or several of the following terminals attached to the 3601:

- IBM 3604 Keyboard Display
- IBM 3610 Document Printer
- IBM 3611 Passbook Printer
- IBM 3612 Passbook and Document Printer
- IBM 3618 Administrative Line Printer
- IBM 3614 Consumer Transaction Facility

The 3614 may also be attached directly to a 3704/3705 and is then regarded as a separate logical unit. As many as 96 terminals may be attached to a 3601. Disk storage in the 3601 contains 288,000 bytes of storage which permits multiprogrammed execution of several APs. Each AP may control several terminals. Individual terminals are transparent to CICS/VS and it is the responsibility of the AP to interpret specific requests from the terminals, communicate them to CICS/VS, interpret the output from CICS/VS, and direct it to the appropriate terminal.

CICS/VS communicates directly with an AP as a separate logical unit, and is unaware of the operation of other APs in the 3601. The other APs may be communicating with CICS/VS as other logical units, or with other VTAM application programs (such as IMS/VS).

3600 sessions permit the following CICS/VS services to be supported.

- Operator signon
- Basic mapping support
- Terminal paging
- Message routing and message switching
- Automatic task initiation
- Master terminal operator functions
- Supervisory terminal operator functions
- Message recovery and resynchronization

Refer to _Introducing the IBM 3600 Finance Communication System_, GA22-2764, for further information on 3600 units and features, and to the _CICS/VS Advanced Communication Guide_, SH20-9049, for information on the support of the 3600 by CICS/VS.

3650 Sessions

The 3651 Store Controller is a programmable control unit to which as many as 191 terminals may be attached for use in a retail store environment. The terminals are:

- IBM 3653 Point of Sale Terminal
- IBM 3275 Display Station
- IBM 3284 Printer
- IBM 3657 Ticket Unit
- IBM 3659 Remote Communications Unit

(Refer to IBM 3650 Retail Store System Introduction, GA27-3075, for further information on 3650 units and features.)

The 3657 and 3659 are not directly used by CICS/VS, but are used by the 3651. The 3651 controller contains up to 9.3 million bytes of disk storage and function programs (FPs) which control the operation of the various terminals attached to the 3651. CICS/VS communicates with the FP and is not aware of individual terminals. It is the responsibility of the FP to interpret specific requests from the terminals, select a relevant session type for communication to CICS/VS, interpret the output from CICS/VS, and direct it to the appropriate terminal. The different session types which may be selected are:

- 3275 host conversational session
- 3653 host conversational session
- Pipeline session
- Application program session

3275 Host Conversational Session: This session permits a 3275 to enter transactions to be transmitted to CICS/VS and to initiate CICS/VS application programs similar to transaction entry from BTAM-supported terminals. A number of 3275 host conversational sessions can be defined in the 3651 (less than or equal to the number of 3275s).

The 3275 terminal operator requests the 3651 controller to connect him to an available 3275 host conversational session. It is then the responsibility of the 3651 to establish the logical connection (session) between the 3275 and CICS/VS. The session is allocated an available terminal entry in the TCT and is known to CICS/VS by the relevant TCT terminal identification.

3275 host conversational sessions permit the following CICS/VS services to be supported.

- Operator signon
- Basic mapping support
- Terminal paging
- Message routing and message switching
- Automatic task initiation

- Master terminal cperatcr functicns
- Supervisory terminal cperator functions

The following service is not suppcrted:

- Message recovery and resynchronization

3275 host conversational sessions are the only 3650 sessions which permit BMS maps to be stored on disk in the 3651 instead of in the CPU. CICS/VS transmits the unformatted output data, plus the map name, to the 3651, which inserts device-dependent characters and, using the specification in the map, formats the output for display cn the 3275. See "Basic Mapping Support Ccmmunication using VTAM" for further details.

3653 Host Conversational Session: This session permits a 3653 to enter transactions to be transmitted tc CICS/VS, and to initiate CICS/VS application programs in a manner similar to that used for 3275 host conversational sessions. A number of 3653 hcst ccnversational sessions can be defined in the 3651 (less than cr equal tc the number of 3653s).

The 3653 terminal operator requests the 3651 stcre controller to connect him to an available 3653 hcst conversaticral session. This connecticn, and allocaticn of an available terminal entry in the TCT, is performed by the 3651 in a manner similar to that used for 3275 host conversational sessicns. The session is then identified to CICS/VS by the relevant TCT terminal identificaticn.

3653 host conversational sessions permit the fcllowing CICS/VS service to be supported:

- Basic mapping support

The following services are nct supported:

- Operator signon
- Automatic task initiation
- Terminal paging
- Message routing and message switching
- Master terminal cperatcr functicns
- Supervisory terminal cperator functions
- Message recovery and resynchronization

Pipeline Session: One pipeline sessicn may be established in a 3651 to support multiple 3653 terminals. The purpose cf this session is to support minimum delay transactions from 3653 terminals, such as a credit status check and update prior to initiating a particular customer transaction, or adjustment of a custcmer's credit status cn cancellation cf a credit transaction.

CICS/VS permits a number cf TCT entries to be specified as belonging to the pipeline session. These are used as a pocl of entries to permit multiple tasks to be initiated fcr different 3653s using the pipeline session. A separate pocl cf TCT entries can be specified for each 3651, or all TCT entries can be combined in a single pool to service all 3651 pipeline sessicns.

The TCT pools represent a number of user-specified tasks to be run. CICS/VS passes an input transaction from a pipeline session to one of the available TCT entries in the pool for that session. This permits the processing of a number of credit transactions to be multitasked for optimum response.

To identify the 3653 initiating pipeline transaction, the 3651 transmits a terminal identification to CICS/VS at the beginning of the input transaction. This is not used by CICS/VS, but is passed to the CICS/VS application program. The program may use the identification to maintain audit trails or statistics. The only change to data that can be made is to the status byte. All other data must be returned unchanged.

Pipeline sessions permit the following CICS/VS service to be supported:

- Basic mapping support

The following services are not supported:

- Operator signon
- Terminal paging
- Message routing and message switching
- Automatic task initiation
- Master terminal operator functions
- Supervisory terminal operator functions
- Message recovery and resynchronization

The following restrictions apply for pipeline sessions:

- CICS/VS logging is not performed.
- Only one input message and one output message are supported for each task.
- Each input message results in the initiation of a new task.
- A pipeline session can only be initiated by the 3651.
- A conversational transaction cannot use a pipeline session.
- The transaction code to initiate a task is defined in the TCT when it is generated. The input message is not examined for a transaction code.
- Unique operator identification is not associated with the TCT entry used by a task.
- Request volume that exceeds the user specified number of concurrent attached tasks is not queued within the system.

To meet the rapid response needed for credit transactions, the pipeline session provides fast throughput for a specific transaction type. In addition, the user can specify to the 3651 a time value at which the 3651 will use an alternate user-defined procedure for responding to these transactions.

Application Program Sessions: This session (also referred to as an Interpreter session) results in communication between CICS/VS application programs and specific application programs in the 3651. The application program session is primarily intended for the noninteractive data transfer of application-oriented information such as:

- Transaction log from 3651 disk to the CPU
- Inventory receipts file from 3651 to CPU
- Batch store messages and reports from the CPU to 3651
- Ticket data to be used in the preparation of magnetic stripe tickets on the 3657

In most cases, where CICS/VS is involved with application programs in the 3651, the logical terminal in the 3651 is disk. In some applications, disk serves as an intermediate or staging area between the CICS/VS application program and the ultimate destination. This usage is not detected by CICS/VS.

Application program sessions may be initiated either by CICS/VS or by the 3651 controller. When initiated by CICS/VS, the CICS/VS application program issues a terminal control PROGRAM macro instruction to identify the particular 3651 application program with which it wishes to communicate. This 3651 program may then exchange data with the CICS/VS program or may perform some function independent of CICS/VS as specified by the user.

Another possibility is to initiate the session from the 3651 controller. This type of session occurs as the result of a user-written program requesting (through the RCP interpreter) to establish a session with the host.

Application program sessions permit the following CICS/VS service to be supported:

- Basic mapping support

The following services are not supported:

- Operator signon
- Terminal paging
- Message routing and switching
- Automatic task initiation
- Master terminal operator functions
- Supervisory terminal operator functions
- Message recovery and resynchronization

Multiple input and output messages may be transmitted between CICS/VS and 3651 application programs.

See the CICS/VS Advanced Communication Guide, SH20-9049, for further information on the use of the 3650 by CICS/VS.

3790 Sessions

The 3791 Controller is a programmable control unit, to which up to 16 terminals and 2 line printers may be attached. It is used in a general-purpose data collection environment by many types of industries. The terminals are:

- IBM 3793 Keyboard Printer
- IBM 3277 Display Station
- IBM 2741 Communications Terminal
- IBM Line Printer
- IBM 3792 Auxiliary Control unit, remotely located from the 3791, to which may be attached 3793 or 3277 terminals, and one additional line printer.

The 3791 Controller contains a diskette and up to 27.5 million bytes of disk storage. Operation of the various terminals attached to the 3791 is controlled by the 3790 function programs.

A 3790 terminal operator can select an appropriate 3790 program to accept data entered at the terminal, edit it, and store it on disk for later transmission to the CPU. General-purpose data entry editing carried out by the 3790 program ensures that:

- Alphabetic fields contain only alphabetic characters, and numeric fields contain only numeric characters
- Fixed-length fields contain the required number of characters
- Variable-length fields do not violate the minimum or maximum length specified
- Required fields are not omitted
- Self-check digits are valid
- Numeric fields fall within a specified value range
- Field values are valid based upon application criteria such as a field's relationship to other fields, or to data held in tables or stored on 3791 disk data sets
- Field values are valid based upon the existence of required 3791 disk data set records, or the status of a particular data set record. For example, the 3790 program can ensure that an inventory update that reduces "quantity on hand" does not produce a negative quantity.

The 3790 can operate completely offline, accepting and editing data from terminals, and storing it on disk for later batch transmission to the CPU. It can also operate online to the CPU, establishing sessions between CICS/VS and 3790 FPs, which edit terminal input, pass input to CICS/VS for further processing, and accept output from CICS/VS to direct to the terminal. The 3790 is suited for remote offices where the functions of data entry, data inquiry, calculation, and document preparation are required.

Refer to An Introduction to the IBM 3790 Communication System, GA27-2767, for further information on 3790 units and features.

CICS/VS application programs communicate with 3790 function programs, and do not support or directly interact with terminals controlled by the 3790 programs. CICS/VS is not aware of any other function programs which may be concurrently executing in the 3790. These FPs may each separately establish sessions with CICS/VS, or with other VTAM application programs (such as IMS/VS).

3790 Host Inquiry Session: This session permits a 3790 FP to accept transactions from the terminals it controls and transmit those transactions to CICS/VS. Each host inquiry session is allocated the specific terminal identification of its TCT entry. Any number of CICS/VS transactions can be transmitted during a session, and each transaction can involve several input and output messages.

3790 host inquiry sessions permit the following CICS/VS services to be supported.

- Operator signon

The following services are not supported:

- Terminal paging
- Message routing and message switching
- Automatic task initiation
- Master terminal operator functions
- Supervisory terminal operator functions
- Message recovery and resynchronization
- Basic mapping support

The following restrictions apply to 3790 host inquiry sessions.

- The FP must initiate the session. The 3790 cannot accept unsolicited output from CICS/VS.
- The FP must start the exchange of data with a CICS/VS transaction by issuing a PUT, and must issue a GET to receive the last output from the CICS/VS transaction.
- The maximum input message is 240 bytes. However, several input messages can be transmitted by the FP and be concatenated by the CICS/VS application program (through user programming) for input greater than 240 characters.
- The FP is responsible for the unchaining of chained output.

Refer to the CICS/VS Advanced Communication Guide, SH20-9049, for further information on the use of the 3790 by CICS/VS.

ESTABLISHING CICS/VS SESSIONS WITH VTAM

Sessions between CICS/VS and VTAM can generally be initiated either from the CPU or from the logical unit. However, the 3790 must initiate a session with the CPU. It cannot accept a session initiated by the CPU.

CPU sessions are initiated by:

- CICS/VS automatically at CICS/VS initialization time

- VTAM system operator request
- CICS/VS master terminal cperator request
- CICS/VS to automatically initiate a task for a logical unit

Logical units initiate sessions by:

- the remote controller automatically at controller initialization time
- the remote controller on terminal operator request

When a connection is established, CICS/VS allocates the logical connection (and hence the logical unit) to a relevant TCT terminal entry. The logical unit may then transmit transactions to CICS/VS for processing. These transactions result in the initiation of CICS/VS application programs the same as for BTAM-supported terminals. Refer to the CICS/VS Advanced Communication Guide, SH20-9049, for further details on establishing connections.

CICS/VS application programs communicate with logical units by means of terminal control or basic mapping support.

TERMINAL CONTROL COMMUNICATION USING VTAM

Each of the basic terminal control operations, READ, GET, WRITE, PUT, WAIT, SAVE, and CONVERSE, is available for communication with VTAM-supported terminals. ERASE, when used to erase the display screen of a 3604 attached to a 3600, can only be specified using BMS.

Terminal I/O Overlap

VTAM permits terminal I/O operations to proceed concurrently with application program processing. This enables terminal I/O to be overlapped with CICS/VS application program processing. The CICS/VS application programmer can specify whether a terminal control request is to be initiated immediately to communicate with a VTAM-supported terminal while application program processing continues. The programmer can check for completion of the request at a later time by issuing the terminal control WAIT macro instruction.

Alternatively, the programmer may specify that a terminal control request is not to be initiated immediately, but is to be delayed until the program issues a terminal control WAIT macro instruction, passes through a user-defined synchronization point, or terminates. Delayed initiation of VTAM terminal control requests provides compatibility with the manner in which BTAM-supported terminals are controlled. Further, it can ensure that no output is transmitted to a logical unit until the task which generated that output is no longer vulnerable to backout in the event of a system failure. (See "Deferred Output Integrity" in Chapter 8.)

Full-Duplex Transmission

VTAM supports full-duplex transmission of data between the CPU and logical units. Thus, input and output may be proceeding concurrently on the same line, to different controllers multi-dropped on that line. CICS/VS enables the application program to request terminal input when needed, and to direct terminal output to the relevant terminal or logical unit when processed (half-duplex processing).

Although CICS/VS application programs process in a half-duplex mode, for optimum line efficiency data can still be transmitted by VTAM in full-duplex mode. Logical units may transmit input to CICS/VS application programs on an anticipatory basis. VTAM queues this input until the relevant CICS/VS application program issues a terminal control READ or GET macro instruction. The input message has already reached the CPU, and is then presented to the application program to satisfy the request.

Function Management Header

Output messages transmitted by terminal control to particular VTAM-supported terminals require certain information within the message to identify the disposition of the output by the logical unit. This information is called a function management header (FMH) and comprises the first part of the output message. It is 3 bytes long for the 3600, and is of variable length, 6 bytes or greater for the 3650. The FMH comprises at least a length byte, a message description byte, and a logical device code byte. When used for communication with a 3275 host conversational session in a 3650 it may also contain a BMS map name,

The message description byte indicates to the logical unit whether the output message was generated by CICS/VS itself or by a CICS/VS application program. It also identifies whether the output message has already been formatted (either by BMS or the CICS/VS application program) and contains device-dependent characters. The logical unit can deliver the formatted message to the relevant device with no further processing, if required.

The logical device code (LDC) in the FMH identifies the description of the output by the logical unit. (See "Basic Mapping Support Communication using VTAM" in this chapter, for additional information about LDCs.)

It is the responsibility of the CICS/VS application program to insert the appropriate FMH for the logical unit at the beginning of the output message. See the CICS/VS Advanced Communication Guide, and the CICS/VS Application Programmer's Reference Manual for further details.

System Programmer Macro Instructions

Additional terminal control macro instructions are available for use by the CICS/VS system programmer. These enable the status of a terminal to be changed in the TCT, or a specific terminal entry to be located in the TCT, using the terminal control STATUS and LOCATE macro instructions respectively. The result of these operations can be checked using the terminal control CHECK macro instruction.

The STATUS, LOCATE, and CHECK terminal control macro instructions are intended primarily to be used by the system programmer in the network error program (NEP). The NEP enables the system programmer to attempt recovery of error exception conditions encountered with transmission to VTAM-supported terminals.

See the CICS/VS Advanced Communication Guide and the CICS/VS System Programmer's Reference Manual, for further details on these system programmer macro instructions.

BASIC MAPPING SUPPORT COMMUNICATION WITH VTAM

The benefits of format (mapping) and terminal device independence offered by BMS to BTAM-supported terminals are also available to VTAM-supported terminals.

Input Mapping

BMS performs input mapping for the 3650 system (3653, 3270HC), but does not perform input mapping for the 3600 or 3790 system. The 3601 application program associated with the logical unit is responsible for removing device-dependent characters from the terminal input message, It is also responsible for formatting input data prior to its transmission to CICS/VS. BMS MAP macro instructions are ignored, and the input data is passed to the CICS/VS application program without change.

BMS IN macro instructions result in BMS issuing a terminal control GET macro for the application program. The received input data is not mapped and is passed to the application program without change.

The application program may use the built-in function INFORMAT macro instruction to locate delimiters inserted in the input message by the 3601. (See "Input Transaction Design" in this chapter for additional information.)

BMS Output Mapping

BMS performs output mapping for VTAM-supported terminals (except for 3790). Device-dependent characters are inserted into output messages by BMS based upon the characteristics of the device intended to receive the output. BMS constructs and inserts the function management header (FMH) into the output message prior to issuing terminal control output requests on behalf of the CICS/VS application program. The FMH has the same format as described in "Terminal Control Communications using VTAM." The message description byte is set up by BMS to define a formatted output message. The CICS/VS application program identifies the output device to BMS for VTAM-supported terminals by means of the logical device code (LDC). The LDC is used by BMS to identify the page size and device type. BMS inserts the LDC in the FMH prior to requesting terminal control output. The CICS/VS application program is relieved of the responsibility of constructing the FMH. This permits programs to be written independent of particular terminal characteristics.

Logical Device Code Uses

The LDC may be used to identify, to BMS and the APB, any 3600 device (except the 3614) attached to a 3600 work station with an appropriate page size. BMS uses the map specified by the application program to format the 3604 output data. It inserts the device-dependent characters into the output stream to ensure that the data is displayed as specified by the map. The LDC is also inserted into the output stream for transmission to the 3601. On receipt of the data, the 3601 application program determines (from the LDC) which device is to receive the output.

In addition to identifying specific devices and their associated page size, the LDC can also communicate other information to the controller application program. For example, the LDC may identify a specific preprinted form number to receive the output on a specific printer, or on any printer available to that logical unit. The LDC may also be interpreted by the logical unit as a request to turn on (or off) particular terminal indicator lights, transmit data accumulated on disk during offline operation to the CPU, or change processing modes (for example, change to after-hours processing operation).

The LDC is used by the CICS/VS application program to communicate logical disposition of output to the logical unit, and can represent any logical meaning useful to the installation's purpose.

CICS/VS permits the use of as many as 255 different logical device codes. Each may have a different meaning, dependent upon the particular controller application program interpretation.

Map Residence in Controllers

Some VTAM-supported controllers, such as the 3651, permit formats to reside outside the CPU on disk in the controller for 3275 host conversational sessions. A BMS output request from a CICS/VS application program results in transmission of the output data (without mapping) and the format name in the FMH, to the remote controller. The controller retrieves the specified format from the 3651 disk, and writes it to the screen on the relevant 3275 attached to the 3651. Thus, CPU processing is reduced, and additional flexibility is available to the installation to tailor a general purpose map to specific 3650 systems in individual retail stores.

BMS Alarm Indicator

The CICS/VS application program, using BMS, can request that an alarm indicator be turned on at the terminal upon receipt of the output message. This alarm indication is transmitted by BMS to the logical unit by means of the function management header (FMH). The logical unit responds to this request by turning on an appropriate indicator light on the terminal that is to receive the output.

BMS I/O Overlap

The CICS/VS application program can request that a BMS operation, and the associated terminal I/O, be initiated immediately. Alternatively, the program can request that the BMS operation be delayed until a WAIT macro instruction is executed, the program passes through a user-defined synchronization point, or the program terminates. This immediate, or delayed, request is specified as part of the BMS macro instruction in the manner described in "Terminal Control Communication using VTAM." It has the same purpose as for terminal control: to provide compatibility with BTAM operation and to improve output message integrity.

TERMINAL DEVICE INDEPENDENCE WITH VTAM AND BTAM

The use of BMS permits CICS/VS application programs to be written independent of the particular terminal to be used. For VTAM-supported terminals, default options provide compatibility with BTAM supported terminal operation. For example, the default option (unless specified otherwise) is to delay the initiation of terminal I/O until the application program issues a WAIT macro instruction, passes through a user-defined synchronization point, or terminates.

If a LDC is not specified in a BMS request to VTAM-supported terminals, a default value is used. This can be specified when the TCT is generated as a unique value for a specific TCT entry, for a group of entries, or for all entries in the TCT. If, however, a LDC is specified in a BMS request to BTAM-supported terminals, it is ignored.

BMS requests which specify 3270 attribute characterists can be used with VTAM supported terminals. In this case, the 3270 attribute information is ignored. The same function may be specified either in the particular map associated with the VTAM-supported terminal, or in the program in the remote controller which receives the output.

For these reasons CICS/VS application programs can be written to communicate with a variety of BTAM- and VTAM-supported terminals. Unique device characteristics may be specified in a map generated specifically for a terminal to achieve a general format function required by the CICS/VS application program. The CICS/VS application program identifies a map name in its BMS request; BMS appends to that map name a character which identifies the particular terminal type which is communicating with the program. BMS then retrieves the unique device-dependent map for that terminal type and uses it to format the terminal data.

Consequently, existing BTAM-supported terminals may be used to test CICS/VS application programs intended primarily for operation with VTAM-supported terminals before the installation of the VTAM terminals. Alternatively, sequential terminals, such as card reader / line printer, disk, or tape, may be used for testing. However, testing must model the operation of remote controller programs. For example, input must be presented to CICS/VS in exactly the same format as would be presented by the remote controller. Output must be accepted from CICS/VS as presented to the remote controller. In the case of input mapping for testing for the 3600, the mapped input from BTAM-supported or sequential terminals must be the same as presented by the 3601. This is necessary because mapping with actual 3600 input is ignored, and the input data is presented to the application program without change.

Testing is limited to those operations which can be performed by the relevant testing terminal, or which can be preplanned in the test input. BTAM-supported and sequential terminals are unable to exercise the same data handling and processing capability possible with remote controllers.

TERMINAL PAGING USING VTAM

Some sessions established with remote controllers support terminal paging. (See "CICS/VS Session Types.") The CICS/VS application program can build pages to be associated with specific logical device codes used by a logical unit. BMS separately controls the page construction for each LDC, and then makes available all pages built for each LDC used by the logical unit.

The terminal operator at the remote controller can request that pages associated with a particular LDC for that logical unit be displayed. (See Figure 3-3 for terminal paging commands.) The appropriate LDC pages desired can be requested by appending the LDC to the terminal paging command; all LDC pages can then be displayed on request. However, any LDC pages which have not been viewed will be lost when the terminal operator requests purging of pages associated with the logical unit.

MESSAGE ROUTING AND MESSAGE SWITCHING USING VTAM

Some sessions established with remote controllers support both message routing and message switching. (See "CICS/VS Session Types.") Pages, built by CICS/VS application programs or by terminal operators, can be associated with particular logical device codes for transmission to one, or a group, of logical units. The only restriction is that the same LDC be associated with all logical units in a message routing or switching request.

Pages delivered to the specified logical units can be viewed by the terminal operator using the appropriate LDC appended to the terminal paging commands as described for "Terminal Paging using VTAM" in this chapter.

The basic concepts and facilities of:

- Basic mapping support
- Terminal device independence
- Terminal paging
- Message routing
- Virtual Telecommunications Access Method (VTAM)

have now been outlined. The remainder of this chapter describes techniques for using these CICS/VS facilities in data communications design.

CONVERSATIONAL APPLICATIONS

The effectiveness of an online application depends to a large degree on man-machine communications. The computer is a tool used to achieve the objectives of the online application. To ensure success of online applications, the computer must provide the user with information to enable him to carry out his function effectively.

Data communication design represents the interface between the application and the machine. This is particularly true for conversational application design.

At all times during conversational message design, the system designer must keep in mind that the main objective of an online application is to assist the terminal operator. Thus, message formats should be designed to make the terminal operator's job easier. For example, input message formats generally should not be designed as fixed-format messages as for a card, but should enable the terminal operator to enter information in a variable-length format. CICS/VS can convert the variable-length input message into a fixed-length format for processing of the application program, as discussed below.

Also, if the task response time for a terminal operator is limited, the operator should be informed of the interval in which he is expected to respond.

TASK INITIATION

CICS/VS determines whether an input message received from a terminal satisfies an outstanding read request placed for that terminal by a currently executing program. If no application is currently active for the terminal which originated the input transaction, a task is initiated to process it.

Task initiation refers to the identification of a particular input transaction, the program to be used, and the creation of a task to process the transaction. Transaction identification can be achieved in several ways, as shown in Figure 3-5.

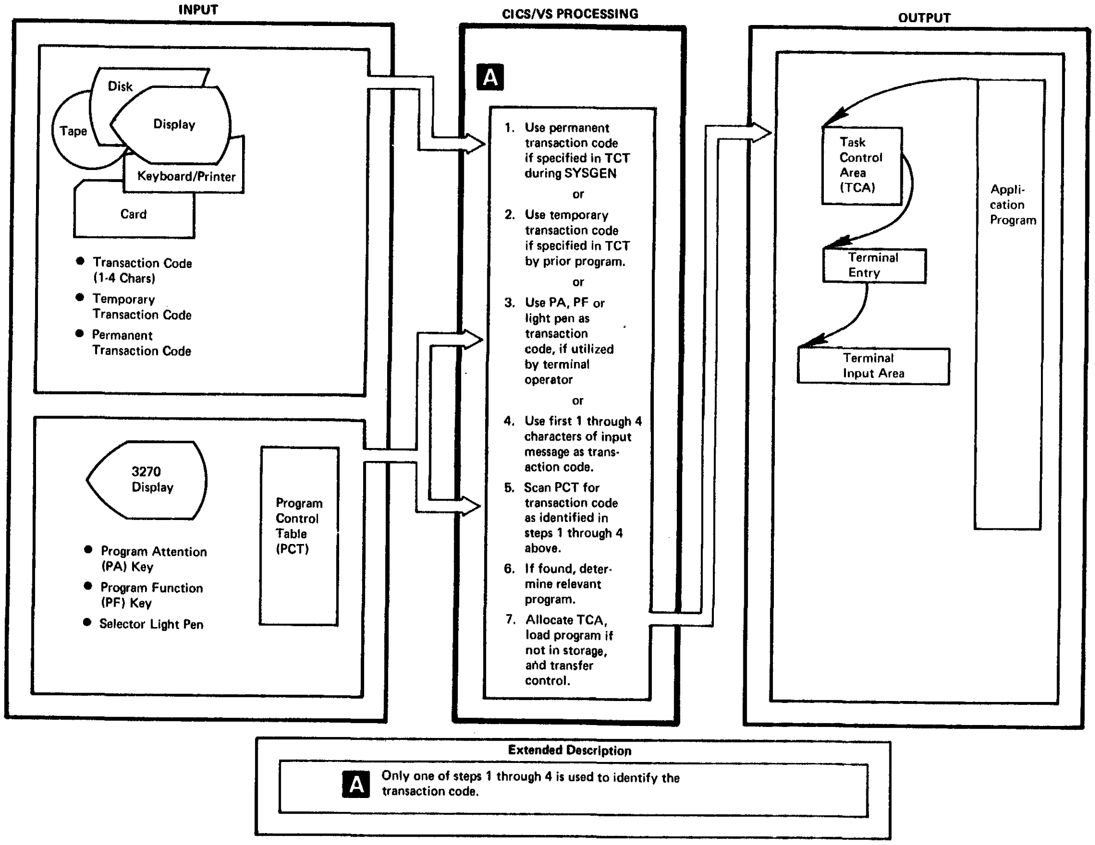

INPUT
CICS/VS PROCESSING
OUTPUT
Disk
Tape
Display
Keyboard/Printer
Card
● Transaction Code (1-4 Chars)
● Temporary Transaction Code
● Permanent Transaction Code
3270 Display
● Program Attention (PA) Key
● Program Function (PF) Key
● Selector Light Pen
Program Control Table (PCT)
A
1. Use permanent transaction code if specified in TCT during SYSGEN
or
2. Use temporary transaction code if specified in TCT by prior program.
or
3. Use PA, PF or light pen as transaction code, if utilized by terminal operator
or
4. Use first 1 through 4 characters of input message as transaction code.
5. Scan PCT for transaction code as identified in steps 1 through 4 above.
6. If found, determine relevant program.
7. Allocate TCA, load program if not in storage, and transfer control.
Task Control Area (TCA)
Terminal Entry
Terminal Input Area
Application Program
Extended Description
A Only one of steps 1 through 4 is used to identify the transaction code.

Figure 3-5. Task Initiation

Transaction Code

The first one to four characters of a terminal message, delimited by a defined character, are used as a transaction code. Valid transaction code delimiter characters are the field name start character or field separator character (both of which can be defined in the system initialization table), and any code with a hex value less than or equal to X'40'. The transaction code is used to search the program control table (PCT) to identify that transaction code. On locating the appropriate entry in the PCT with the same transaction code, the name of the program to be first used to process the transaction is obtained. CICS/VS then creates a task control area (TCA) to control the processing of the transaction by the program. The PCT can also identify the size of a transaction work area (TWA) to be appended to the TCA and used as a program work area during processing.

The program name identified in the PCT entry is located by CICS/VS using an address pointer in the PCT pointing to the relevant program entry in the processing program table (PPT). The PPT entry for that program indicates the language in which it was written (Assembler, COBOL, or PL/I), the size of the program in bytes, whether it is presently in CICS/VS address space and if so, the number of other tasks concurrently using it, and the location of the program on the CICS/VS program library on disk. If the program is not already in CICS/VS address space, it is loaded from the program library and control is passed to it to process the transaction.

3270 Attention ID Transaction Initiation

In the case of the 3270 Information Display System, each of the three Program Attention (PA) cr twelve Program Function (PF) keys or the selector light pen can be defined in the PCT tc initiate specified programs. By pressing the relevant PA or PF key, or by selecting a pen-detectable field with the selectcr light pen, the appropriate program is initiated. This is equivalent to entering a transaction code of from one to four characters.

The use of the selector light pen for transaction initiation is discussed in more detail in "Multiple Choice Format" later in this chapter.

Temporary Transaction Code

On completing the processing cf an input transaction, an application program optionally may identify the transacticn code to be used with the next input sent from that terminal. The next input need not be preceded by any transaction code, or PA or PF key, or be selected by the light pen.

This program-identified transaction code is referred to as a temporary transaction code, and is specified in the program control RETURN macro instructicn pricr tc termination of a task associated with that terminal. This temporary transaction code is used, with the next input frcm the terminal, to identify a program to be used to process that input. Any PA, PF, light pen, or transacticn code in the input is ignored. After use, the tempcrary transaction code is removed, and must be reestablished by a subsequent RETURN macro instruction, if it is tc be used with further input from the terminal. Therefore, an application program can transmit a response to a terminal requesting further informaticn frcm the operator. The next transaction code to be used is set by the prcgram so that, when the requested information is supplied by the operator, the program to process that information will automatically be initiated.

Permanent Transaction Code

A permanent transaction code can te defined for any CICS/VS terminal, at the time the CICS/VS terminal ccntrol table (TCT) is generated. This is particularly useful for those terminals which, by the nature of their device characteristics, are unable to start an input transaction with a valid transaction identification. In this case, every input message is initially passed to the same application program, which is related to the permanently defined transaction code for that terminal. This application program examines the input to determine the processing required, and identifies subsequent application programs which operate on that transaction. The permanent transaction code is used with any input message from a terminal which does not satisfy a pending read request issued ty a program. It also overrides any PA, PF, selector light pen, cr transacticn code used in that message. A temporary transaction code cannot be used with a terminal utilizing a permanent transaction code. Certain VTAM sessions established for the 3650 or 3790 require that a permanent transaction code be specified in the relevant TCT entries for the sessions. (See "CICS/VS Session Types" in this chapter.)

INPUT TRANSACTION DESIGN

The following design techniques for input messages may be used for terminals attached directly to CICS/VS or for terminals attached to programmable controllers such as the 3601, 3651, and 3791.

Fixed-Format Messages

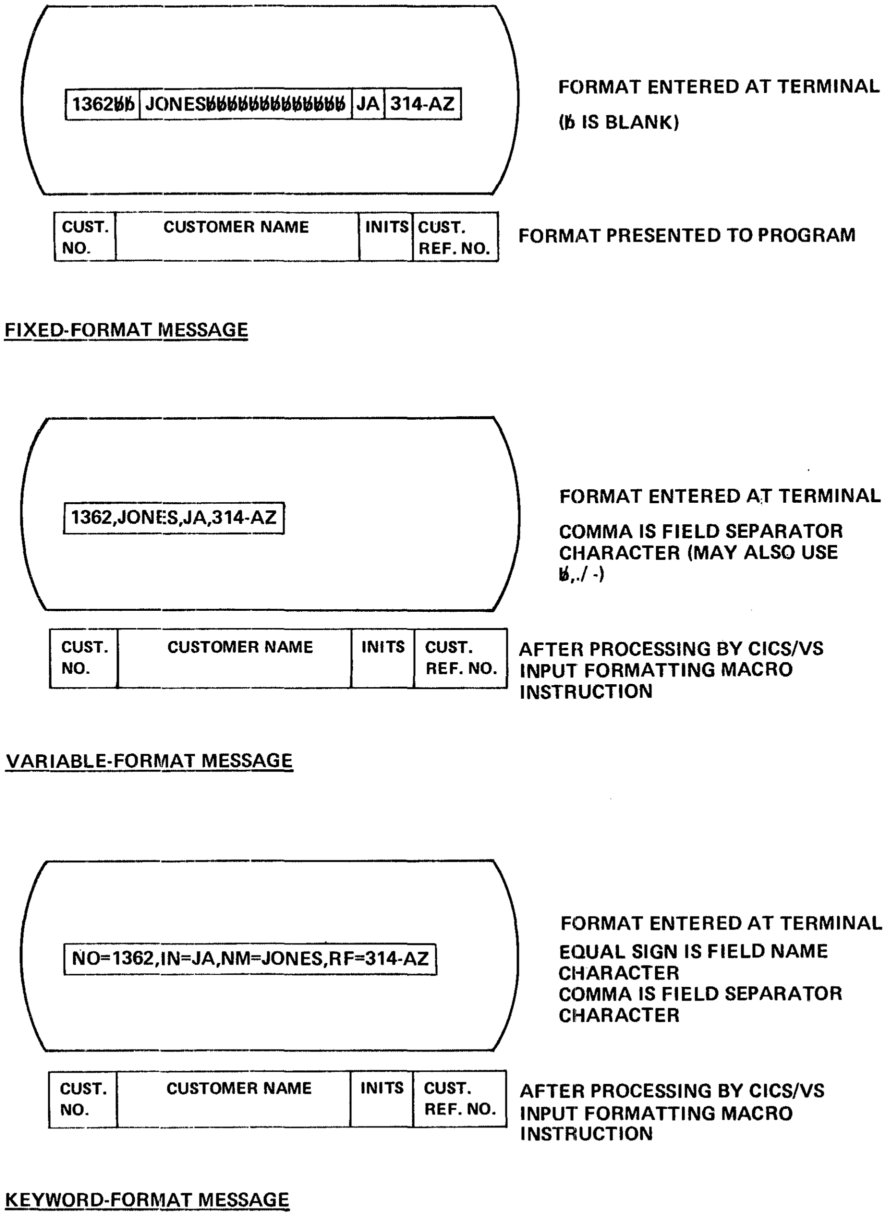
The fixed-format technique relates to the design of input messages such that each field of information occupies a fixed location in the input message (see Figure 3-6). While this is the normal technique for design of transactions entered from cards, it is not generally suitable for conversational applications. While a fixed message format makes it easy for the application program to extract information from the message for processing, this technique makes it more difficult for the terminal operator to enter that information, and is subject to operator error.

Variable-Format Messages

The variable-format technique is similar to the fixed-format technique previously described, except that required fields need not always be located in the same positions in the input message (see Figure 3-6). Fields are identified by their relative positions within the message as for fixed-format messages, but each field is separated from others by a delimiter character or characters. Possible delimiter characters are the blank, slash (/), equal (=), comma (,), or dash (-). Using delimiters, the terminal operator can enter information in the required sequence, without concern for the actual physical location of fields within the message.

CICS/VS provides a built-in function, which uses an input formatting macro instruction to convert this variable-format message into a fixed-format message for processing by the application program. Fields are located in the input message based upon the field separator character, which is a delimiter character defined by the user at CICS/VS system initialization time. These fields are inserted by CICS/VS in the appropriate locations in the converted fixed-format message based upon the requirements of the application program. Refer to the *CICS/VS Application Programmer's Reference Manual*, SH20-9003, for more details.

Consequently, the terminal operator can enter the required information in the specified sequence, without concern for the actual physical location of that information in the message transmitted to the CPU. The input message is converted to fixed format and presented to the application program as if it had been entered by the terminal operator as a fixed-format message. This enables the terminal operator to enter the input message in variable format suitable to him, yet be presented to the program in fixed format for easy processing.


1362₺₺ | JONES₺₺₺₺₺₺₺₺₺₺₺₺₺ | JA | 314-AZ
FORMAT ENTERED AT TERMINAL
(₺ IS BLANK)
CUST. NO.
CUSTOMER NAME
INITS
CUST. REF. NO.
FORMAT PRESENTED TO PROGRAM
1362,JONES,JA,314-AZ
FORMAT ENTERED AT TERMINAL
COMMA IS FIELD SEPARATOR CHARACTER (MAY ALSO USE ₺,./-)
CUST. NO.
CUSTOMER NAME
INITS
CUST. REF. NO.
AFTER PROCESSING BY CICS/VS INPUT FORMATTING MACRO INSTRUCTION
NO=1362,IN=JA,NM=JONES,RF=314-AZ
FORMAT ENTERED AT TERMINAL
EQUAL SIGN IS FIELD NAME CHARACTER
COMMA IS FIELD SEPARATOR CHARACTER
CUST. NO.
CUSTOMER NAME
INITS
CUST. REF. NO.
AFTER PROCESSING BY CICS/VS INPUT FORMATTING MACRO INSTRUCTION

FIXED-FORMAT MESSAGE

VARIABLE-FORMAT MESSAGE

KEYWORD-FORMAT MESSAGE

Figure 3-6. Fixed-, Variable-, and Keyword-Format Input Messages

This technique can be used with CICS/VS application programs designed to process input from the 3600 Finance Communication System. BMS does not map input data from a 3601 controller, but passes it to the application program without change. (See "Basic Mapping Support using VTAM.") Therefore, the 3601 Controller can format the input message prior to transmission to CICS/VS for processing by the CICS/VS application program by using the built-in function INFORMAT macro instruction. This formatting involves insertion (by the 3601 AP) of delimiter characters in a variable format message entered from terminals attached to the 3601. The same delimiter characters defined by the user at CICS/VS system initialization time should also be used by the appropriate 3601 APs which prepare the input for transmission to CICS/VS.

Keyword-Format Messages

This format is similar to the variable-format message described above, except that each field is preceded by a field name start character (defined at system initialization time) and a unique keyword. The keywords and fields can be variable format. Because each field is identified by its appropriate keyword, the sequence of fields in the input message may vary.

The terminal operator enters information in variable format, in the sequence which is best suited to his requirements. The CICS/VS input formatting macro instruction locates each field based on its keyword and positions that field in a fixed-format message presented to the application program. If a particular keyword is omitted, that field is left blank in the converted fixed-format message.

Both variable-format and keyword-format information can be included in an input message. The CICS/VS input formatting macro instruction can process both input techniques as part of the same message.

Figure 3-6 illustrates typical fixed-, variable-, and keyword-format input messages.

Keyword-format messages offer maximum flexibility to the terminal operator, not only in the positioning of information in the message, but also in the sequencing of information in the message. However, the information can still be presented to the application program in fixed format for processing.

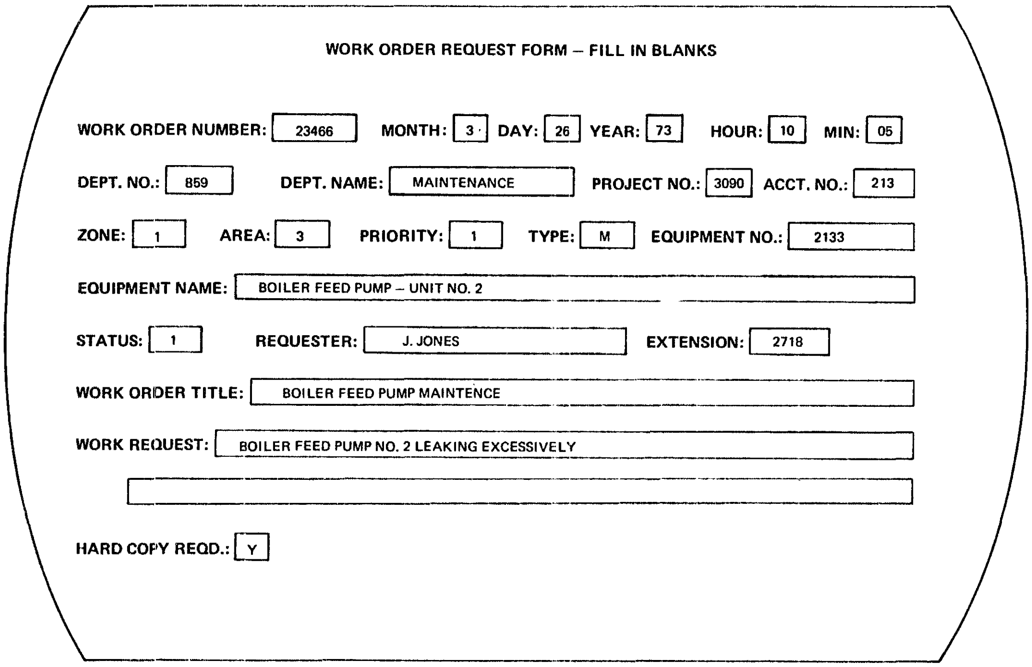
A disadvantage for the terminal operator, however, is that he must accurately key in additional information, namely, the keyword for each input field. This additional keying takes time and is vulnerable to error, although it provides positive identification of each field. Because this keyword format permits a number of input fields to be present or absent, depending upon the characteristics of the application, it could in some instances result in less keying than for variable-format messages.

The keyword format technique can also be used for input from a 3601 Controller, as described in "Variable Format Messages." The additional keying overheads of the keyword format technique are a consideration with input from 3601 controllers. The terminal operator can enter input to the 3601 in any convenient format. The 3601 AP can then insert the necessary keywords and delimiter characters before transmission to CICS/VS.

Fill-in-the-Blanks Format

This message format accommodates the inexperienced terminal operator. It involves the display of descriptive information identifying each field to be entered, as illustrated in Figure 3-7, and applies mainly to display terminals.

The most useful approach is to display an image of the information normally provided on the input documents used by the application. For example, an image of a product order form may be displayed for an order entry application. The terminal operator, using the description preceding each field, enters the required information. In the case of the 3270, only modified fields, such as that information entered from the keyborad, will be transmitted to the computer. The descriptive information is not transmitted, unless specified by the application program for identification purposes. Each input field transmitted from a 3270 is identified by its buffer address. This buffer address is



WORK ORDER REQUEST FORM – FILL IN BLANKS

WORK ORDER NUMBER: 23466 MONTH: 3 DAY: 26 YEAR: 73 HOUR: 10 MIN: 05

DEPT. NO.: 859 DEPT. NAME: MAINTENANCE PROJECT NO.: 3090 ACCT. NO.: 213

ZONE: 1 AREA: 3 PRIORITY: 1 TYPE: M EQUIPMENT NO.: 2133

EQUIPMENT NAME: BOILER FEED PUMP – UNIT NO. 2

STATUS: 1 REQUESTER: J. JONES EXTENSION: 2718

WORK ORDER TITLE: BOILER FEED PUMP MAINTENCE

WORK REQUEST: BOILER FEED PUMP NO. 2 LEAKING EXCESSIVELY

HARD COPY REQD.: Y

Figure 3-7. Fill-in-the-Blanks Input Message Format

used by basic mapping support (BMS), in conjunction with the input map defined for the transaction, to identify each input field and map the input message into a fixed-format message. Figure 3-7 illustrates a typical fill-in-the-blanks input message format.

An example of the use of this technique is found in the Display Management System II (Program numbers 5736-XC4 for DOS/VS, and 5736-XC4 for OS/VS). Refer to "Related Publications" in the Preface for relevant DMS II publications.

Multiple Choice Format

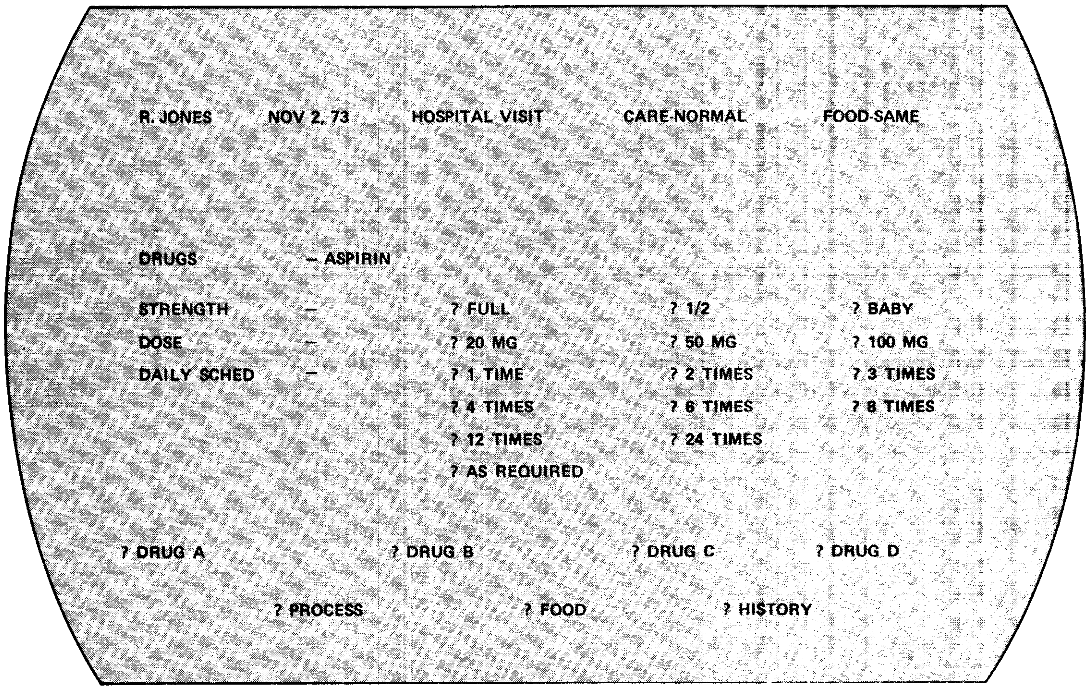
This format uses the optional selector light pen on the 3270 Information Display System and involves the display of a number of pen-detectable fields. These fields present a series of multiple choices, one or several of which can be selected by the terminal operator by placing the light pen to the pen-detectable fields to be selected.

The output response from a previous application program may define certain fields displayed on the 3270 screen as pen-detectable. Such pen-detectable fields are identified by a question mark, "greater than" (>) symbol, or blank character at the start of the field. A pen-detectable field with its first character blank is referred to as an "immediate" field.

The appropriate choices are made by the operator, by simply touching the light pen to the relevant fields. A question mark is changed to a "greater than" (>) sign to signify selection of that field. The greater-than character is changed back to a questicn mark if the field is selected by the light pen a second time, to indicate that the previous selection of that field is to be ignored. Selection of an immediate field (first character blank) results in the transmission of a message to the CPU. This message contains only the buffer addresses of fields selected by the pen and changed to a greater-than character.

The attention ID (AID) character transmitted from the terminal on selection of an immediate field is used to locate the PCT entry for immediate pen-detectable fields, and to transfer control to a common user-written program. This program examines the buffer addresses representing selected fields, and interprets these selections through the last BMS map used with that terminal. When designing pen-detectable screen formats, each screen format to be supported by this common program should be identified.

Another technique for multiple choice input is for the application program to list (or display) several choices, identifying each choice by number. The terminal operator may then select an appropriate response by keying in its identifying number.



```
R. JONES      NOV 2, 73      HOSPITAL VISIT      CARE-NORMAL       FOOD-SAME


DRUGS           - ASPIRIN

STRENGTH        -             ? FULL              ? 1/2            ? BABY
DOSE            -             ? 20 MG             ? 50 MG          ? 100 MG
DAILY SCHED     -             ? 1 TIME            ? 2 TIMES        ? 3 TIMES
                              ? 4 TIMES           ? 6 TIMES        ? 8 TIMES
                              ? 12 TIMES          ? 24 TIMES
                              ? AS REQUIRED

? DRUG A             ? DRUG B            ? DRUG C          ? DRUG D

          ? PROCESS           ? FOOD            ? HISTORY
```

Figure 3-8. Multiple Choice Input Message Format

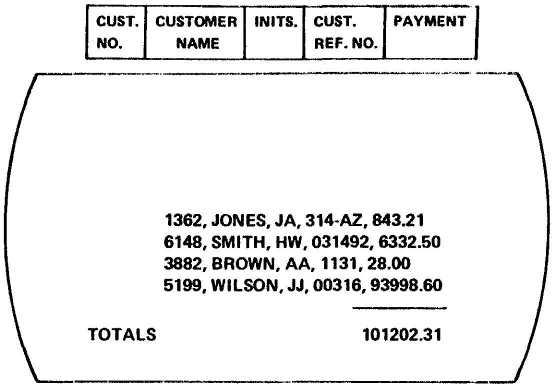
Selection of multiple choice fields can be used by unskilled terminal operators in user departments, to enter information for online applications. Figure 3-8 illustrates a typical multiple choice input message format using pen-detectable fields.

Any of these transaction formats may be used for terminals which communicate with CICS/VS either directly or through programmable controllers.

TRANSACTION EDITING

After defining the methods to be used for transaction initiation and designing the input message formats, the editing and validation to be performed on the message by the CPU or programmable controller must be defined.

With the combination of BMS and the CICS/VS input formatting macro instruction, the application program is presented with an input message in a defined fixed format. The editing to be done by the application program is application-dependent; Figure 3-9 suggests some of the techniques available. Some of these editing techniques are supported by CICS/VS built-in functions, while others must be supported by user-written routines.



CUST. NO.	CUSTOMER NAME	INITS.	CUST. REF. NO.	PAYMENT

1362, JONES, JA, 314-AZ, 843.21
6148, SMITH, HW, 031492, 6332.50
3882, BROWN, AA, 1131, 28.00
5199, WILSON, JJ, 00316, 93998.60

TOTALS 101202.31

TRANSACTION EDITING TECHNIQUES

- FIELD VERIFY/EDIT**
  - ALL ALPHABETIC (A–Z)
  - ALL NUMERIC (0–9)
  - ALL PACKED DECIMAL
- CHECK DIGIT
  - MODULUS 10
  - MODULUS 11
- HASH OR CONTROL TOTALS
- ZERO PROOF TOTALS
- LIMIT RANGE
- TABLE SEARCH**
  - BINARY
  - SEQUENTIAL
- REASONABLENESS CHECK
- DATA SET CHECK
- SIGHT VERIFICATION
- KEY VERIFICATION

**CICS/VS MACRO INSTRUCTIONS AVAILABLE FOR APPLICATION PROGRAM USE

Figure 3-9. Transaction Editing Techniques

Built-in functions are not supported by the subset option of CICS/DOS/VS. (See "CICS/DCS/VS Subset Option" in Chapter 7.)

Many of these techniques may be implemented by the user in programmable controllers to provide for detection and correction of invalid data before transmission to the CPU. Editing of data at the time of initial entry at the source permits: earlier detection of errors, more efficient data transmission, and reduced CPU processing. With the offline capability of BTAM supported programmable controllers such as the 3735, 3740, and VTAM supported programmable controllers such as the 3600, 3650, and 3790, data entry application availability is enhanced. Data may be edited and collected offline on disk storage for later transmission to CICS/VS. Only validated data need be transmitted to the CPU. This data may be transmitted at line speed, resulting in significant time and cost savings.

Field Verify/Edit

CICS/VS provides editing macro instructions as built-in functions to enable the contents of a data field to be verified as either alphabetic or numeric. On determining the contents of the data field, CICS/VS branches to the appropriate routine in the application program. Any field may be checked for the following:

- Entirely alphabetic (blanks, or A to Z)
- Entirely EBCDIC numeric (0 to 9)
- Entirely packed decimal (COMPUTATIONAL-3 in ANS COBOL or FIXED DECIMAL in PL/I)

If alphabetic characters have been entered into a data field that must be all numeric, the CICS/VS field verify function enables control to be passed to a user-specified routine. Usually an error message is sent to the terminal operator notifying him that nonnumeric data was entered in the particular field.

The CICS/VS field edit function allows the application program to present a field containing EBCDIC numbers intermixed with other values (for example, part number 119-445/B) to CICS/VS, and receive a result with all nonnumeric characters removed. The result can be in EBCDIC decimal format. In effect, this function is the reverse of editing (that is, a de-edit operation) that is performed on a field for output. See the *CICS/VS Application Programmer's Reference Manual*, SH20-9003, for additional information.

Check Digit

Numeric fields may be checked by user-written routines for validity, by means of a check digit appended to the end of the field. Modulus 10 or Modulus 11 check digit editing is provided by the user program to verify the correctness of a field or to identify errors. This technique can be used for an identification field such as a part number. When the number is first assigned, a user program computes the revelant check digit and appends it to the identification number. The check digit is then considered an integral part of the number, and can be used to check the accuracy of the entered number whenever that number is referenced.

Hash or Control Totals

Control totals can be developed by user programming of specified fields in a number of transactions. Control totals can also be developed for a batch of transactions and compared by the user program against similar control or hash totals developed manually prior to entry of the batch into the system. If the program-developed and manually developed totals do not agree, the particular error or errors can be located by comparing the information entered in that field with the origine original source information for each transaction in the batch.

Zero Proof Totals

Generally, zero proof totals operate on a number of data fields within one transaction. These fields are added and subtracted together according to the requirements of the application. A nonzero result indicates an error in one or several of the fields. This editing technique may be supported by user-written routines.

Limit Range

This technique checks that the value in a data field lies between certain application-dependent limits. A field lying outside those limits is identified as an error. The user is responsible for developing this editing support.

Table Search

This editing technique uses the data field contents to locate a similar entry in an application-dependent table. If the exact field contents cannot be located in the table, an error is indicated. A substitute value can be presented to the application program, if required.

CICS/VS provides a table search built-in function to assist application programs in utilizing tables for transaction editing or processing. Either a sequential or a binary table search may be specified. Refer to the *CICS/VS Application Programmer's Reference Manual*, SH20-9003, for more information.

Reasonableness Check

The program applies various logical tests (provided by user-written routines) to the contents of a data field, to determine the reasonableness of that information as related to the particular application. If the contents of the field have not met the application criteria, it is identified as having an error. For example, the program may examine a product number entered as part of an order entry input message, in relation to the quantity of that product ordered. The application may require that certain products only be ordered in particular quantities. An ordered quantity outside that defined for the product would then be regarded as an error.

Data Set Check

This editing technique is similar in concept to the table search technique previously described, but is far more extensive and comprehensive. It requires user-written routines to satisfy the unique application editing requirements. Information in the input transaction is used by the application program to access a data set related to that transaction. Information in that data set record is then used to validate other information in the transaction. For example, an application might require a customer's number and name to be entered. The customer number is used by the application program to access the relevant customer record, and the name in the record is compared with the name in the input transaction to determine that the ccrrect customer number was entered with the custcmer name.

Depending upon the disk capacity and capability of the programmable controller, application data sets or subset information may be stored on disk in a remcte ccntrcller. The 3601 controller permits up to 288,000 bytes of data to be stored on disk. The 3651 and 3791 controllers support as many as 9.3 million and 27.5 millicn bytes of disk storage respectively. This permits some data set validation of input to be carried out in the remcte controller before transmission to the CPU.

Sight Verification

Sight verification can be used by the terminal cperator in conjunction with data set checking as described abcve. In this instance, information from the input transaction is used to retrieve a relevant data set record. Infcrmaticn in that record is then transmitted back to the terminal fcr sight verification by the terminal cperator. For example, in an crder entry application, the customer number entered at the start cf an crder can be used to access the relevant customer record. The customer name and address are then displayed at the terminal for confirmation against the actual name and address cf the perscn placing the crder.

A less effective editing technique is the sight verification of keyed information against the source information, pricr to transmitting the keyed transacticn into the ccmputer. This technique is subject to error and is completely dependent upon the accuracy and conscientiousness of the terminal cperator.

Key Verification

Certain data fields cannot be edited by any of the techniques described above. An example of such data fields can be sales amounts relating to products, or dcllar amounts to be entered. While control or hash totals can be developed across a series cf transactions to identify an error, the applicaticn could require mcre complete checking than ccntrol totals alone. Key verification refers to the double keying of specified fields at different times. This must be supported by user-written routines. The first entry of the field is saved by the computer and compared against the second entry of that field at a later time. If both entries disagree, one or both of the fields are in error and correction is necessary.

Many of the editing techniques previously described can be implemented not cnly in CICS/VS, but also in programmable controllers such as the 3735, 3740, 3600, 3650, or 3790. The 3740 and 3790 are specifically designed for data entry and editing applications for many different industries.

These editing techniques may be integrated directly into CICS/VS applicaticn programs, cr may be carried out in a prior data entry step. This step may be accomplished offline for later batch transmission to CICS/VS. Alternatively, it may be carried out online to CICS/VS with terminals such as the 3270, by using Video/370 (Prcgram number 5736-RC3 under DOS/VS, or 5734-RC5 under OS/VS). Refer tc "Related Publications" in the Preface fcr relevant Videc/370 publications.

ERROR CORRECTION

This section identifies some techniques which may be utilized by the system design team for error correction. Identification (through editing) of a transaction error to be corrected by the terminal operator can either be made:

1) on the first occurrence of an error in the message, or

2) after the entire message has been edited and all errors have been detected.

In conversational applications, the terminal operator should generally be notified by the application program of any errors immediately after the input transaction has been edited. The error message should be concise and meaningful, and should identify the particular field or fields in error, the nature of the error, and the action required by the terminal operator. The operator should be given the opportunity to obtain more information describing the particular type of error detected if he needs it.

Error Message Contents

The following types of error messages may be used, depending upon the requirements of the application:

- Error number
- Error number and text
- Abbreviated text, with a user-written HELP facility

An error number enables the error to be uniquely identified. Additional information describing the cause of the error may be provided in Terminal Operating Procedures documentation for the application. The user-written HELP facility enables the terminal operator to obtain more detailed information (that would otherwise be included in an operating procedures manual), by a special inquiry requesting the computer to provide the necessary detail. This technique has the advantage of keeping the most current operating procedures available to all terminal operators. It reduces the user's cost of developing, distributing, and maintaining written information on operating procedures for terminal operators. However, it has the disadvantage that it is utilizing available computer resources to provide information which can alternatively be documented in an operating procedures manual.

CICS/VS-generated system error messages to be transmitted to the 3600 consist of only the CICS/VS error messages and numbers documented in the CICS/VS Messages and Codes Manual, SH20-9008. The 3601 AP must recognize the error numbers and insert the necessary text before transmission to the terminal operator. Additional information can be found in the CICS/VS Advanced Communication Guide, SH20-9049.

Error Message Documentation

The information which should be provided in documentation detailing an error includes:

- Error number and error message

- Cause of the errcr

- Operator correction

This documentation of errcr messages should be made available online or incorporated into the user's terminal operating procedures documentation. Other required dccumentation should include CICS/VS terminal operating procedures and CICS/VS-supplied terminal transactions and error messages. See the CICS/VS Messages and Codes Manual, SH20-9008, and the CICS/VS Terminal Cperator's Guide, SH20-9005 for additional information on error messages.

Error Field Correction

To use terminal operator time most effectively, the application should be so designed that the cperator is required to enter only the field or fields in error. The operator should nct be required to reenter the entire input transacticn.

For example, in the case cf the entry of new-business insurance policies that can approach 1000 characters in an input transaction, it would be unwise to require the entire 1000 characters be reentered if one field was in error.

However, in an crder entry application, the infcrmation entered for each line item ordered is generally cnly product number and quantity. Detection of an invalid product number could require the reentry not only of the correct product number, but also of the quantity. Figure 3-10 illustrates an error field correction procedure which may be utilized by applicaticn prcgrams.

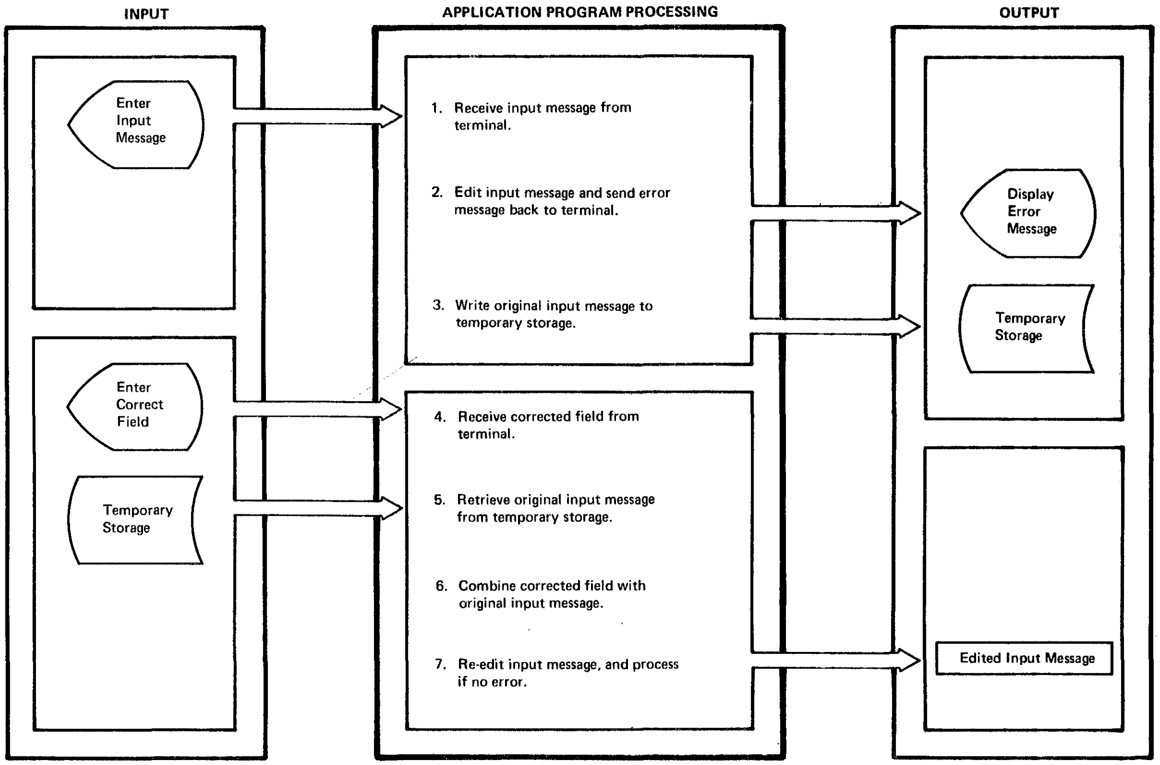

INPUT
APPLICATION PROGRAM PROCESSING
OUTPUT
Enter Input Message
1. Receive input message from terminal.
2. Edit input message and send error message back to terminal.
Display Error Message
3. Write original input message to temporary storage.
Temporary Storage
Enter Correct Field
4. Receive corrected field from terminal.
Temporary Storage
5. Retrieve original input message from temporary storage.
6. Combine corrected field with original input message.
7. Re-edit input message, and process if no error.
Edited Input Message

Figure 3-10. Error Field Correction

Use of Temporary Storage

If the terminal operator is required to input only the field in error, the application program must save the valid sections of the input transaction. Temporary storage enables application programs to save data either in dynamic storage or on disk, identifying the data uniquely for later retrieval by the same program or another program. As the terminal operator may take some time to enter the necessary correction, the valid part of the input transaction should normally be stored in temporary storage on disk, rather than in dynamic storage. Dynamic storage may then be utilized more efficiently for other purposes.

When transmitting an error message to a terminal, the application program may set a temporary transaction code (see "Task Initiation" in this chapter). Using this, when the corrected field is retransmitted from the terminal, a unique correction program may be initiated, based on the temporary transaction code, requiring no further action by the terminal operator other than correction of the field.

The user's correction program retrieves from temporary storage the transaction that was originally entered by the terminal operator. The corrected field is then inserted in place of the error field, and the entire transaction is reedited to determine whether the correction is valid, and that no other errors have been introduced. In the event of other errors being detected, further error messages are sent to the terminal operator.

The error correction process may be iterative until the input transaction has been completely validated. In the event that the terminal operator is unable to correct the transaction, he should be allowed to enter a unique code (such as "CANCEL") instead of the corrected field, indicating that this error transaction is to be ignored. The transaction will then need to be completely reentered at a later time.

OUTPUT FORMATTING

The actual format of output responses is application-dependent. However, a number of guidelines may prove useful here.

The output response is the main interface between the online application, under the control of the computer, and the terminal user. Accordingly, it should be easily read and understood by a typical user of that online application.

The amount of information that must be presented in response to an input request depends upon the requirements of that request. For example, an inquiry requesting display of a customer's current account balance is a request for specific information. However, a request for display of a customer's account details generally requires all information relating to that account.

Terminal Paging

Depending upon the particular terminal device being used, the amount of information to be displayed may exceed the physical capacity of that device. For example, a 3270 Model 1 displays 480 characters in 12 lines of 40 characters per line. The display of 15 lines of information requires that information be broken into two pages.

The use of terminal paging in CICS/VS enables considerable flexibility to be achieved in output formatting, regardless of the physical capacity of the terminal which will receive the output. This feature is available only for BMS-supported terminals. Refer to "Terminal Paging" for further detail.

BATCH APPLICATIONS

Batch applications are generally associated with high-speed data transmission terminals such as the 2770, 2780, 3600, 3650, 3735, 3740, and 3790, or computers used as terminals, such as the System/3 Models 6 and 10, the System/7, or the System/370.

In this environment, the emphasis is on the transmission of data from the terminal (or remote computer) to the central computer. Because of the nature of these devices, they are not designed for easy terminal operator interaction as is the case for conversational terminals. Generally, a batch of transactions is transmitted to the central computer, which processes that batch and then transmits any error messages back to the remote terminal or computer.

This online application environment is similar to the normal batch processing environment. In both cases, a batch of transactions is read and processed, and error messages are produced in an error list for offline correction.

This application approach is useful in an online environment where considerable amounts of information are to be transmitted across long distances. A high-speed batch terminal is able to transmit larger volumes of information than a conversational terminal, thus utilizing expensive long-distance transmission lines more economically. In this instance, the emphasis is on transmitting the data to the central computer as quickly and efficiently as possible, editing that data, and then transmitting any error messages back to the remote location quickly and economically.

ASYNCHRONOUS TRANSACTION PROCESSING

CICS/VS provides a function, called asynchronous transaction processing (ATP), which is designed for easy implementation of batch applications. ATP allows transactions, and the data associated with those transactions, to be transmitted in batches. Each batch is given a unique identification by the terminal operator. CICS/VS accepts each transaction from a batch terminal and delays its initiation until all specified input batches have been transmitted.

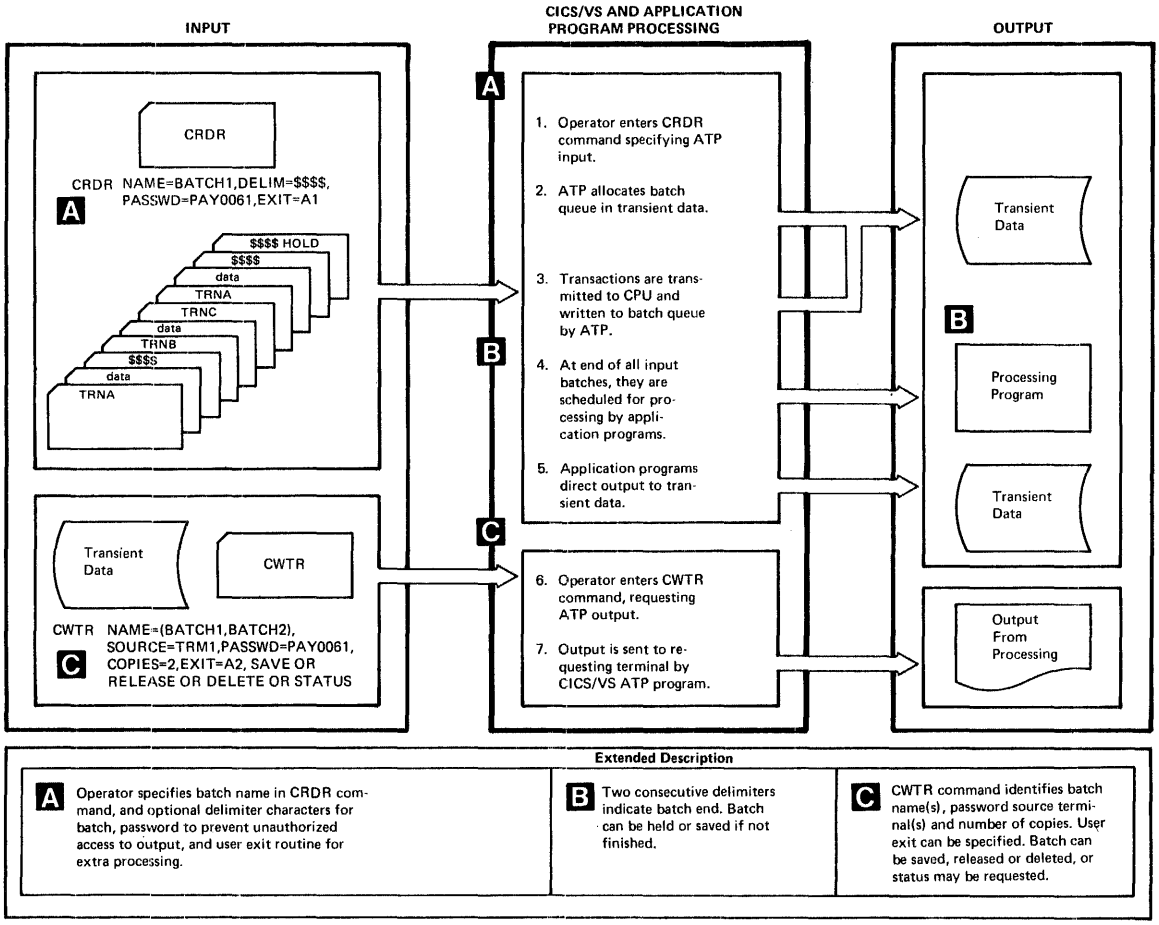
ATP requires that transient data intrapartition file support be generated as part of the user's CICS/VS system. This enables ATP to save batches of data for future processing and editing.

When all input batches have been transmitted, the transactions within the batch are then processed by application programs based upon their respective transaction codes. Any error messages are directed by the editing program to transient data for later transmission back to the terminal.

When the batches have completed processing, the terminal operator may then request that the output, if any, be sent to the terminal that originated the batch, or to a different terminal. Depending upon the amount of processing to be carried out on transmitted batches, the batch terminal may be disconnected from the transmission line by the user until output is available to be transmitted back to it.

The ATP facility is designed specifically for handling input from batch terminals such as the 3780, 2780, or 2770. Generally, ATP can be used from other interactive terminals (such as the 2741). However, ATP does not support input from 3270 or 2980 terminals. Also, application programs which intend to execute under control of ATP must not use the BMS terminal paging macro instructions. Figure 3-11 illustrates the use of the CRDR and CWTR ATP commands, which are used respectively for input and output of batched data.

The subset option of CICS/DOS/VS does not support ATP.


INPUT
CICS/VS AND APPLICATION PROGRAM PROCESSING
OUTPUT
CRDR
CRDR NAME=BATCH1,DELIM=$$$$, PASSWD=PAY0061,EXIT=A1
A
$$$$ HOLD
$$$$
data
TRNA
TRNC
data
TRNB
$$$$
data
TRNA
Transient Data
CWTR
CWTR NAME=(BATCH1,BATCH2), SOURCE=TRM1,PASSWD=PAY0061, COPIES=2,EXIT=A2, SAVE OR RELEASE OR DELETE OR STATUS
C
A
B
C
Transient Data
B
Processing Program
Transient Data
Output From Processing
Extended Description

1. Operator enters CRDR command specifying ATP input.
2. ATP allocates batch queue in transient data.
3. Transactions are transmitted to CPU and written to batch queue by ATP.
4. At end of all input batches, they are scheduled for processing by application programs.
5. Application programs direct output to transient data.
6. Operator enters CWTR command, requesting ATP output.
7. Output is sent to requesting terminal by CICS/VS ATP program.

A Operator specifies batch name in CRDR command, and optional delimiter characters for batch, password to prevent unauthorized access to output, and user exit routine for extra processing.

B Two consecutive delimiters indicate batch end. Batch can be held or saved if not finished.

C CWTR command identifies batch name(s), password source terminal(s) and number of copies. User exit can be specified. Batch can be saved, released or deleted, or status may be requested.

Figure 3-11. ATP Terminal Operator Commands

GENERAL BATCH PROCESSING

Some guidelines are presented below to assist in the design of batch applications when the design team does not wish to use ATP.

Execution of a batch application is, by its nature, of long-term duration. Accordingly, any storage required in executing batch application programs will be in use for a relatively long time, compared to conversational applications. Depending upon the amount of dynamic storage available for both batch and conversational applications, this requirement for long-term storage may affect the performance of conversational applications. To minimize the amount of storage used by batch programs, the following approach may be considered.

Ideally, an application program should be written to accept batch input transactions from the terminal, and queue these transactions on transient data. At the completion of each batch, the last transaction may automatically initiate a user program to validate and process the queued batch. In the meantime, the remote terminal is freed to allow further data input. On completion of batch processing, any error messages which were queued on transient data to send back to the remote batch terminal may be either automatically transmitted back as soon as that terminal is idle, or transmitted on request by the remote terminal.

This approach is similar to that adopted by ATP as described previously. It offers the principal advantage of very efficient utilization of data transmission lines, and the overlapping of processing one batch with the transmission of the next batch to be processed. On the other hand, ATP enables all input batches to be transmitted to the CPU, and then allows the user to disconnect the batch terminal from the transmission line until all of those batches are processed. At that time, the user or the CPU may reestablish connection between the terminal and CPU for output transmission.

This ATP approach is particularly economical when the processing time for all batches is longer than the input transmission time.

An alternative approach that can be used involves the batch application program reading a transaction, immediately following which the input transaction received is edited and error messages are queued on transient data for later transmission. However, this approach suffers from the disadvantage of less efficient data transmission. Data is transmitted from the remote terminal, followed by a pause for processing. Then the next transaction is transmitted and processed, with the line again being idle while the second transaction is being processed. No overlap of processing with data transmission is possible.

Either the first method described above, or the use of ATP, is recommended for most efficient and economic line utilization.

TERMINAL ERROR RECOVERY

CICS/VS uses BTAM, BGAM, TCAM, or VTAM for the control of terminals. These telecommunications access methods detect transmission errors between the central computer and a remote terminal, and automatically invoke error recovery procedures, if specified. These error recovery procedures generally involve the retransmission of data a defined number of times, or until that data is transmitted error-free. In the event that the error is not corrected after the specified number of retries, CICS/VS passes information connected with the error to the terminal abnormal condition program (BTAM-supported terminals) or to the node abnormal condition program (VTAM-supported terminals) for additional processing.

TERMINAL ABNORMAL CONDITION PROGRAM (TACP)

The TACP is used by BTAM-supported terminal. After determining that the error is unrecoverable, the TACP sets default actions based on keeping the network live. These may involve:

- Setting the terminal out-of-service

- Setting the line out-of-service

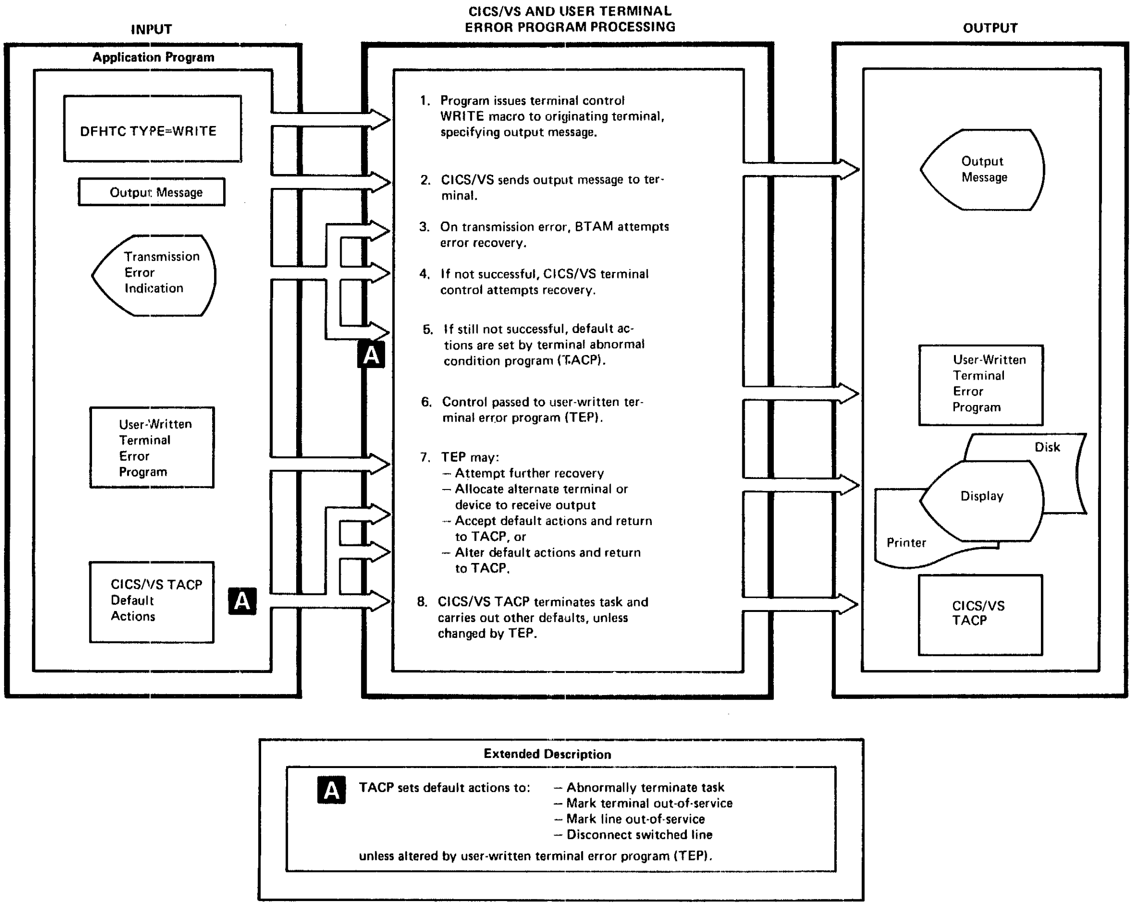
- Abnormally terminating the transaction

- Disconnecting a switched line

Before these default actions are taken, CICS/VS passes control to a user-supplied terminal error program (TEP) for application-dependent action if necessary (see Figure 3-12). On return from the terminal error program, TACP performs the indicated action as previously set by TACP or as altered by the TEP.

CICS/VS provides a sample TEP, which can be used to generate a specific TEP to meet the user's terminal error recovery requirements. A generated example of a TEP is supplied as part of the subset option of CICS/DOS/VS. (See the Subset User's Guide (DOS) for additional information.) This TEP can be used without change or as an example when developing a unique user-written TEP.

Generation of a sample TEP is described in CICS/VS System Programmer's Reference Manual, SH20-9004.


INPUT
Application Program
DFHTC TYPE=WRITE
Output Message
Transmission Error Indication
User-Written Terminal Error Program
CICS/VS TACP Default Actions
A
CICS/VS AND USER TERMINAL ERROR PROGRAM PROCESSING
A
OUTPUT
Output Message
User-Written Terminal Error Program
Disk
Display
Printer
CICS/VS TACP
Extended Description

1. Program issues terminal control WRITE macro to originating terminal, specifying output message.
2. CICS/VS sends output message to terminal.
3. On transmission error, BTAM attempts error recovery.
4. If not successful, CICS/VS terminal control attempts recovery.
5. If still not successful, default actions are set by terminal abnormal condition program (TACP).
6. Control passed to user-written terminal error program (TEP).
7. TEP may:
   - Attempt further recovery
   - Allocate alternate terminal or device to receive output
   - Accept default actions and return to TACP, or
   - Alter default actions and return to TACP.
8. CICS/VS TACP terminates task and carries out other defaults, unless changed by TEP.

A TACP sets default actions to:
- Abnormally terminate task
- Mark terminal out-of-service
- Mark line out-of-service
- Disconnect switched line

unless altered by user-written terminal error program (TEP).

Figure 3-12. CICS/VS Terminal Error Recovery

TERMINAL ERROR PROGRAM

The terminal error program may be supplied by the user to attempt further error recovery, if necessary. Alternatively, a sample TEP can be generated or the CICS/DOS/VS subset option TEP may be utilized. (See CICS/VS System Programmer's Reference Manual, SH20-9004.) For example, a user-written TEP can specify additional retries to be carried out by CICS/VS before the error is considered completely unrecoverable.

Alternatively, the user-written TEP can request that the output message be queued on disk using CICS/VS transient data, to be automatically transmitted to the error terminal when the problem has been rectified.

The user-written terminal error program might specify that the error terminal and line are not to be marked out-of-service, a switched line is not to be disconnected, or the task is not to be abnormally terminated. On return from the TEP, the task may be reactivated as if the error had not occurred.

This may be a satisfactory solution, if transmission of the output message is not critical to the application, but continued processing of the task is. For example, it may be necessary to allow the task to continue processing to enable various data sets to be completely processed and updated. Alternatively, the task may be allowed to abnormally terminate, and a program control SETXIT routine provided by the user may be utilized to complete urgent processing for the task (see "Program Error Recovery" in Chapter 2).

Generally however, all processing associated with a transaction and task, and updating of relevant data sets, should be completed before the programmer makes any attempt to transmit an output message to the terminal. This can be ensured on VTAM-supported terminals by specifying that transmission be delayed until a terminal control WAIT is issued, the program passes through a user synchronization point, or terminates. This is also the standard method used for BTAM-supported terminals. Receipt of an output message at the terminal should be regarded as an indication that all of the processing for the particular input transaction has been completed successfully.

The error recovery procedures described above for the terminal error program are discussed in more detail in the sections "Terminal Backup" and "Dynamic Terminal Reconfiguration" in Chapter 4.

NODE ABNORMAL CONDITION PROGRAM (NACP)

The NACP is used for VTAM-supported terminals to process abnormal situations associated with logical units. Information concerning the processing state of a logical unit is contained in the relevant TCT terminal entry, and in the VTAM request parameter list (RPL). There is no accompanying line entry as there is for BTAM-supported terminals.

NACP is scheduled any time a VTAM request made by CICS/VS completes in error or cannot be honored. The receipt of a negative response sent by a logical unit also causes NACP to be scheduled. This permits analysis of the sense information and issuance of any appropriate messages.

Whenever NACP is scheduled, its analysis routines determine the actions that are mandatory to the recovery procedure. Prior to performing these actions, NACP links to the user-written node error program (NEP).

NODE ERROR PROGRAM (NEP)

The user is responsible for coding an NEP for VTAM-supported terminals. To aid the user, certain optional actions are generated in the NACP. (For example, retry of a message.) If the user wishes any of these actions to be performed, he can set the relevant optional action codes in the TCT during NEP processing.

The user can issue VTAM responses or commands in the NEP. (See "Terminal Control Using VTAM.") The user can also issue VTAM responses or commands from remote programmable controllers. For example, if a printer on a programmable controller runs out of paper, the user may code the controller to send a negative response to the CPU, specifying a relevant user sense code. This will cause NACP (and NEP) to be scheduled in the CPU. The NEP can then quiesce the logical unit using that printer, until the paper supply is replenished. Refer to the *CICS/VS Advanced Communication Guide* for additional information.

MESSAGE LOGGING

Input and output messages may be automatically logged by CICS/VS for message recovery and resynchronization. In the event of loss of contact with VTAM-supported terminals, logging and recovery protect message integrity. Transactions requiring message integrity are specified in the PCT. The programmable controllers should also log (as a minimum requirement) the VTAM sequence numbers of protected tasks.

For performance reasons, transactions that do not change the system environment (such as inquiries that do not update data sets) should not specify message integrity.

In the event of system failure, CICS/VS emergency restart identifies in-flight tasks and backs out in-flight task activity. The input message for an in-flight-protected task can be used during emergency restart to establish message resynchronization with the controller. This is also true for a committed output message for which a positive indication of receipt was not received by the CPU before system failure. (See "Transaction Recovery" in Chapter 8.)

SECURITY DESIGN

The main objective of an online application is to make timely, complete, and accurate information available to the people who need it. The availability of up-to-the-minute information will help maintain control over online applications, or facilitate changes which could be made only at considerable time and expense. Applications should be responsive to the needs of the application user, and should provide improved service to the company's customers.

However, responsiveness and ready availability of information can also be disadvantages if that availability is not controlled. Information accessible online should only be made available to those people who are authorized to use it. Thus:

- Only manufacturing personnel may inquire into or change manufacturing work orders

- Only bank tellers or authorized personnel may initiate savings or loan transactions

- Only authorized personnel may inquire into a customer information system for banking, insurance, or utilities

- Only authorized doctors or medical staff may inquire into a patient's history

- Only authorized police personnel may inquire into a police information system

- Only authorized perscnnel may place orders in the pharmaceutical or distribution industries

Regardless of how effective an cnline applicaticn is, if it does not have security provisicns to prevent unauthorized access to or abuse of information, the consequences can be far-reaching.

CICS/VS CPERATOR SECURITY

CICS/VS provides an cptioral cperatcr security facility. Each terminal operator is identified to CICS/VS in an operator signon table (SNT). The following information is contained in the table:

- Operatcr name
- Operator initials
- Operator password
- Operator security codes
- Operator security class
- Operatcr pricrity

Each terminal operator is required to signon to CICS/VS at a terminal, by entering the signon transaction code CSSN, together with his allocated 4-character password and his name, up to 20 characters in length (see Figure 3-13). Cperatcr security is also discussed in CICS/VS Terminal Operator's Guide, SH20-9005.

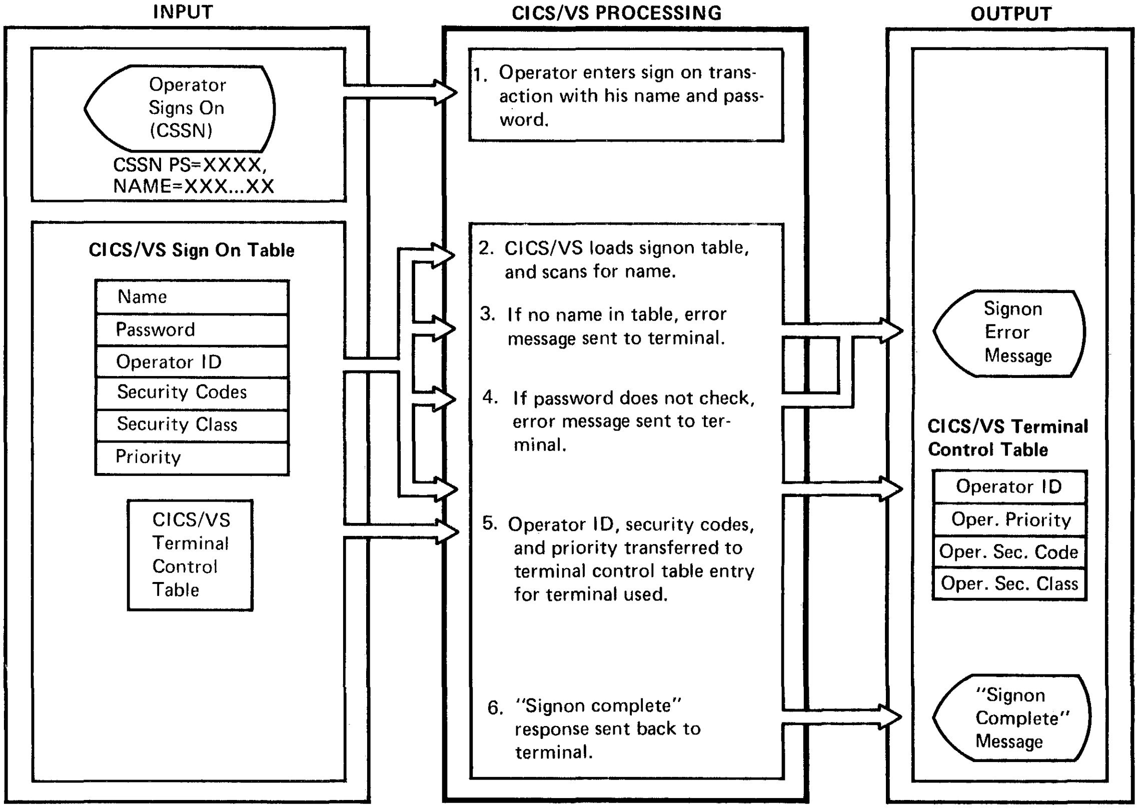

INPUT
CICS/VS PROCESSING
OUTPUT
Operator Signs On (CSSN)
CSSN PS=XXXX, NAME=XXX...XX
CICS/VS Sign On Table
Name
Password
Operator ID
Security Codes
Security Class
Priority
CICS/VS Terminal Control Table
1. Operator enters sign on transaction with his name and password.
2. CICS/VS loads signon table, and scans for name.
3. If no name in table, error message sent to terminal.
4. If password does not check, error message sent to terminal.
5. Operator ID, security codes, and priority transferred to terminal control table entry for terminal used.
6. "Signon complete" response sent back to terminal.
Signon Error Message
CICS/VS Terminal Control Table
Operator ID
Oper. Priority
Oper. Sec. Code
Oper. Sec. Class
"Signon Complete" Message

Figure 3-13. Operator Signon

The CSSN transaction code initiates the CICS/VS signon program (SNP). This program loads the signon table (SNT) and locates the operator name and password in the table. If these two do not agree exactly, the operator is prevented from entering further transactions until he signs on successfully.

Once signon is achieved, the signon program extracts the operator identification (for example, his initials), security codes, and class and operator priority from the signon table. This information is transferred to the terminal control table (TCT) entry for the physical terminal to which he has signed on. This information remains in the TCT entry until the operator signs off with a CSSF transaction.

The three-character operator identification is used for subsequent operator identification, and the operator priority is used in conjunction with terminal and transaction priorities to establish the overall task priority. This is discussed in more detail in "Priority Processing" later in this chapter.

The operator security codes consist of a series of numbers ranging from 1 to 24. The function of these security codes is defined by the user, but conventionally security code 1 implies low security while security code 24 implies high security.

These security codes are used in conjunction with a security code defined for each application transaction code. A transaction code with a defined security code of 10, say, can be used only by those operators who also have a security code of 10. An operator may have more than one security code. Operator security codes 5, 6, 10, and 12, for example, would enable those operators to use only those transaction codes which also have been defined as security 5, 6, 10, and 12.

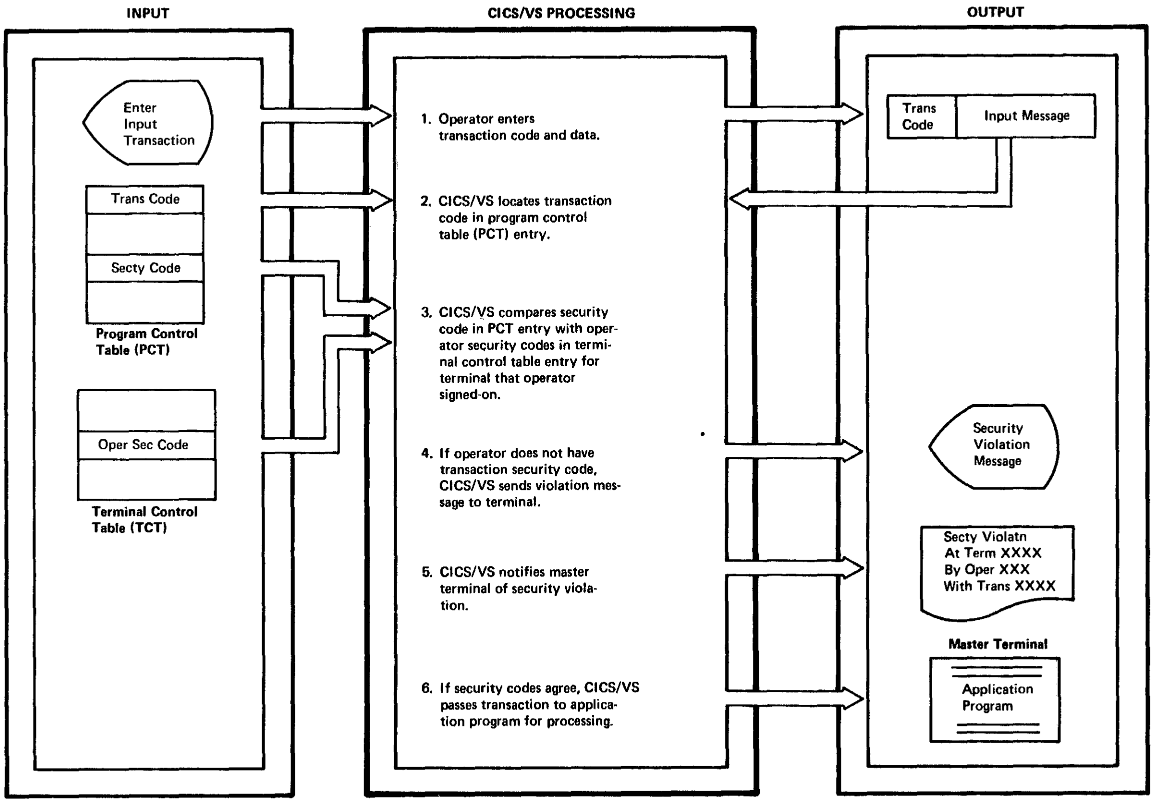
The power of the CICS/VS operator security lies in the way the system designer defines the relevant transaction security codes for the application. For example, in an inquiry system, low security transactions may be given a security code of 1, which allows any operator to use that transaction code. However, only those operators who are authorized to make certain other high security inquiries are given the same security code as allocated to those inquiry transactions.

All operators may be allowed to see general information following an inquiry, while only authorized operators are presented additional information based on their security codes. This may be achieved by having two versions of the inquiry program: one which displays limited amounts of information, and another which displays the full information. The limited information program may be given a security code of 1, for example, while the more detailed information inquiry transaction code may be given a different security code.

If an operator attempts to enter an unauthorized transaction code, CICS/VS will reject the transaction and send an error message indicating a security violation to the terminal operator. The master terminal operator is also notified by CICS/VS of the attempt to enter an unauthorized transaction code. The operator identification, terminal identification, and transaction code used are detailed in the notification message to the master terminal, as shown in Figure 3-14. (See the *CICS/VS Messages and Codes Manual*, SH20-9008, for additional information.) The master terminal operator may then take appropriate action.

The operator security class is used primarily in conjunction with the CICS/VS message routing facility. Messages may be directed to specific terminals, specific operators, or all operators with a specific security class. An operator may have more than one security class.

Messages directed to specific operators, or to specific operator classes, are nct transmitted until the particular cperator or operators sign on to CICS/VS. Refer to "Message Routing" in this chapter for more detail.



1. Operator enters transaction code and data.
2. CICS/VS locates transaction code in program control table (PCT) entry.
3. CICS/VS compares security code in PCT entry with operator security codes in terminal control table entry for terminal that operator signed-on.
4. If operator does not have transaction security code, CICS/VS sends violation message to terminal.
5. CICS/VS notifies master terminal of security violation.
6. If security codes agree, CICS/VS passes transaction to application program for processing.

Figure 3-14. CICS/VS Operatcr Security

SECURITY ENHANCEMENTS

The CICS/VS operator security feature relies cn an operator's name and his knowledge of a unique password to allow him to signon. Once he has signed on, he has full access to all transaction codes which he is authorized to use.

However, a password is like the combinaticn to a safe. It is effective when it is known only ty those persons authorized to use it. To avoid the possibility cf tnauthcrized persons learning the signon procedure and an operator name and relevant passwcrd, the design team may incorporate some security enhancements into their system design if required by the application.

The security enhancements which may be developed depend upon the particular application requirements and the cost cf providing that security in time and effort, as well as the potential cost to the organization if that additional security is not provided.

The following techniques are suggested user enhancements which may be considered as part of the system design, and which could be readily implemented by user-written coding in the application programs. These enhancements build upon the CICS/VS security features and provide increasing degrees of security with each technique discussed. They may be implemented within CICS/VS application programs or in application programs written for programmable controllers. Implementation of these security enhancements in programmable controllers permits authorization to be performed before transmission to the CPU, and enables security checks to be carried out based on each remote location's requirements.

Physical Terminal Security

This provides security on the basis of authorized operators entering transactions only from authorized physical terminals. Normally a terminal operator may sign-on to any terminal supported by CICS/VS. This includes conversational and batch terminals, together with simulated terminals such as card reader, tape, or disk.

On initiation of a task, CICS/VS makes the terminal identification available to the user's application program. For security purposes, the CICS/VS (or programmable controller) application program may check this terminal identification against a user-supplied table of authorized terminal identifications. If the terminal is unauthorized, the transaction can be rejected by the application program, together with an error message. The program may also notify the master terminal operator.

Function Password or Security Code

For a transaction entered by an authorized operator using an authorized terminal, the system designer may require the terminal operator to provide an additional password (or security code) to the user program to permit access to high security functions or information. This additional password may be provided in the main body of the transaction when it is first entered, or be explicitly requested by the application program when it reaches that point in its execution.

Data Set Passwords

This security technique requires the terminal operator to supply a password to the user program before a specific data set or data base can be accessed. A data record password may additionally require a specific password to be supplied by the operator to the program before information in particular records is displayed. This password may be incorporated as part of the record. A data field password is an extension of data record passwords, and requires a password to be provided before specific fields can be displayed for the operator.

Dynamic Password

This applies to all of the passwords described above, and requires that a password be changed frequently by the user to prevent unauthorized persons gaining knowledge of it.

For maintenance purposes, the current password is best recorded on disk, and is changed on disk by a specific transaction. This reduces the need to modify programs, but introduces the requirement that access to this password data set be strictly controlled both in the online environment and in the batch environment. To guard against possible unauthorized access of this data set, the passwords may be recorded by the user in a coded (scrambled) form on the data set; this code is unintelligible and useless unless it is translated using, for example, a unique translaticn table in the application prcgram.

This translation table can also be changed dynamically by the user, if required, to further reduce the possibility of unauthcrized access to the password data set and the scrambled passwcrds.

Dynamic Operator Passwords

Support of dynamic operatcr passwords is achieved by periodic regeneration of the signcn table (SNT). An alternative which is also equally effective is dynamic passwords as described previously.

The extent of security precauticns is limited only by the imagination of the design team. However, the application requirements will generally dictate the point at which security prccedures should stop.

A battery of locks on a door is useless if the person authcrized to open that docr dces not have all cf the keys. In the same way, the use of dynamic passwords may prevent even authorized access if the person attempting that access forgets the current password and procedures. Furthermore, the security precautions adopted by the user may be sc stringent as tc prevent him frcm ever finding out those passwords and procedures.

OPERATOR ERROR STATISTICS

As a by-product of the cperatcr signon feature of CICS/VS, a count is maintained of all transactions entered by that operator, together with all operator errors as indicated by abnormal termination of applicaticn prcgrams using the prcgram control ABEND macro instruction. When the operator signs off, using the CSSF transaction code, CICS/VS directs a message containing the operator identification, number of transactions entered, and number cf transaction errors to a transient data destination. This transient data destinaticn may be allocated to a terminal, a disk or tape data set, a line printer, or any cther CICS/VS-supported device. When directed to tape cr disk, these operator statistics may be accumulated for audit, control, or evaluation purposes.

PRIORITY PROCESSING

Each terminal operator is allccated a pricrity code as well as security codes. This operatcr pricrity is used in conjunction with terminal and transaction pricrities to establish the overall task processing priority.

TASK PRICRITY

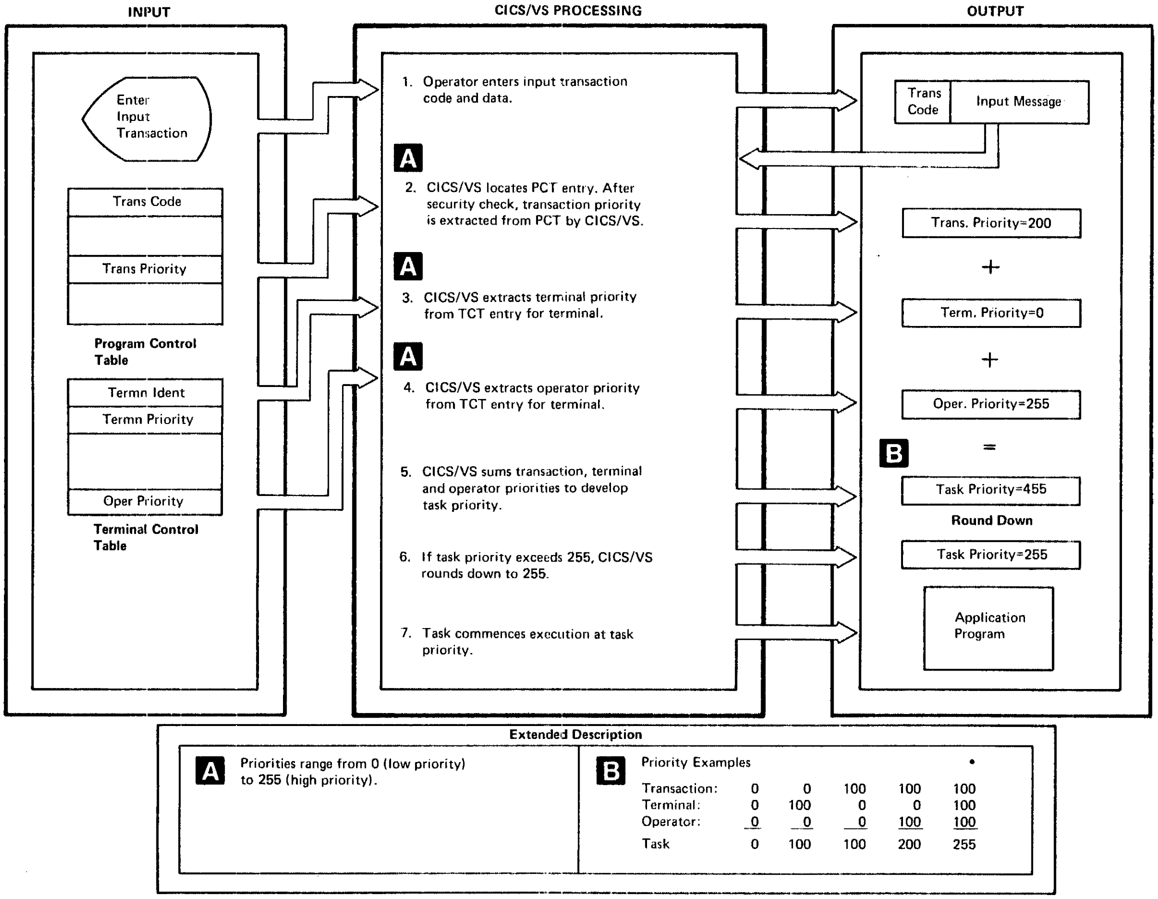
CICS/VS uses priority ccdes ranging from 0 to 255. The 0 represents low priority while 255 represents high priority.

Each operator, terminal, and transaction code can be allocated a priority ccde ranging frcm 0 to 255. The operator priority is contained in the signon table (SNT) and is ccpied across tc the terminal control table (TCT) when the operator signs on. The terminal pricrity is also contained in the TCT, while the transaction priority is contained in the program control table (PCT) entry for that transaction code (see Figure 3-15).

When an operator enters a specific transaction code, his priority and the priority of the terminal he is using are extracted, together with the priority associated with the transaction code entered. These three priorities are added together to produce a total priority. This total priority is used as the task priority, and also ranges from 0 to 255. In the event that the sum of the three priorities exceeds 255, the task priority is rounded down to 255.

This calculation of task priority provides the design team with considerable flexibility to ensure that the best performance and response time are provided in the areas where they are most needed. Thus, operators carrying out higher priority functions than other operators may be given a higher priority code by the user. Similarly, some terminals may be given higher priorities than other terminals. Also, high priority transactions may be given a higher priority value than other transactions.

A very high priority transaction may be given a priority value of 255. In this case, regardless of the operator or terminal priority, that transaction is always given the highest task priority. In the same way, very high priority operators or terminals may be given operator or terminal priorities of 255.



1. Operator enters input transaction code and data.

A

2. CICS/VS locates PCT entry. After security check, transaction priority is extracted from PCT by CICS/VS.

A

3. CICS/VS extracts terminal priority from TCT entry for terminal.

A

4. CICS/VS extracts operator priority from TCT entry for terminal.

5. CICS/VS sums transaction, terminal and operator priorities to develop task priority.

6. If task priority exceeds 255, CICS/VS rounds down to 255.

7. Task commences execution at task priority.

Extended Description

A Priorities range from 0 (low priority) to 255 (high priority).

B Priority Examples

Transaction:	0	0	100	100	100
Terminal:	0	100	0	0	100
Operator:	0	0	0	100	100
Task	0	100	100	200	255

Figure 3-15. Task Priority

The task priority is useful in those cases where, because of the transaction volume, there may be several tasks concurrently executing. In this event, CICS/VS passes control to the highest priority task which is able to continue executing, and that task retains control of the CPU until it requests various CICS/VS services. If the high priority task is not able to continue processing until a particular event (such as an I/O operation) has occurred, CICS/VS passes control to the next highest task which is able to execute. A high priority task is given preference in the use of the CPU and other facilities even if entered later than a lower priority task.

CICS/VS ensures that such high priority tasks are given first preference in processing to enable good performance to be achieved by that task. In the event that two tasks with the same high priority value (for example, 255) are both ready to process, CICS/VS gives control to that task which reached the system first.

CICS/VS is an event-driven system, and as such does not seize control from a currently dispatched (executing) task. Therefore, even a low priority task will continue to execute once it has been dispatched, until it voluntarily relinquishes control by issuing a CICS/VS macro instruction. If no CICS/VS services are required by such a task, it should periodically issue a task control dispatchable WAIT, or a CHAP (change priority), macro instruction. The CHAP need not change the task's priority, but merely relinquish control. (See the next topic, or Chapter 6, for more detail.)

CHANGE PRIORITY

A task may commence execution at one priority, and then may need to change its priority at another phase in its processing. CICS/VS provides this capability through the task control change priority (CHAP) macro instruction (see Chapter 6). A high priority task may be changed through the use of this macro instruction to low priority, or vice versa. In this way, sections of an application program may be given a high priority for processing, while other sections may be given lower priority. This enables a task to dynamically change its priority based on differing requirements determined through execution. Some examples when sections of a program may wish to change the task priority are illustrated in "Priority Change" in Chapter 6.

CHAPTER 4. CICS/VS DATA MANAGEMENT DESIGN

This chapter discusses CICS/VS temporary storage and transient data in a tutorial fashion. Experienced CICS users may wish to omit the section on transient data. However, it is recommended that such users read the section on temporary storage in its entirety. The temporary storage control program (TSP) is changed from that available in previous CICS versions. While still providing compatibility with previous CICS versions, this new TSP provides additional sequential as well as direct accessing capability, and utilizes VSAM.

CICS/VS, together with DL/I, provides extensive data base capability to online applications. In addition to this data base capability (which is discussed in Chapter 5), CICS/VS offers additional facilities for internal data management. This chapter first identifies various application requirements which demand the services offered by CICS/VS temporary storage management and transient data management. It then describes those services which can be used as "design tools" by the system designer to satisfy his own application requirements.

APPLICATION REQUIREMENTS

It is first necessary to define the various data management functions (as distinct from data base capability) which online applications require of a DB/DC system. These functions are briefly described below.

WORK FILE CAPABILITY

Most online applications require the ability to store information for later retrieval and use. This function is sometimes referred to as a "scratchpad" or work file capability, and is analogous to a person using sheets of paper to jot down the intermediate results of calculations for later use in processing.

The following are two main work file requirements used for most online applications:

- Scratchpad capability
- Queuing capability

Scratchpad Capability

This capability refers to the temporary storage of information for later retrieval. In a batch environment, this capability is often provided through the use of work data sets. In CICS/VS, this capability is provided by the CICS/VS temporary storage control program. The application program identifies data which is to be temporarily stored by name, and subsequently retrieved by name without any consideration of its physical location. Online application uses of temporary storage include the following.

Intermediate Results: The storage of intermediate results developed during the processing of a transaction, for use later in the processing of that transaction.

Error Correction: The storage of input transactions which were found to be in error, for subsequent use when the corrected error fields are received from the terminal.

Data Transfer: A temporary storage of data so it can be used to transfer data between programs. This data transfer may occur immediately or at some future time.

Terminal Paging: An application program may develop several pages of information to be displayed at a terminal. This information should be temporarily stored until the terminal operator requests that it be displayed for his attention.

Queuing Capability

In addition to the temporary storage of data, online applications generally require a facility which will enable data to be queued for subsequent processing. The difference between this and temporary storage is that temporary storage stores and retrieves individual sections of data, while a queuing capability enables several different sections of the same type of data to be queued, and then all sections retrieved together, sequentially, in the order that they were queued. In CICS/VS, this queuing capability is provided by the CICS/VS transient data control program. Examples in which online applications may utilize a queuing capability follow:

Batch Transaction Processing: Transactions of a particular type may be received from many terminals. If the application requires that all of these transactions be processed together, they may be stored in a unique queue for that transaction type, in the order that they reach the CPU. This queue of transactions may then be processed in the CICS/VS partition as a small sequential group of transactions.

Batched Message Transmission: The online application may require that messages be batched and transmitted to specific terminals.

Batch Partition Data Transfer: The online application may require that information be transferred to batch partitions for further processing, and the results of that processing be provided to the online application for input at a later time.

CICS/VS TEMPORARY STORAGE MANAGEMENT

The temporary storage management facility of CICS/VS provides a scratchpad capability for online application programs. It enables data to be stored either in dynamic storage or on auxiliary storage. Data to be stored can be identified symbolically, and retrieved symbolically, without application programs being concerned with the actual physical location of that data. Data can be retrieved on request by an application program in either a sequential or a direct access manner. Temporary storage allows records to be up to 32,000 bytes in length, but supports variable-length records only.

TEMPORARY STORAGE USAGE

The previous discussion of application requirements identified the general use of a scratchpad facility by application programs. CICS/VS temporary storage management is used to meet these application requirements as follows.

Data Transfer Facility

The ability to temporarily save data for later use, and retrieve it symbolically by name at a future time, enables easier implementation of complex processing. This complex processing may be broken into several logical steps, each step carried out by a separate module. Information may be passed between these modules using temporary storage.

Depending upon the amount of information to be passed, and the time period before that information will be used, this data may be stored either in dynamic storage or, alternatively, in auxiliary storage.

Scratchpad Facility

Temporary storage may be used to save information for later use. An example would be the saving of error transactions for later combination with corrected fields received from a terminal, as described in "Error Correction". Using this capability, correct information in the original error transaction does not have to be reentered by the terminal operator. Consequently, error correction is easier, and the potential for further operator errors is reduced.

Terminal Paging

Terminal paging in CICS/VS is also supported through the use of temporary storage. Pages of information developed by application programs are presented by them to CICS/VS. These pages are stored in temporary storage for transmission to the terminal operator on request. Refer to "Terminal Paging" in Chapter 3 for more detail.

Message Routing

The ability to transmit messages from one terminal to another terminal, using the CICS/VS message switching transaction CMSG, or the BMS ROUTE macro instruction, is supported through the use of temporary storage. These messages are automatically transmitted to the relevant terminal when that terminal is able to receive them, or the specified operator has signed on to CICS/VS. Refer to "Message Routing" in Chapter 3 for more detail.

Interval Control

The CICS/VS interval control program uses temporary storage to pass data from one task to another task which is to be initiated at a future time. An application program may indicate the task to be initiated at a specified time (based on elapsed time or, alternatively, time of day) and may transfer data to that future task. The interval control PUT macro instruction results in the data to be transferred being written to temporary storage on disk for subsequent retrieval by the interval control GET macro instruction. Refer to "Interval Control" in Chapter 6 for more detail.

DATA IDENTIFICATION

Each record may be presented to temporary storage with a unique eight-character data identification. Alternatively, several records may be presented with the same data identification. A queue of records associated with a particular logical function (as indicated by the data identification) can be developed, and subsequently retrieved in the same sequence.

The data identification is used by CICS/VS to develop a data element which contains that identification, the sequence or entry number of the record in a queue of records with the same data identification, and the location of the record either in dynamic storage or on disk. These data elements are maintained in CICS/VS dynamic storage. As records are written to temporary storage, data elements are dynamically built by CICS/VS and saved in dynamic storage. The number of temporary storage records which may be retained is limited only by the availability of dynamic storage and/or the amount of disk space allocated to the temporary storage data set.

Because many tasks may concurrently use the same program, the use of a constant in the program for identification of individual records is not advisable. The data identification may be dynamically generated by the program based upon information such as:

- A combination of transaction identification (four characters) and operator identification (three characters) will enable that operator to store one record at a time for each transaction identification.

- A combination of operator identification and time of day, or transaction identification and time of day, will enable the record to be uniquely identified. However, it requires the application program to determine the time of day and then respond to the terminal operator with the allocated data identification. He may then use it to uniquely identify the record in a later transaction.

- Each task initiated by CICS/VS is given a unique task sequence number. This task number may be used as the data identification; it may be returned to the terminal operator for subsequent reentry by him when the relevant record is to be retrieved.

The techniques for unique data identification described above assume an application environment where information is to be stored by the user's program, and directly retrieved at some future time under control of the terminal operator.

If temporary storage is used to pass data from one application program to another, the allocated data identification may be passed to a subsequent application program (executed under control of the same task) through the transaction work area (TWA) appended to the task control area (TCA) for that task. The program (executing under the same TCA) which is to retrieve the data from temporary storage can obtain the allocated data identification from the TWA. This data identification is then used to identify the record to be retrieved.

If a record within a temporary storage queue is to be directly retrieved, it must be uniquely referenced by the data identification (ID) and its relevant entry (or sequence) number. When a record is written to a temporary storage queue (data identification is nonunique), it is placed at the end of the queue of records with that same data identification. Temporary storage management will allocate the next sequential entry number and return this entry number to the program. The record is now uniquely identified by the data ID and the entry number.

This data ID and entry number may be transmitted to a terminal operator for subsequent reentry, if the retrieval is to be initiated by the terminal operator. If the retrieval is to be initiated automatically by subsequent application programs executed by the same task, the data identification and entry number should be saved in the TWA. The program which is to retrieve that unique record may then extract this information from the TWA for use.

USE OF DYNAMIC STORAGE BY TEMPORARY STORAGE

Dynamic storage is a valuable resource, and the overall performance of the online system is directly related to the amount of available dynamic storage and its relationship to real storage available for use as a virtual storage page pool (see "CICS/VS Working Set" in Chapter 7).

Generally, dynamic storage residence of records should be used only when the life of those records is to be of very short duration. Its main purpose is in passing data between program modules which are executed under control of the same task. Once the data has been passed between modules through dynamic storage, that data should be deleted and the storage occupied by it freed. Dynamic storage may be used for record queues as well as unique entries; however, write requests to dynamic and auxiliary storage with the same data identification cannot be used. CICS/VS will force all subsequent write requests with the same data identification to use the same storage facility specified by the first request.

The length of records to be stored in dynamic storage may be up to the VSAM control interval size specified during CICS/VS system initialization, less 84 bytes for CICS/VS control information.

CICS/VS permits temporary storage records to reside in dynamic storage only if the CICS/VS system is generated indicating no auxiliary storage residence support is required. The specification of no auxiliary storage support removes the requirement for VSAM by temporary storage. Instead, virtual storage is utilized; temporary storage information is only paged into real storage when referenced.

Any temporary storage information residing in dynamic storage is lost if a controlled or uncontrolled shutdown occurs. See the CICS/VS System Programmer's Reference Manual, SH20-9004, and "Temporary Storage Recovery" in Chapter 8 for additional information.

As a general rule, if a record must be stored for more than one second, it should be directed to auxiliary or secondary storage rather than to dynamic or main storage. Dynamic storage is then available as much as possible for use in initiating concurrently executed tasks. Certainly, the writing of records to disk, and the subsequent retrieval from disk, will involve file accesses and so increase the processing time of those particular tasks. However, the overall effect on the entire online system is one of potentially better performance than would result if considerable dynamic storage were utilized for temporary storage residence.

ACCESSING RECORDS IN TEMPORARY STORAGE

Temporary storage supports variable-length records only. A queue or message set of records may be developed by issuing a temporary storage PUTQ macro instruction for each record, using the same data identification. As each record is written, temporary storage allocates the next sequential entry number and returns it to the application program.

Using the data identification and the entry number, the records in the queue can be retrieved by application programs either sequentially, in the chronological order in which they were written, or directly accessed by referencing a specific entry number.

A queue of records can be retrieved sequentially by specifying the data identification allocated for that queue and issuing a temporary storage GETQ macro instruction. Temporary storage management retrieves the first record in the queue for that data identification and presents it to the application program. Each subsequent GETQ macro instruction retrieves the next record in sequence until the last record has been retrieved, when an end-of-queue indication will be returned to the program.

Alternatively, if it is required to commence sequential retrieval, not from the beginning of the queue but from a logical point within the queue, both the data identification and the entry number are specified by the program. GETQ macro instructions are then issued to retrieve each record sequentially from the logical starting point in the queue.

The program may directly retrieve records by issuing a temporary storage GETQ macro instruction with the specific entry number of the record in a queue to be directly retrieved.

A record can subsequently be updated by issuing a temporary storage PUTQ macro instruction specifying the relevant entry number.

The facilities offered by temporary storage for direct and sequential retrieval of information make it a powerful work file capability for online applications. Information may be retrieved as often as required until it is no longer needed. At that time, the records may be deleted.

Queues of records based upon a specific data identification may be purged by an application program PURGE macro instruction. The deletion or purging of these records results in the logical deletion of those records in the temporary storage data set, with the disk space occupied by those records being reclaimed when the space is subsequently used for another record. The data elements describing the deleted or purged records are freed, and the dynamic storage occupied by these records is reclaimed for other uses.

The CICS/VS temporary storage control program supports requests for specific records using the PUT, GET, and RELEASE macro instructions provided in previous versions of CICS. However, PUT, GET, and RELEASE are mutually exclusive with PUTQ, GETQ, and PURGE on a data identification basis. That is, a record written by a PUT macro instruction cannot be retrieved by a GETQ, or deleted by a PURGE, for example.

TEMPORARY STORAGE RECOVERY

After a controlled or uncontrolled termination of CICS/VS, temporary storage records on disk may remain available for use, if desired. Temporary storage in dynamic storage is lost.

On restart of CICS/VS, either a "cold start," "warm start," or emergency restart may be specified. If a cold start of temporary storage is specified, any information recorded on disk is lost.

If a warm start is specified on system restart, the information in the temporary storage data set is retained. The temporary storage keypoint recorded at system termination (see "Termination Keypoints" in Chapter 8) is used to reconstruct the data elements in CICS/VS dynamic storage, to enable subsequent retrieval of information by application programs once the system has been restarted.

If an emergency restart is specified, the information in the temporary storage data set is retained. The contents of that data set and any temporary storage update activity automatically logged to the CICS/VS system log prior to uncontrolled shutdown are used to reconstruct temporary storage tables in dynamic storage. These tables identify the status of temporary storage at uncontrolled shutdown. The data identification of temporary storage records and queues, the number of entries in queues, the location of each entry in auxiliary storage, and the status of available space in the temporary storage data set are reconstructed during emergency restart.

The processing of in-flight tasks is also backed out during emergency restart. A task is considered in-flight if it did not pass through a user synchronization point (with no subsequent logging activity) or terminate before uncontrolled shutdown.

Thus, a consideration in the use of dynamic storage or auxiliary storage as a temporary storage medium is the requirement for recoverability. Information stored in main storage will be lost; information stored in auxiliary storage may be recovered, if a warm start is specified on restart.

SELECTION OF TEMPORARY STORAGE OR TRANSACTION WORK AREA DATA TRANSFER

The design team must indicate whether information to be passed from one module to another may be transferred using the transaction work area (TWA) appended to the task control area (TCA), or that temporary storage must be used.

TWA for Data Transfer

The TWA can be used only if the information will be subsequently used by the same application program, or by another application program which executes under control of the same TCA. That is, control must be passed to the subsequent program either by program control XCTL or by LINK macro instructions. If the information is to be passed to some future task initiated by time, or by a subsequent transaction entered by a terminal operator, then the TWA cannot be used. This is because the TCA and associated TWA are destroyed when the task which generated the information terminates execution. Consequently, the TWA may be used for data transfer of a short-term nature, while temporary storage is generally used for data transfer of long-term nature.

TWA Size

A consideration in the use of a TWA or temporary storage is the amount of data to be stored. The size of the TWA associated with a transaction code is stored in the program control table (PCT). This TWA size is used to allocate a TWA appended to the TCA. Thus, if a TWA of 200 bytes is indicated in the PCT, the TCA is allocated 200 bytes more than if no TWA size is specified.

TWA for Short-Term Data Transfer

A further factor is the duration of execution of the task, and the amount of time between when data may be stored in the TWA and when it will be subsequently retrieved from the TWA. As a general rule, if data may remain in the TWA for longer than one second it should be stored in temporary storage. This would be particularly advisable if a TWA much larger than 200 to 300 bytes was to be used. Furthermore, because of the relatively low activity of use of this data (because of the long execution time), it should be stored on disk rather than in dynamic storage address space.

Variable TWA Size Requirements

Another factor is the possible requirement of the program for different size TWAs based upon the processing required. For example, 90% of transactions which use the same transaction code and application program may require a TWA of 50 bytes. However, the remaining 10% of these transactions may require a TWA of 500 bytes, say. If a TWA was used for all transactions by this program, a 500-byte TWA would have to be specified in the relevant PCT entry. This would mean that for 90% of transactions using that program, 450 bytes of storage would be wasted.

A more efficient solution in this case would be to allocate a 50-byte TWA, and utilize this TWA for the 90% of transactions which need 50 bytes. In the case of the remaining 10% of transactions, temporary storage on disk should be utilized. Thus, storage is used most efficiently, with the additional time to store information on disk and retrieve it from disk only affecting 10% of the transactions in this example.

CICS/VS TRANSIENT DATA

The queuing facility provided by CICS/VS for online applications is supported by the transient data management routine of CICS/VS. There are two types of transient data queues. These are:

Extrapartition: Extrapartition queues are sequential data sets used for transfer of information between CICS/VS and batch partitions.

Intrapartition: The intrapartition data set supports queues used within the CICS/VS partition itself, to transfer information between CICS/VS tasks.

TRANSIENT DATA USAGE

The application uses of extrapartition and intrapartition data sets will now be discussed.

Extrapartition Data Sets

Extrapartition data sets in CICS/VS are used for the following main purposes:

Batch Data Transfer: Information which is to be passed from CICS/VS to batch partitions is directed to extrapartition data sets or queues. These data sets are normal sequential data sets using QSAM for OS/VS or SAM for DOS/VS.

Similarly, information to be passed from a batch partition to CICS/VS is read by the relevant CICS/VS task from an extrapartition input data set.

Sequential Devices: Extrapartition data sets may be used by CICS/VS to communicate with various sequential devices, such as line printers. Because the standard sequential access method under OS/VS or DOS/VS is used to support extrapartition data sets, those devices supported by the standard sequential access method can be utilized by CICS/VS. These include card reader, line printer, disk, and tape. Particularly because of OS/VS device independence, most sequential devices which are supported by QSAM may be utilized as either input or output data sets by CICS/VS, when the user specifies them as extrapartition data sets.

Intrapartition Data Set

Intrapartition queues are used to pass direct access organized data (chained sequentially) between CICS/VS tasks. A number of applicaticn-oriented uses for intrapartition files are detailed below.

Batch Queues: Data received from many terminals for the same application may be consolidated in one queue for processing as a batch. Each concurrently executing task may direct the data to the relevant batch queue, where it is chained sequentially. Subsequently, this batch or queue of data may be prccessed as an input file of information by a CICS/VS task.

Automatic Tasks: Data stored as a queue as described above may be automatically prccessed by a CICS/VS task when a specified amount of information has been queued. Based upon a trigger level (or count) for that queue, a specified task may be automatically initiated to process that quantity of data. The trigger level may vary from 0 (which implies nc automatic task initiation) through 1 (which initiates a task each time information is written tc the queue) tc a trigger level of greater than 1.

Terminal Output: Output may be autcmatically directed to a terminal from several tasks. This automatic cutput may nct be able to be sent to the terminal for some time, because it is engaged in other activity such as entering an input transaction or receiving output from previous transactions.

In addition, the terminal may be one to which output is only sent when requested. An example cf such a terminal wculd be a video terminal. Automatic output directed to a video terminal may not be displayed at a convenient time, cr may not allow sufficient time for assimilaticn of the information displayed. Hard-copy terminals, however, may be able to receive automatic output at any time they are not active, unless they are used with preprinted stationery. In this case, automatic output for a terminal must be queued on disk until the terminal is able to receive it, cr until the terminal operator has explicitly requested it.

Output to be directed autcmatically to a terminal is queued on an intrapartition queue. A trigger level may be associated with this queue such that when a specified number of output messages have been queued a task is autcmatically initiated to transmit those messages to the terminal, if the terminal is able to receive those messages at that time.

Audit: Intrapartition (cr extrapartition) queues may be used to accumulate informaticn for audit purposes. Intrapartition queues may be specified as being ncnreusable. Data written to these queues is accumulated throughout the operaticnal period of CICS/VS, and will only be deleted (and the disk space used will cnly be freed) by an explicit transient data PURGE macro instructicn issued by an application program.

Alternatively, queues may be specified as reusable, in which case information on these queues is purged automatically by CICS/VS when the data has been read by application programs. The subset option of CICS/DOS/VS supports extrapartition data sets but not intrapartition data sets. (See "CICS/DCS/VS Subset Option" in Chapter 7.)

EXTRAPARTITION TRANSIENT DATA

As discussed above, extrapartition data sets provide a sequential data set capability tc CICS/VS. Standard access methods such as QSAM for OS/VS or SAM for DOS/VS are utilized. The specification of the particular sequential data set is made at system generation time. Further information describing that data set may be provided at CICS/VS system initiation time from OS/VS DD, or DOS/VS DLBL and EXTENT, job control statements. Extrapartition data sets can be either fixed-length or variable-length, blocked or unblocked data sets.

Record Accessing

Each extrapartition data set is identified by a four-character destination identification. This destination identification is specified by a CICS/VS task when it requests input (GET) or output (PUT) on a particular data set (see Figure 4-1).

This destination identification is used to locate the relevant entry in a destination control table (DCT) describing that particular extrapartition data set. CICS/VS transient data management then issues the appropriate DOS/VS or OS/VS GET or PUT macro instructions for the particular sequential access method. (See the CICS/VS Application Programmer's Reference Manual, SH20-9003.)

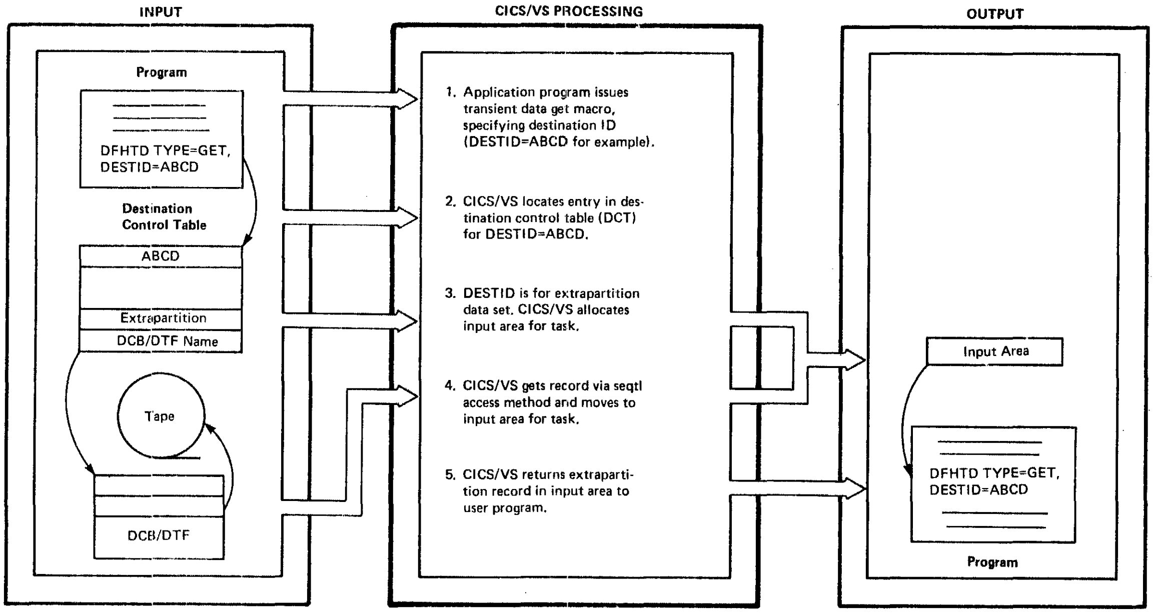

INPUT
Program
DFHTD TYPE=GET,
DESTID=ABCD
Destination
Control Table
ABCD
Extrapartition
DCB/DTF Name
Tape
DCB/DTF
CICS/VS PROCESSING
1. Application program issues transient data get macro, specifying destination ID (DESTID=ABCD for example).
2. CICS/VS locates entry in destination control table (DCT) for DESTID=ABCD.
3. DESTID is for extrapartition data set. CICS/VS allocates input area for task.
4. CICS/VS gets record via seqtl access method and moves to input area for task.
5. CICS/VS returns extrapartition record in input area to user program.
OUTPUT
Input Area
DFHTD TYPE=GET,
DESTID=ABCD
Program

Figure 4-1. Extrapartition Data Set Accessing

For output to a sequential data set, an application program first requests the CICS/VS storage control program to allocate storage to be used as an output area. The output record is then constructed by the application program, after which the program issues a transient data PUT macro instruction indicating the relevant destination identification of the output data set. The output record is then written by transient data to the specified sequential data set. On successful completion of output, without error, the allocated output area is automatically freed by CICS/VS and returned to dynamic storage for use by other tasks.

When an application program has to initiate input from a sequential data set, it issues a transient data GET macro instruction specifying the relevant destination identification. Transient data determines the data set involved, automatically requests that an input area large enough to contain the next record be allocated for the particular task, and moves the next sequential input record into that area. The address of that input area is returned to the requesting task after successful completion without error. The accessing of extrapartition data sets is illustrated in Figure 4-1.

Extrapartition data sets may be either fixed-length or variable-length, blocked or unblocked.

Recovery of Extrapartition Data Sets

CICS/VS does not attempt to recover extrapartition data sets after a controlled shutdown or in the event of abnormal termination or system failure. (This subject is discussed in more detail in Chapter 8.)

INTRAPARTITION TRANSIENT DATA

As discussed previously, the intrapartition data set provides a useful queuing facility for passing information between CICS/VS tasks. Its main use is to provide support for accumulation of data to be either processed as a batch or automatically transmitted to a terminal, for example.

Record Accessing

Data is written to or read from intrapartition queues by CICS/VS application programs in exactly the same way as for extrapartition data sets. However, only variable-length records are supported. An application program requests an output area to be allocated to it by CICS/VS storage control, sets up the output record, and issues a transient data PUT macro instruction specifying the relevant four-character intrapartition destination identification.

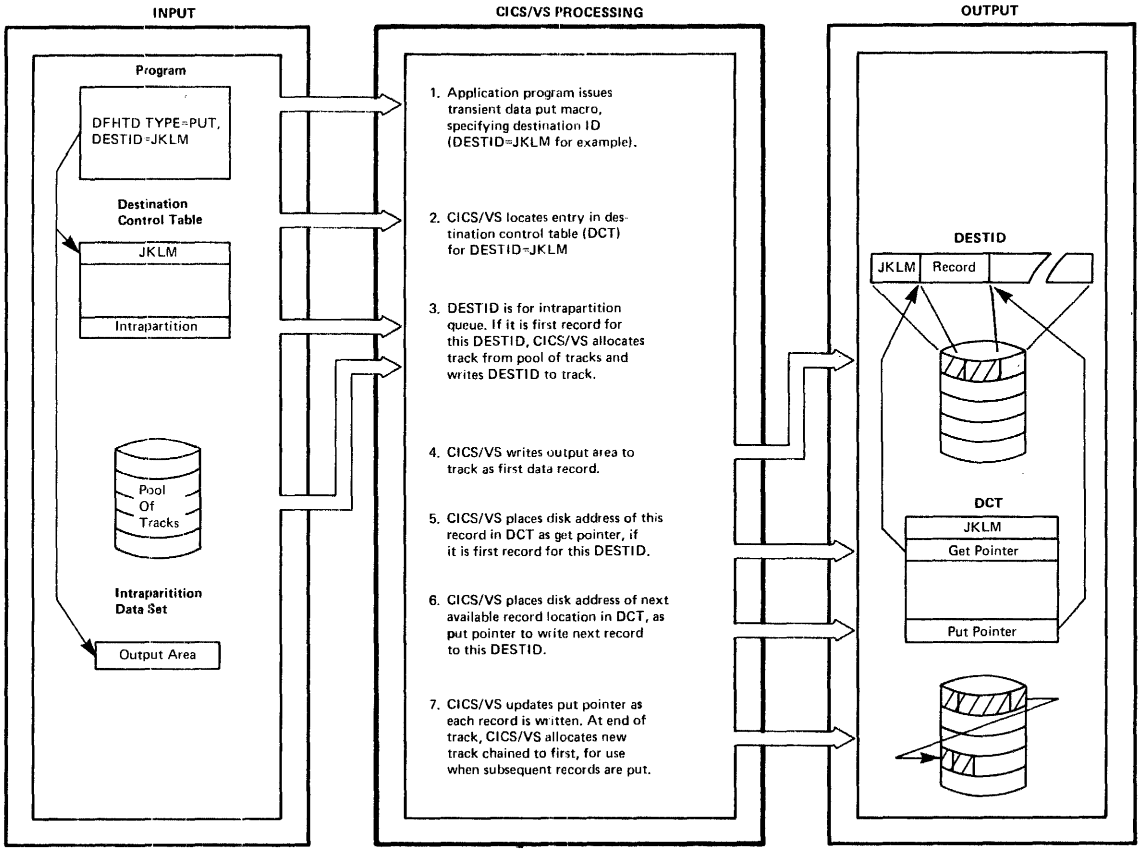
Similarly, for input, when a transient data GET macro instruction is issued by an application program, transient data requests that an input area be allocated. The record is then read and passed to the requesting task.

From a general programming point of view, there is no effective difference between reading and writing extrapartition data sets or intrapartition queues. The indication by the program as to whether an extrapartition data set or an intrapartition queue is to be used is the specification of the relevant destination identification.

One main difference between extrapartition and intrapartition queues, however, is that intrapartition queues may be specified as being reusable, if required. Thus they can be used as work files if needed, queuing data to be processed, and then, after processing that data, deleting it so that the disk space it occupied can be utilized for other purposes. This is discussed in more detail in "Reusable Intrapartition Queues" later in this chapter.

Intrapartition Disk Organization

CICS/VS uses a direct access data set to support intrapartition queues. The disk space allocated for the intrapartition data set is regarded as a pool of tracks which may be allocated to intrapartition queues (destinations) as required (see Figure 4-2).



1. Application program issues transient data put macro, specifying destination ID (DESTID=JKLM for example).
2. CICS/VS locates entry in destination control table (DCT) for DESTID=JKLM
3. DESTID is for intrapartition queue. If it is first record for this DESTID, CICS/VS allocates track from pool of tracks and writes DESTID to track.
4. CICS/VS writes output area to track as first data record.
5. CICS/VS places disk address of this record in DCT as get pointer, if it is first record for this DESTID.
6. CICS/VS places disk address of next available record location in DCT, as put pointer to write next record to this DESTID.
7. CICS/VS updates put pointer as each record is written. At end of track, CICS/VS allocates new track chained to first, for use when subsequent records are put.

Figure 4-2. Intrapartiticn Disk Crganization

Transient data maintains a series of tracks allocated to each active destination, based on the dyramic requirements fcr intrapartition disk space. However, reccrds are logically read frcm a destination in the sequence in which they were written; it thus appears to the CICS/VS task as if it were operatirg on a normal sequential data set. The data set is actually a direct access file.

INTRAPARTITION QUEUE USAGE

As discussed above, intrapartition queues are generally used to accumulate and process data as a batch of records. The various application uses to which intrapartition queues can be applied are ncw discussed in more detail.

Batch Retrieval

Records may be accumulated as a batch on an intrapartition queue or destination. Retrieval and processing of these records are achieved by a task issuing transient data GET macro instructions specifying that intrapartition destination. The initiation of these tasks may be achieved in one of three ways:

Transaction Initiation: A transaction entered by a terminal can initiate a task which issues transient data GET macro instructions for a particular intrapartition destination. Data retrieved in this way can then be processed as required.

Interval Control Initiation: A task can be initiated at a future time based upon elapsed time or time of day. This task can issue transient data or interval control GET macro instructions to read records queued on an intrapartition destination and process them. (See "Interval Control.")

Automatic Task Initiation: A task may be automatically initiated by transient data when a specified number of records have been written to an intrapartition destination. The trigger level specified in the DCT entry for that destination is compared with the count of records which still remain to be read. When the queue count equals the trigger level, a specified task (as identified by a transaction code in the DCT entry) is initiated. This task may issue transient data GET macro instructions to read and process the data on that queue. This is discussed further in the section, "Terminal Output."

Intrapartition Recovery:

CICS/VS supports the recovery of intrapartition transient data queues on a warm start following a controlled shutdown, and on an emergency restart following an uncontrolled shutdown. The DCT status of each intrapartition destination is reestablished to reflect the GET pointer, PUT pointer, queue count and trigger level status as it was prior to the shutdown. Intrapartition queues are recovered, and automatic task initiation can then proceed after CICS/VS restart as if shutdown had not occurred.

On emergency restart following an uncontrolled shutdown, any intrapartition destinations can be recovered to reflect all activity against those destinations up to the point of uncontrolled shutdown. This is called "physical recovery." Alternatively, any intrapartition destinations can be recovered to reflect the activity of completed tasks prior to uncontrolled shutdown; all in-flight task activity at uncontrolled shutdown is backed out during an emergency restart. This is called "logical recovery." The user specifies in the DCT, at system generation time, whether an intrapartition destination requires physical recovery or logical recovery. Refer to "Transient Data Recovery" in Chapter 8 for additional information.

Terminal Output

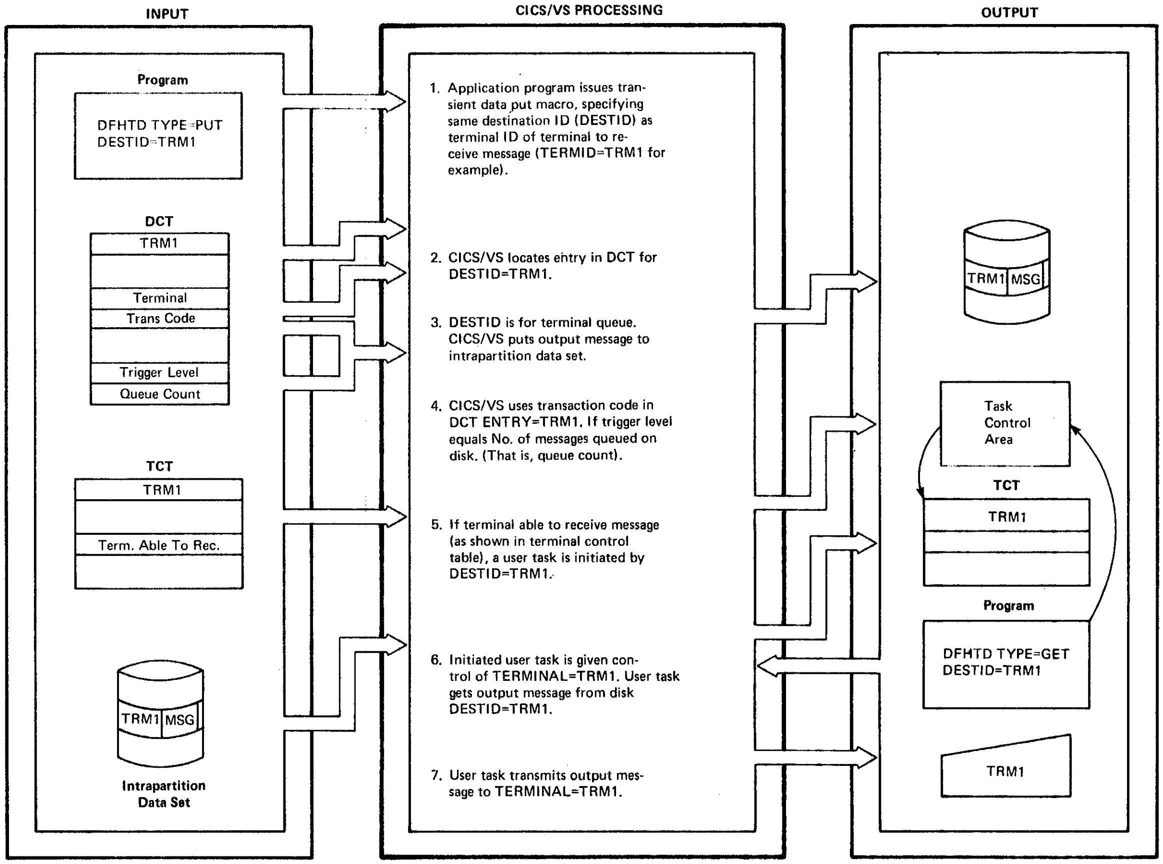
Data may be directed to a terminal from many tasks. That terminal may presently be active either entering input or receiving output from one task. When other tasks wish to transmit output messages to that same terminal, it is necessary for these messages to be queued on disk until the terminal is ready to receive them.

This message queuing is achieved by requiring the other tasks to write the terminal output messages to a transient data destination. This destination is intrapartition, and furthermore is identified as being associated with a terminal. The destination identification used must be identical to the terminal identification in the terminal control table (TCT) entry for the associated terminal. Tasks may issue transient data PUT macro instructions specifying as a destination identification the terminal identification. These output messages will then be queued in the intrapartition data set (see Figure 4-3).

To initiate transmissicn of these output messages to the relevant terminal, a trigger level cf 1 is generally specified for that terminal destination. As socn as cne cutput message has been written to that terminal intrapartition destination, a task (identified by a transaction code in that DCT entry fcr the relevant destination) is eligible to be automatically initiated. The prcgram used by that transaction code will conventionally be a ccmmcn prcgram, developed by the installation, to transmit data from various destinations to their relevant terminals.

However, to be able to transmit messages from the intrapartition destination to the terminal, that terminal must be idle and able to receive automatic output--that is, the output sent to the terminal is not in response to a transaction entered earlier by the terminal.

Accordingly, unless the associated terminal is idle and able to receive output, a task is not automatically initiated based upon the trigger level of a terminal destination. If these conditions exist, the task is initiated. The terminal is allocated to that task as if the terminal itself had entered the transaction code which initiated the automatic task. The automatic task may now issue transient data GET macrc instructions tc retrieve output messages from the particular terminal intrapartition destination. These messages may be transmitted directly tc the terminal using CICS/VS terminal ccntrol or basic mapping support macro instructions.



1. Application program issues transient data put macro, specifying same destination ID (DESTID) as terminal ID of terminal to receive message (TERMID=TRM1 for example).

2. CICS/VS locates entry in DCT for DESTID=TRM1.

3. DESTID is for terminal queue. CICS/VS puts output message to intrapartition data set.

4. CICS/VS uses transaction code in DCT ENTRY=TRM1. If trigger level equals No. of messages queued on disk. (That is, queue count).

5. If terminal able to receive message (as shown in terminal control table), a user task is initiated by DESTID=TRM1.

6. Initiated user task is given control of TERMINAL=TRM1. User task gets output message from disk DESTID=TRM1.

7. User task transmits output message to TERMINAL=TRM1.

Figure 4-3. Terminal Output Via Intrapartition Data Set

Terminal Status

To indicate whether a terminal may receive autcmatic output or not, a processing status is defined fcr each CICS/VS terminal. The processing status codes are:

- TRANSACTICN status
- TRANSCEIVE status
- RECEIVE status
- INPUT status
- PAGE status
- AUTOPAGE status

TRANSACTION processing status indicates that a terminal is unable to receive automatic output. It can receive output only as a result of an input transaction entered from that same terminal. Output queued from other tasks for a TRANSACTION status terminal can be transmitted to it only when the terminal operator enters a transaction code which will read the data from the relevant intrapartition queue, and send it to that terminal. The terminal operator has control over when he will receive the queued output. Generally, and particularly for video terminals, one intrapartition message would be transmitted each time the relevant transaction code is entered. The terminal operator can then assimilate the information presented to him before the next output message is requested.

TRANSCEIVE status indicates that a terminal may enter input transactions, but can also receive automatic output from other tasks. This is generally used for hard-copy terminals, where several lines of output may be automatically transmitted when the terminal is idle.

RECEIVE status indicates that a terminal is unable to enter any input data, but is only able to receive automatic output from other tasks. This is generally used for printers.

INPUT status indicates that a terminal can enter data but cannot receive data.

PAGE status indicates that a terminal can only retrieve pages on request, one at a time.

AUTOPAGE status indicates that a terminal will receive all pages queued for it.

Two additional terminal status codes are used to indicate the activity status of each CICS/VS terminal. These are:

- IN-SERVICE status
- OUT-OF-SERVICE status

IN-SERVICE status indicates that the terminal is presently active and able to process as defined above.

OUT-OF-SERVICE status indicates that the terminal is presently inactive, either because it has been marked out-of-service by the master terminal operator for example, or because of an unrecoverable I/O error which occurred on that terminal. In this case, it is unable to enter any messages or receive any output, automatic or otherwise.

Thus, for a task to be automatically initiated based upon a terminal intrapartition destination trigger level, the relevant terminal must have the following status:

- IN-SERVICE status
- TRANSCEIVE status, or RECEIVE status

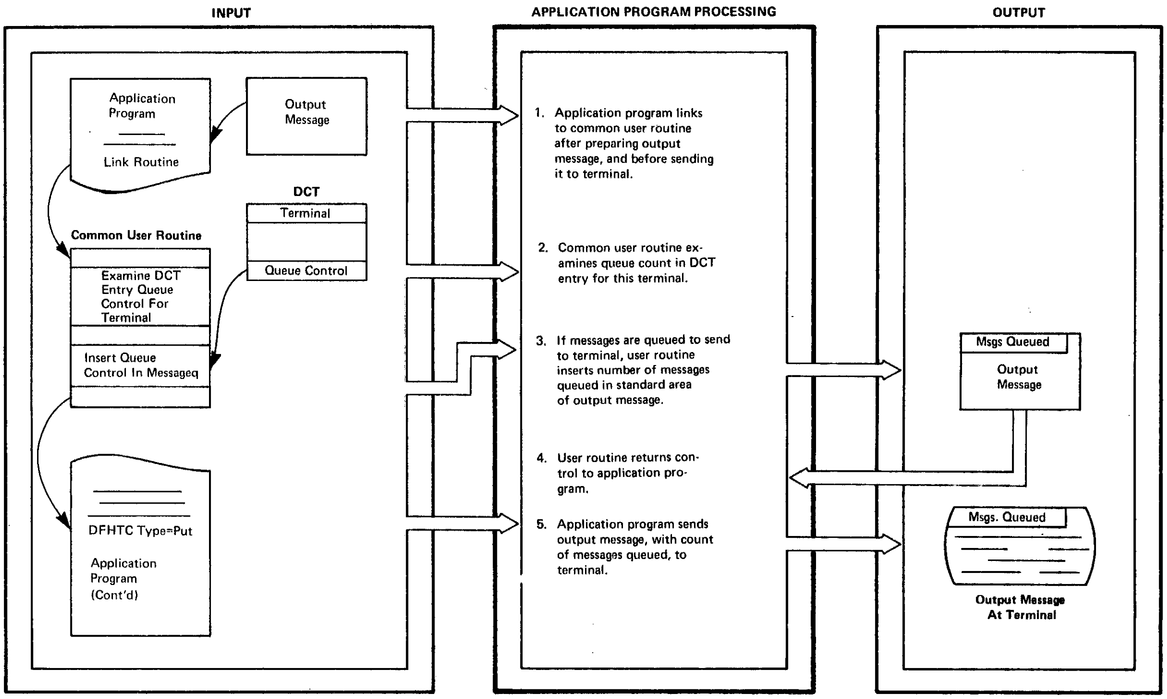
If the terminal is OUT-OF-SERVICE, messages are accumulated on the intrapartition destination queue until the terminal is placed IN-SERVICE. If the status is TRANSACTION, messages are also accumulated on the intrapartition queue until either the status is changed to RECEIVE or TRANSCEIVE, or the terminal operator enters the transaction code to initiate a task which will read the messages from transient data and send them to the terminal.

A VTAM-supported terminal (such as the 3600) which supports automatic task initiation, may be IN-SERVICE and in TRANSCEIVE or RECEIVE status as indicated in its relevant TCT entry but may not currently be connected to CICS/VS. It may be operating offline or be communicating with other VTAM application programs. If a task is to be automatically initiated for that terminal, CICS/VS will request VTAM to establish connection with the relevant logical unit. This may require VTAM to request that another VTAM application program communicating with the logical unit release it for connection to CICS/VS, or may require VTAM to establish a new logical connection (session) to the logical unit currently in offline mode.

Notification of Queued Output Messages

In the case of a TRANSACTION status terminal, some indication should be given to the terminal operator that messages are queued. This can be done either by the terminal operator periodically requesting that any messages queued be sent to him, or through the techniques shown in Figures 4-4 and 4-5.

Figure 4-4 shows one terminal operator notification technique. The application program that retrieves the data from Transient Data may indicate in a standard area of a display screen the number of messages to be sent. This is then presented to the program for incorporation into the output message that is sent to the terminal. Part of the response sent back to that terminal then indicates the number of messages presently queued to be transmitted to the operator upon his request.


INPUT
APPLICATION PROGRAM PROCESSING
OUTPUT
Application Program
Link Routine
Output Message
DCT
Terminal
Queue Control
Common User Routine
Examine DCT Entry Queue Control For Terminal
Insert Queue Control In Messageq
DFHTC Type=Put
Application Program (Cont'd)
Msgs Queued
Output Message
Msgs. Queued
Output Message At Terminal

1. Application program links to common user routine after preparing output message, and before sending it to terminal.
2. Common user routine examines queue count in DCT entry for this terminal.
3. If messages are queued to send to terminal, user routine inserts number of messages queued in standard area of output message.
4. User routine returns control to application program.
5. Application program sends output message, with count of messages queued, to terminal.

Figure 4-4. Notification to Terminal Operator of Automatic Output

A second technique is shown in Figure 4-5, and utilizes the terminal paging facility of CICS/VS to control automatic output to the terminal.

The terminals must be specified as TRANSCEIVE or RECEIVE status, such that automatic output may be sent to them. Tasks preparing output to be transmitted to a specific terminal prepare that output as a series of pages to be displayed to the terminal. These pages, however, are directed to temporary storage through the use of the BMS terminal paging macro instructions, instead of to the intrapartition destination for that terminal.

The terminal operator may then request at his convenience pages of information to be displayed in whichever sequence he requires.

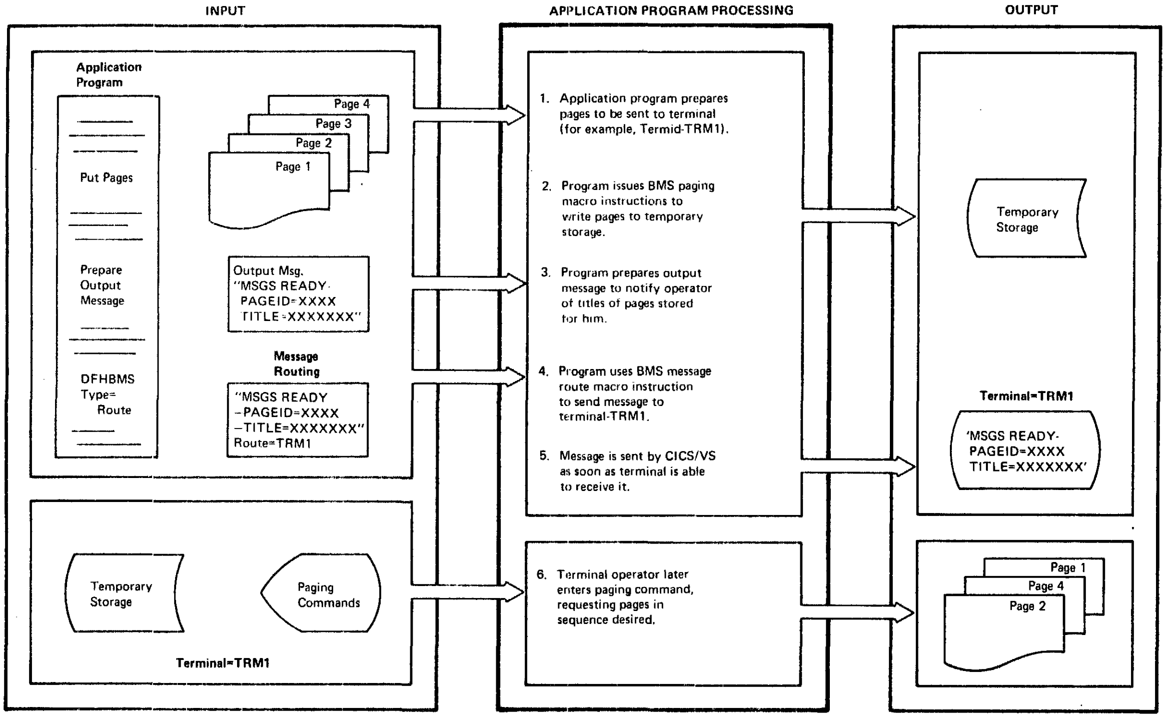

INPUT
APPLICATION PROGRAM PROCESSING
OUTPUT
Application Program
Put Pages
Prepare Output Message
DFHBMS Type= Route
Page 4
Page 3
Page 2
Page 1
Output Msg. "MSGS READY PAGEID=XXXX TITLE=XXXXXXX"
Message Routing
"MSGS READY -PAGEID=XXXX -TITLE=XXXXXXX" Route=TRM1
Temporary Storage
Paging Commands
Terminal=TRM1
1. Application program prepares pages to be sent to terminal (for example, Termid-TRM1).
2. Program issues BMS paging macro instructions to write pages to temporary storage.
3. Program prepares output message to notify operator of titles of pages stored for him.
4. Program uses BMS message route macro instruction to send message to terminal-TRM1.
5. Message is sent by CICS/VS as soon as terminal is able to receive it.
6. Terminal operator later enters paging command, requesting pages in sequence desired.
Temporary Storage
Terminal=TRM1
'MSGS READY-PAGEID=XXXX TITLE=XXXXXXX'
Page 1
Page 4
Page 2

Figure 4-5. Notification of Paged Output

The task that generated the pages for display may also issue the BMS ROUTE macro instruction to send a message to the terminal notifying it of the fact that pages have been stored, and identifying the pages so that they can be displayed, when convenient, by the operator entering CICS/VS terminal paging commands. Thus the amount of information the terminal operator has to read as the result of automatic output is limited to one line, and he can use the CICS/VS paging commands to request subsequent output when he desires it. Terminal output formats should be designed to reserve at least one line on display terminals for automatic system-to-operator messages of this nature.

This second technique is based on terminal paging, which utilizes temporary storage and VSAM.

If a task is to be automatically initiated to send output to a VTAM-supported terminal such as the 3600, CICS/VS establishes a logical connection, if the relevant logical unit is not currently connected to CICS/VS. The 3600 AP controlling that logical unit is then notified of the requirement by CICS/VS to automatically initiate a task on behalf of that logical unit. This is achieved by CICS/VS requesting VTAM to send a "bid" command. On receipt of the bid, the AP can notify the terminal operator (perhaps by displaying a message or by turning on an indicator light on the 3604) that automatic output is to be sent to him. If he indicates that he can receive that output, the AP can respond positively to the bid. CICS/VS then automatically initiates the task to send data to the AP, and hence the terminal operator. If, however, the terminal operator does not wish to accept automatic output at that time, the AP can respond negatively to the bid. CICS/VS will not reissue the bid at a later time. When the terminal operator is able to accept the automatic output, he notifies the AP. The AP then transmits a "ready to receive" command to VTAM, and hence CICS/VS. CICS/VS then automatically initiates the task as discussed above. Refer to the CICS/VS Advanced Communication Guide for further information.

If none of the above techniques is to be utilized, the terminal operator can periodically enter a user transaction which reads any messages queued for that terminal destination in transient data, and transmits those messages to the terminal. This does not require the use of the techniques previously described, but has the disadvantage that it is completely dependent upon the terminal operator.

Low or High Priority Processing

CICS/VS intrapartition queues may be utilized for low (or high) priority processing. A program can receive transactions from a terminal, validate them, and notify the terminal of any error messages. Valid transactions are directed to an intrapartition destination, and queued for that destination until a specified trigger level is reached.

A terminal is not associated with this destination. When the trigger level is reached, a task is automatically initiated based upon the transaction code specified for that destination. As no terminal or operator is associated with this task, the task priority used in processing these transactions is the transaction priority as specified for that transaction code in the program control table (PCT). The initiated task may read the transactions queued to that intrapartition destination, process them, and update any required data sets depending upon the application requirements. Processing of data may then proceed independently of subsequent terminal input.

This technique is utilized by the asynchronous transaction processing (ATP) facility in CICS/VS (see Chapter 3). A batch of transactions may be entered from a batch terminal using the ATP transaction, CRDR (refer to Figure 3-11). This batch is given a batch name by the terminal operator, and each transaction is queued on a transient data intrapartition queue until all batch input is completed. At this time, a task (or tasks) is initiated, based upon the transactions in the batches, to process those batches. In the meantime, the terminal operator is free to enter any other transactions, including other ATP batches.

During processing of the ATP batches, terminal output is directed by application programs to intrapartition destinations. This terminal output may be retrieved and transmitted to the terminal, when requested by the terminal operator. This is achieved by entering the ATP transaction code CWTR (see Figure 3-11).

REUSABLE INTRAPARTITION QUEUES

Intrapartition destinations can be specified as nonreusable or reusable. Nonreusable queues accumulate data over the entire CICS/VS operational period, including any warm starts following termination of

CICS/VS (see "CICS/VS Initialization" in Chapter 8). Data on nonreusable destinations is not destroyed until transient data is cold started, or until explicitly purged by user programs.

If reusable queues are employed, when an application program issuing a transient data GET macro instruction causes data to be read from a new track, the track just read is automatically returned by transient data to the pool of tracks available for use in satisfying other PUT requests. This also causes transient data to reformat the returned track for later use, and may in some cases result in performance degradation during this reformatting.

The intrapartition data set can therefore be utilized most efficiently for those destinations for which data does not need to be retained; however, other destinations containing data which must be retained for audit or recovery purposes, are not disturbed.

INDIRECT DESTINATIONS

CICS/VS transient data uses extrapartition, intrapartition, and indirect destinations.

An indirect destination has its own destination identification, but in turn identifies another destination. Output eventually to be directed to specific devices may be written to a "logical" intrapartition destination. This logical destination identification is an indirect destination, which in turn specifies the destination for the physical device to be used to receive that output (see Figure 4-6).

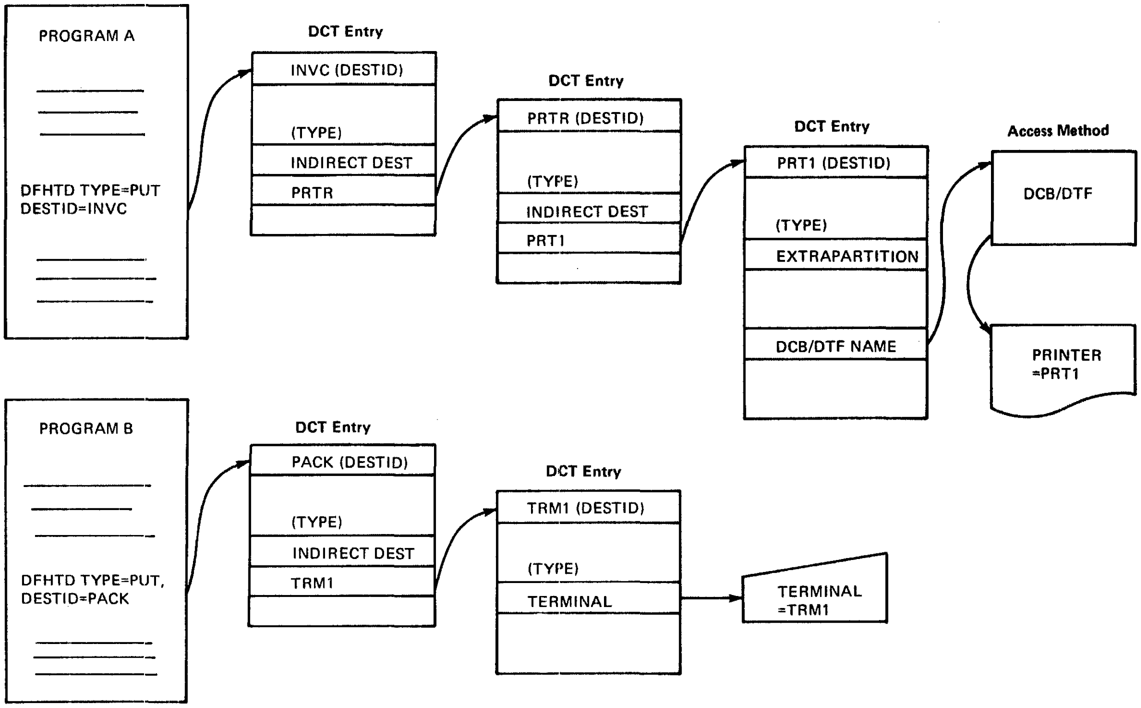


Figure 4-6. Indirect Destinations

If the output is to be subsequently directed to some other device, the application programs do not have to be changed. The output is directed to the relevant logical destination. However, the entry for that indirect logical destination is changed in the DCT to refer to the new device, which may be either intrapartition, such as a terminal, or extrapartition, such as a tape, disk, or printer.

Thus the amount of maintenance resulting from a change in the terminal network configuration, for example, is reduced to only a change and reassembly of the DCT.

Different types of output to be directed to the same terminal should be written to different logical indirect destinations. These different destinations may refer indirectly to the same terminal destination. If, at some later time it is decided to separate logical output across terminals, instead of having it appear on the same terminal, this can be achieved merely by changing the relevant indirect logical destinations to point to the new terminals to receive that output. No change need be made to the application programs.

As well as reducing the amount of program maintenance resulting from a change in the terminal network configuration or a change in application requirements directing output to different terminals, indirect destinations have other useful purposes. These are summarized below.

Device Independence

By directing output to logical indirect destinations instead of to specific terminal destinations, the programs now become independent of the particular device selected to receive that output. An indirect destination may point to any intrapartition or extrapartition destination. For example, the output which may normally be directed to a terminal printer may be directed to an extrapartition destination line printer. This can be achieved by writing the output to an indirect transient data destination, and then reassembling the DCT to point to the line printer extrapartition destination identification.

Terminal Backup

The use of indirect destinations and device independence raises the question of terminal backup. Through the use of indirect destinations, programs are no longer dependent upon the availability of specific terminals. In the event of a terminal going down, an alternative terminal or device (tape, disk, or printer) may be assigned to receive the output logically directed to the failing terminal.

On terminal failure (see Figure 3-12), it is not practical to reassemble the destination control table to change the indirect destination to the backup device, without terminating CICS/VS. In this case, the system design team should evaluate the requirement for a backup capability to enable critical information to be received.

If it is necessary that information be directed to an alternative device, then the destination control table may be changed dynamically by user-written programs. The user may write an application program (initiated by a specific transaction code) to search the destination control table for the specified indirect destination. The destination identification pointing indirectly to the failed device can then be modified to point indirectly to the destination of an alternative device. Data already queued for the original destination cannot be sent to the failed terminal, but must then be copied by the user program to the destination queue of the allocated device. Subsequent data written to the indirect destination will then automatically be directed to that device (see Figure 4-7).

Note: In the event of abnormal termination because of a power failure or machine check, and subsequent reinitiation of CICS/VS, such user modifications to the DCT may be lost. The DCT will be initialized by CICS/VS as if the user modification had not occurred, since it is not aware that the DCT was changed by the user. This can be overcome by the user program journaling each DCT modification and reestablishing each modification itself after reinitiation. (See "Journaling.")

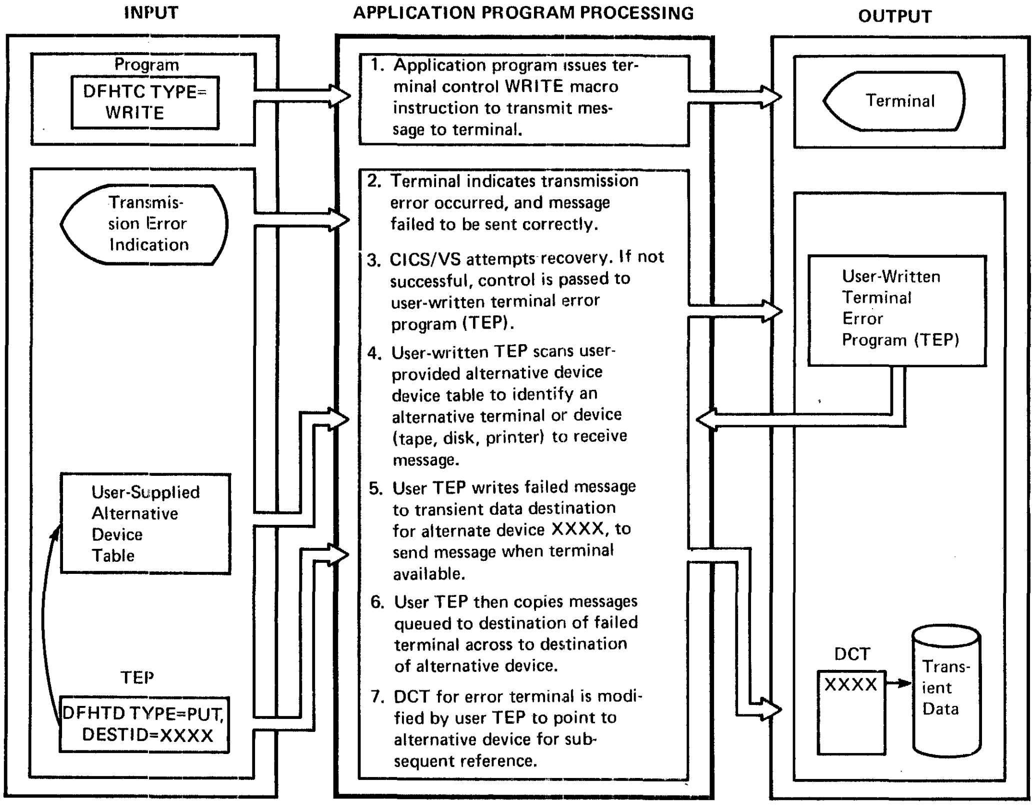

INPUT
APPLICATION PROGRAM PROCESSING
OUTPUT
Program
DFHTC TYPE=WRITE
Transmission Error Indication
User-Supplied Alternative Device Table
TEP
DFHTD TYPE=PUT, DESTID=XXXX
1. Application program issues terminal control WRITE macro instruction to transmit message to terminal.
2. Terminal indicates transmission error occurred, and message failed to be sent correctly.
3. CICS/VS attempts recovery. If not successful, control is passed to user-written terminal error program (TEP).
4. User-written TEP scans user-provided alternative device device table to identify an alternative terminal or device (tape, disk, printer) to receive message.
5. User TEP writes failed message to transient data destination for alternate device XXXX, to send message when terminal available.
6. User TEP then copies messages queued to destination of failed terminal across to destination of alternative device.
7. DCT for error terminal is modified by user TEP to point to alternative device for subsequent reference.
Terminal
User-Written Terminal Error Program (TEP)
DCT
XXXX
Transient Data

Figure 4-7. Terminal Backup and Reconfiguration

The transaction code allocated to the DCT modification program may be given a security code so that only certain authorized terminal operators, such as the master terminal operator, may use it.

Dynamic Terminal Reconfiguration

The user-written DCT Modification program for terminal backup described above may also be utilized for dynamic terminal reconfiguration. If at different times of the day it is required to change the destination of logical output to different physical devices, this can be achieved by using the DCT modification transaction code and program.

This raises the possibility of dynamically reconfiguring the terminal network, or other devices, to receive output. For example, at one time of the day output may be directed to a particular terminal printer, while at other times it may be directed to a display screen, and again, to a line printer.

As described above, any dynamic DCT modification made by user-written programs should be journaled by the user, and utilized after CICS/VS reinitialization to reestablish the modified DCT.

The availability of CICS/VS terminal device independence, which enables application programs to present output messages in a standard form regardless of the terminal type which will receive those messages, lends itself to such dynamic reconfiguration capability. Dynamic terminal reconfiguration is discussed further in "Terminal Backup" in Chapter 8.

Because CICS/VS allows any terminal (or simulated terminal such as card reader, disk, or tape) to enter any transaction, the user-developed support of dynamic reconfiguration also enables the master terminal operator to exercise control over where output is to be directed based upon online application requirements. Used in this way, transient data and indirect destinations become powerful online application tools.

Some VTAM-supported terminals (such as the 3600) permit dynamic terminal reconfiguration to be performed by the controller for the devices controlled by an AP. Through use of logical device addresses, the AP identifies devices to be used for I/O. The controller relates their logical device addresses to physical device addresses using a table associated with that AP. The controller also permits this table to be changed dynamically so that a specific logical device address may refer to a different physical device. The 3600 system operator may request this reassignment to be carried out by a user-developed AP. For example, a 3600 system operator may reassign an alternative printer for use by an AP with an inoperative printer.

This device reassignment is transparent to CICS/VS. CICS/VS communicates output disposition to an AP through use of logical device codes (LDCs). The AP then relates the logical device code to a logical device address and issues the relevant output request. The controller then relates the logical device address to a physical device as previously described.

The AP can interpret the LDC based upon application requirements and identify a logical device address to the controller. The controller can then identify the physical device currently assigned to that logical device address for that AP.

Use of alternative devices and device reassignment support in the 3601 provides additional system flexibility and availability for 3600 users.

CHAPTER 5. CICS/VS DATA BASE DESIGN

This chapter presents a tutorial discussion of data base design. Experienced CICS users may wish tc omit reading about those facilities which are identical to those provided in previous versions of CICS. New facilities provided in CICS/VS include support of VSAM data sets and DL/I data bases. Appreciaticn of these facilities can be obtained ty reading the following topics:

- Data base implementaticn for applications
- Randcm record deleticn (VSAM cnly)
- Locate mode prccessing (VSAM cnly)
- Mass record insertion (VSAM cnly)
- Skip sequential browsing
- Weighted retrieval function (VSAM only)
- Record identification
- Phonetic conversion function
- Segment updating (key-sequenced VSAM)
- Recovery considerations
- DL/I products
- Data base selection criteria

Because of the significant capability available both to the CICS/DOS/VS and the CICS/CS/VS user for accessing DL/I data bases, it is strongly recommended that the CICS/VS user whc is not familiar with DL/I concepts and advantages read the DL/I Products section in its entirety.

In designing data bases, it is important to identify the data base requirements of each application tc be implemented. Accordingly, this chapter first examines the data tase requirements cf a number of different applicaticns, to identify thcse factors most important to the applications in selecting the appropriate data base support. Follcwing this, the services cffered by CICS/VS file control and by the various DL/I products are describted. Techniques are identified for the design of data bases using these facilities to satisfy various application requirements. At the end of the chapter, a number of selection criteria are discussed for the determination of the most appropriate data base support in specific environments.

APPLICATION REQUIREMENTS OF DATA BASES

DATA BASE DEFINITION

The term "data base" may have a different meaning to different online applications or installations. A general definition of a data base, which covers most considerations, is:

> "A structured nonredundant collection of interrelated information accessible to many users at the same time."

Structures

The term "structured" in the definition refers to the organization of information in a manner by which it can be easily retrieved. The following two structuring approaches can be used:

- Physical structure
- Logical structure

To require an application program to be aware of the physical structure of the data base implies that any change to the organization of information on that data base might also necessitate modification of the application programs which access the data base.

A logically structured data base is one in which an application program can refer to information in that data base by name, without necessarily being aware of the physical organization or location of data on the data base. The physical structure or organization may be separately described by a data description table, while the application program can describe its logical accessing and usage of the data using a program description table. These tables provide an interface between the application program and the physical structure of the data base.

The advantage of logical structures is that a change in the data base generally only requires a change in the relevant tables, often without necessitating any change in the application programs. This is termed data independence, and results in reduced maintenance of programs following modification of a data base.

Data bases which are referenced by physical structure usually have limited (or no) data independence, and programs may require considerable modification following a data base change. However, programs that refer to data bases logically, exhibit a much higher degree of data independence. Any data base changes are reflected in the data base tables and program tables rather than in the program itself.

Data Redundancy

The term "nonredundant" in the above definition refers to the ability of a data base to record certain information (for example, a customer's name and address) once only, but make that information available to other programs that use it.

Traditionally, batch applications and programs are developed with their own data sets, often disregarding information that is recorded on separate data sets for separate applications. The result in the traditional batch environment is the existence of redundant information--that is, the same information is often recorded in many data sets. A change to that information must be propagated through all data sets to ensure that the information remains in step across all applications. The advantage in recording information only once, and yet making that one record of information available to all applications, is that once that information is changed, the change is reflected across all applications that use the information.

A further advantage resulting from nonredundant storage of information is storage economy, either on disk or tape.

Collection of Interrelated Information

The term "collection of interrelated information" in the definition refers to the consolidation of information relating to applications at one common point. The advantages offered by such consolidation include:

- More readily available information
- More timely information
- Elimination of redundant information
- Saving in disk or tape storage requirements
- Easier maintenance of information
- Development of information relationships

The last advantage listed refers to a significant advantage of data bases: the determination of the logical relationship of all information referring to a particular entity. The identification of such logical relationships of information enables that information to be utilized for better management of an organization's activities. This information may have been available previously, but may not have been utilized effectively before implementing the data base.

To illustrate the importance of the data base factors discussed above, namely:

- Interrelated information
- Logical structuring of information
- Data independence
- Nonredundant information

the application requirements for data base support in the following industries are discussed:

- Manufacturing
- Banking
- Insurance
- Medical
- Pharmaceutical
- Distribution
- Law Enforcement
- Utilities

The entire data base requirements will not be described for each industry. However, some data base requirements will be identified here, to use as a base for subsequent discussion in this chapter and in Chapter 11. The reader may wish to read only that section below relevant to his own industry, and then refer to the topic "Data Base Implementation for Applications."

One of the most important resources in each of the industries next discussed is data. Without access to such data or information in these industries, the following applications cannot exist. The availability of such information can open application potentials which were not practical previously.

The application design in each of these industries is discussed in more detail in Chapter 11. However, the data base requirements introduced here highlight the main requirements for data base support in each industry. References are also made to diagrams in Chapter 11.

The DL/I products provide extensive data base support, which exceeds that provided by CICS/VS file control. File control is a data management system which can be utilized to access data sets. However, the user must be aware of the physical organization of data on those data sets. DL/I refers to data in a data base, while CICS/VS file control refers to data in data sets. While recognizing the different levels of data base support offered by DL/I and by CICS/VS file control, to avoid confusion of terminology between data sets and data bases, all information is discussed below as being part of a data base, regardless of whether CICS/VS file control or DL/I is used.

MANUFACTURING INDUSTRY

Some of the information requirements of this industry relate to manufacturing (or production) work orders. The data describing the products to be manufactured, parts to be utilized, availability of materials and resources, and status and location of work orders in the manufacturing process are essential to manufacturing control. Figure 11-1 illustrates a manufacturing production order and status reporting system.

Manufacturing Work Order Data Base

At least three data bases may be required for control of information relating to manufacturing. One is the manufacturing order data base, on which the entire manufacturing application depends. This generally contains the record for each manufacturing order as illustrated in Figure 11-2. This manufacturing order data base contains information describing the manufacturing requirements for each work order, together with information reporting the status of that work order at each step in the manufacturing process.

Part Number Cross-Reference Data Base

A second data base is used to indicate relationships between part numbers and open manufacturing orders using a particular part. This data base is called the part number cross-reference data base. By accessing this data base for a particular part, each current manufacturing order which uses that part may be identified, as illustrated in Figure 11-3. Further information relating to that manufacturing order may be obtained from the manufacturing work order data base.

Manufacturing Planning Data Base

A third data base is the manufacturing planning data base, which contains information for each manufacturing order that has been planned and that will be released to the shop floor for work. This data base is used mainly for audit trail and control functions. The information contained in this data base is illustrated in Figure 11-4.

Data Base Usage

As manufacturing orders are received, they are added to the manufacturing order data base, and all parts utilized by that manufacturing work order are added to the part number cross-reference data base. As each work order is planned for manufacturing, it is entered into the manufacturing planning data base.

Data Base Requirements

The data base requirements for this application should support:

- Logical relationships of data to be established; for example, all work orders using a particular part.
- Multiple occurrences of information relating to particular data, for example, multiple status information relating to a work order, or multiple work orders relating to a part.
- Adding information to the data base, replacing or altering information, or deleting information. This enables new work orders to be added, existing work orders to be altered, or completed work orders to be deleted. In addition, one work order may be split into two or more work orders. This implies the need for changing the original work order and adding each new split work order.

Additional facilities which should, ideally, be provided by data base support in this industry are:

- Data independence
- Nonredundant information
- Easy maintenance
- Batch and online access to the data base

BANKING INDUSTRY

The applications often of interest in this industry are:

- Savings bank and mortgage loan system
- Customer information system (called Customer Information File - CIF)

The savings bank and mortgage loan system, when implemented as an online application, enables savings bank deposits and withdrawals and mortgage loan transactions to be entered from teller terminals located in various branches of the bank. These transactions are used to update savings bank and loan accounts. Figure 11-7 illustrates a typical savings bank and mortgage loan system.

The customer information system is used to identify all information relating to a customer's activities with the bank. This enables the following information to be identified for each customer:

- Savings accounts
- Checking accounts
- Loan accounts held

Furthermore, given a customer's name and account number, it can also be determined from that account all other accounts owned by that customer. Figure 11-11 illustrates a typical banking customer information system.

To implement these applications, several data bases are required. These data bases should be interrelated, to allow implementation of the customer information system. A description of these possible data bases follows.

Savings Account Data Base

The savings account data base contains information relating to each savings account--such as account number and current balance. Information describing each transaction against that account, such as deposits, withdrawals, and interest, is also contained in this data base. There may be multiple transactions against each account, as illustrated in Figure 11-8.

Mortgage Loan Data Base

The mortgage loan data base may be similar to the savings account data base, as illustrated in Figure 11-8, except that the multiple transactions that occur against a loan account are normally payments. The initial granting of a loan usually results in the creation of a new loan account, against which there may be multiple transactions reflecting periodic payments against that loan, and interest calculations based upon the current balance.

Checking Account Data Base

The checking account data base is somewhat similar to the savings account data base, except that the multiple transactions that occur against the checking account are checks written, or deposits received, together with fees charged against the account, as illustrated in Figure 11-12.

Customer Account Cross-Reference Data Base

The customer account cross-reference data base generally provides information describing the customer, such as name, address, and telephone number, and contains information describing every type of account and account number held by that customer at the bank. Because a customer can have many different types of accounts, multiple account references may exist (see Figure 11-9).

Data Base Requirements

The data base support required for this application should support:

- Multiple occurrence of transacticns relating to an account.

- Logical relationship of accounts to a specific customer, to produce an interrelated data base.

- Add new accounts, change accounts, and delete accounts from the data base.

- Add new transactions, change transactions, and delete transactions from accounts.

- Using related data tases, record custcmer information only once, but allow that informaticn tc be available tc all of the customer's accounts sc as tc avcid redundant infcrmation.

- Modify the data tase without requiring program modification--that is, data independence.

- Access the data base bcth online and offline.

INSURANCE INDUSTRY

The applicaticns within the insurance industry, covered in the following paragraphs include:

- Policy information system

- New-business policy entry system

A policy information system is somewhat analogous to a customer information system in the banking industry. It utilizes a number of data bases which describe all of the policies issued by the insurance company for each customer (see Figure 11-14). Other information relating to claims and renewals against each policy enable the insurance company to assess more accurately a customer's insurance value.

The new-business policy entry system enables new policies to te entered and added to the pclicy data tase (see Figure 11-18). In addition, provision is made through this system to alter or delete policies already in the policy data tase.

Policy Data Base

The policy data base generally contains all the information relating to each current insurance policy. In additicn, claims and renewals against that policy are associated with the policy information. There may be multiple claims or renewals against each policy, as shown in Figure 11-14.

Customer Cross-Reference Data Base

The customer crcss-reference data base contains information relating to the customer, such as name and address, together with identification of each policy number cwned (see Figure 11-16).

Representative/Territory Data Base

In some cases, a representative or a territory data base may be used. This identifies each insurance company representative, or geographic territory, together with all the policy numbers relating to that representative or territory (see Figure 11-17).

Data Base Requirements

Data base support for these applications should support the following:

- Multiple occurrences of information, such as claims and renewals to be associated with each policy
- Multiple policy numbers to be associated with each customer
- Logical relationships of a customer's ownership of various policies
- Nonredundant storage of information, so that information describing a particular customer or policy may appear only once and yet be accessible in many different ways
- Data independence, to enable the data base to be modified without requiring corresponding program maintenance
- Batch and online access to data base

MEDICAL INDUSTRY

One online application in this industry is the control and maintenance of information relating to all the patients in a hospital or clinic, in a patient information system (see Figure 11-20). This application enables a history to be developed for each patient, describing all visits, diagnoses, and treatments received for that patient. In addition, each patient receiving certain medication or with a particular disease can be noted to enable rapid identification of such patients receiving certain treatment or suffering from particular diseases.

Patient History Data Base

The patient history data base generally contains all the information describing each patient, such as name, address, sex, and physical characteristics. In this data base may be information describing each visit by each patient to the hospital or clinic, each diagnosis made for that patient, and all medication or treatment received (see Figure 11-21).

Medication Cross-Reference Data Base

The medication cross-reference data base may contain information describing the particular medication, together with identification of all patients who have received that medication, as illustrated in Figure 11-22.

Diseases Data Base

A diseases data base may be used, which contains information relating to each disease, together with identification of all patients suffering from that disease. Alternatively, the patient history data base can be searched sequentially, to select all patients with a particular disease. The use of a separate diseases data base is reccmmended if there will be a significant number of disease inquiries.

Data Base Requirements

Data base support fcr such a patient informaticn system should support:

- Multiple visits, diagncses, and treatments tc be recorded for each patient, or multiple patients to be recorded for each medication or disease
- Logical relaticnships of medication or disease to patient history and vice versa
- Add, change, or delete patients or patient history information such as visits, diagnoses, and treatment
- Add, change, cr delete identification of patients receiving certain medications cr suffering from specified diseases
- Nonredundant storage of information, such that information describing a patient occurs cnly once
- Data independence, such that a mcdification of the data base will not require corresponding modification of programs
- Access to the data base from batch and online programs

PHARMACEUTICAL INDUSTRY

One online application in this industry is the entry cf orders from pharmacists for varicus prcducts and the filling cf those orders in the pharmaceutical company's warehouse, as illustrated in Figure 11-24.

Pharmacist Data Base

This application uses a pharmacist data base, describing the information relating tc a pharmacist such as name, postal address, ship-to address, and credit rating (see Figure 11-26).

Product Data Base

The products which may be ordered are held in a product data base which describes the infcrmaticn relating to each product.

Synonym Data Base

The synonym data base may contain display images of all product names with the same first few characters (first fcur, five, six, or seven characters for example), tcgether with product information such as unit price, unit size, discounts, and warehouse location, as shown in Figure 11-25. This data base is used to identify by name the product ordered. The display image is transmitted to the terminal operator to enable the appropriate product name to be identified.

Accepted Order Data Base

The accepted order data base contains orders which are to be filled by the warehouse. The products within each order may subsequently be sequenced into warehouse location sequence for production of warehouse packing slips (see Figure 11-27).

Order In-Progress Data Base

The order in-progress data base may have the same record format as the accepted order data base (see Figure 11-27), and is used for temporary storage of products ordered, until the entire order is complete, to allow for changes in the order.

Data Base Requirements

The data bases used for this application may need to support multiple occurrences of information, such as multiple product details for products stored in more than one warehouse. The particular data base support used should provide:

- Access of information relating to pharmacists and products
- Add orders to the order data base
- Multiple occurrence of details for products in the data base which may be stored in several warehouses
- Nonredundant storage of data, with all the information relating to a pharmacist or a product, for example, stored only once
- Data independence, such that modification of the data base does not require corresponding modification of programs
- Access to the data base by batch and online programs

DISTRIBUTION INDUSTRY

A common application in the distribution industry is order entry and invoicing. This application is sometimes similar to the pharmaceutical order entry system described above, but differs mainly in the area of stock status checking. Orders are accepted against products, and the product inventory is immediately updated. Furthermore, information such as issues and receipts against each product can be retained online as part of the current product information. Figure 11-28 illustrates one example of an order entry and invoicing system.

Some of the data bases used in this application may be similar to those used in the pharmaceutical order entry system, but with additional inventory control information in the product data base.

Customer Data Base

The customer data base is generally a record of information relating to the customer, such as name, postal address, ship-to address, and credit rating (see Figure 11-30).

Product Data Base

The product data base may contain information describing the product, together with current balance, minimum balance, reorder quantity, unit price, and discounts across several warehouses (see Figure 11-31). In addition, each accepted product order may result in an issue against inventory, while each reorder placed against suppliers may eventually result in a receipt into inventory when the reordered quantity is delivered.

These issues and receipts may be recorded in separate data bases or associated with the original product in the product data base. In this case, there may be multiple issues and receipts against each product in the product data base.

Accepted Order Data Base

The orders data base may contain information describing each accepted order, and the products comprising that order (see Figure 11-29). Products in the accepted order data base may be sequenced into warehouse location sequence for production of a warehouse packing slip.

Order In-Progress Data Base

The order in-progress data base temporarily holds each product ordered until the entire order is completed, in case it is necessary to alter the order before completion (see Figure 11-29).

Data Base Requirements

The data base support requirements for this application should support:

- Accessing of customer information
- Accessing and updating of product information and inventory levels of products in the data base which may be stored in several warehouses
- Application of issues and receipts against a product in the product data base
- Addition of accepted orders to the order data base
- Nonredundant information, by associating product issues and receipts with the original product record
- Data independence, such that modification of the data base will not require corresponding modification of the programs
- Access to the data base from both batch and online programs

LAW ENFORCEMENT INDUSTRY

One online application in this industry is a police information system. This is generally an integrated data base system containing all information relating to known criminals, and data available on crimes such as *modus operandi* and suspects (see Figure 11-33).

A police information system provides a useful function, not only in the recording and maintenance of all information relating to criminals and crimes, but also through the relationships of information. A police information system can be utilized as a powerful law enforcement tool, by examining various relationships; for example:

- The personal characteristics of suspects against personal characteristics of known criminals
- Comparing the *modus operandi* used for a particular crime against the known *modus operandi* of various criminals
- The examination of all factors relevant to a particular crime and the relation of those factors to other information

A number of separate data bases may be used in this application to produce an integrated police data base. Some of these data bases are described in the following paragraphs.

Criminal Data Base

The criminal data base may contain all information relating to each known criminal, such as name, known addresses, aliases, and reference information in other data bases, such as personal characteristics and particular *modus operandi*.

Crimes Data Base

The crimes data base may contain all known information relating to particular crimes, together with details of the crime, such as *modus operandi*, and reference to possible suspects.

Suspects Data Base

The suspects data base may be similar in nature to the criminal data base, and records all known information relating to each suspect of a crime.

Convictions Data Base

The convictions data base may relate solved crimes to criminals, and may indicate the punishment dispensed.

Personal Characteristics Data Base

This data base may contain the personal characteristics (such as height, weight, age, and description) of all known criminals, suspects, and participants in crimes.

Modus Operandi Data Base

This data base may contain the *modus operandi* used by known criminals, suspects, and participants in crimes.

Data Base Requirements

The data base support requirements for this application are quite extensive. The data base support should allow:

- The multiple occurrence of information, such as multiple references to crimes and convictions for each criminal, or multiple references to criminals for each crime

- The ability to readily add, change, or delete information on the various data bases

- The ability to manipulate coded and textual information of data bases--that is, the support of variable-length text information

- The ability to identify logical relationships between various information

- The capability to search data bases, selecting information based upon specified criteria

- Nonredundant information, to reduce the amount of disk storage space required

- Data independence, such that modification of the data bases will not require corresponding modification of programs

- Access to the data bases by batch and online programs

UTILITIES INDUSTRY

One online application in this industry may be a Customer Information System (see Figure 11-35). This may contain all information relating to the utility company's customers, such as name, address, account details, appliances installed, and their maintenance history.

Customer Data Base

The customer data base contains descriptive information such as name, address, appliances installed, consumption history, account details and history, installment loan account details, and history and maintenance history for appliances (see Figure 11-36). The historical information may be part of the customer data base and requires that multiple entries of information relating to the particular account or appliance be associated with each customer.

Maintenance Technician Data Base

To enable scheduling of maintenance technicians to repair faulty appliances as reported by customers, a maintenance technician data base can be used. It may contain information describing the maintenance technician, his particular experience, a planned work schedule for that technician, and, possibly, service calls and repairs already carried out. Based upon customer service calls, and unallocated slots in a technician's schedule, technicians may be allocated to answer particular service calls. This service call information refers to the particular customer and appliance involved. Information describing the service call and repair may also be added to the maintenance history for that customer's appliance in the customer data base.

Data Base Requirements

The data base support requirements for this application should permit:

- Multiple occurrence of information, such as account history or maintenance history, to be associated with each customer or appliance, or each maintenance technician

- Ability to add, change, or delete customers, appliances, account information, and maintenance information

- Ability to generate maintenance work orders for technicians, accessing prior maintenance history for an appliance

- Ability to contain varying amounts of information for each customer, such as information relating to multiple appliances, or no appliances, and extensive account history, or no account history, while still utilizing disk storage space efficiently

- Nonredundant information

- Data independence

- Access to the data bases from batch and online programs

DATA BASE IMPLEMENTATION FOR APPLICATIONS

Data Base Requirements Summary

The most common requirements of data base support for the foregoing applications are:

- Ability to support the multiple occurrence of information, with the number of occurrences varying from zero to many

- Utilize disk storage most efficiently, without requiring storage space to be allocated for information which is not present for a particular record

- Handle variable-length information such as names, addresses, or textual information for better disk storage efficiency

- Add, change, or delete records in a data base

- Add, change, or delete multiple occurrences of information for a record

- Nonredundant storage of information

- Data independence

- Access to the data bases by batch and online programs

Multiple Occurrence Implementation

Before examining the various data base support techniques available to determine how these can satisfy the above requirements, it is particularly important to examine the way in which multiple occurrences of information for a particular data base record can be implemented. The two techniques are:

- Fhysically related cccurrences

- Logically related cccurrences

Physically related occurrences generally are implemented ty utilizing separate data sets. The main "rcct" information is stored in one data set. This may be specific customer data in a customer information system, account information in a savings bank and loan system, or product information in an order entry system.

The multiple cccurrences cf related informaticn are then stored in a separate data set cr data sets, and are related tack to the main root information in the root data set ty means of pointers. Furthermore, the separate occurrences cf information relating tc a root can be chained ty means of pointers.

For example, in the banking industry, all the accounts relating to a bank's customers may te recorded in savings and loan account data sets, with each account record ccrtaining pointers which refer back to the customer's rcot informaticn, such as name and address. Each account record for that customer may also contain a pointer to the next account for that same customer in a chain cf accounts. A further data set, a transaction data set, contains depcsits and withdrawals for accounts. Each transaction refers back to its related account record by means of a pointer, and tc the next transacticn against the same account in a chain of transactions, using another pointer. This is illustrated in Figure 5-1.

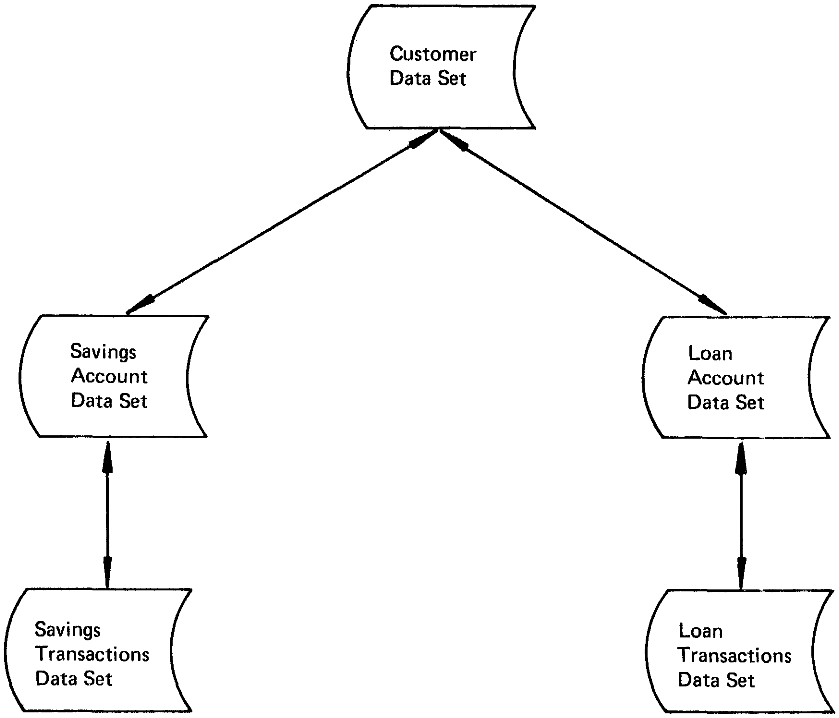


Figure 5-1. Savings and Loan Data Base Chaining

The separation of the root information in one data set, with the variable transaction information in other data sets chained logically to the root data set and also to other transactions for that same root, enables standard access methods to be utilized in providing data base support. The root information may be organized as a standard DAM (Direct Access Method), ISAM (Indexed Sequential Access Method), or VSAM (Virtual Storage Access Method) data set. Generally, the transaction data set would be organized as a DAM data set, or an entry-sequenced VSAM data set, to enable direct retrieval of transaction records. Retrieval of all the transactions relating to a particular root requires retrieval of the root information itself, followed by retrieval of each transaction in the chain--with a possible separate physical access for each transaction.

This physically related chaining technique may be supported by the CICS/VS file control indirect access feature, which is discussed in more detail later in this chapter.

The logically related technique for the multiple occurrence of information generally incorporates the multiple transactions in the same data set (or data base) with the root information. Most of the transactions relating to the root information are potentially accessible in fewer physical disk accesses than for physically related information. The data base support endeavors to place multiple transactions as close physically to their logically related root information as possible. For example, root information such as customer details is recorded immediately followed by multiple occurrences of information, each detailing a separate account for that customer and transaction activity against each particular account.

The data base support to implement a logically related technique must enable new information related to the root information to be added to other information for that root, existing information to be changed, or information to be deleted. This may require the utilization of internally controlled pointers and chains which are known only to the data base support and which are transparent to the application program. The application program may logically regard the multiple occurrences of information as if that information were physically adjacent to the root information.

Alternatively, the data base support may attempt to physically insert added information with the root and existing information, thus shifting along other information in the data base.

The data base support available for such logically related information is:

- CICS/VS file control segmented record feature

- DL/I products

These are discussed in more detail in the remainder of this chapter.

DATA BASE SUPPORT FOR CICS/VS

The features and system design aspects of CICS/VS file control and DL/I products are presented below. Included are the factors to consider when selecting appropriate data base support and selection criteria.

CICS/VS FILE CONTROL (ISAM, DAM, VSAM)

The CICS/VS file ccntrcl prograш provides data tase support for application programs executing under its control. It uses the standard access methods available under DCS/VS and OS/VS1 or OS/VS2 -- namely the Indexed Sequential Access Method (DOS/VS ISAM cr OS/VS BISAM), Direct Access Method (DOS/VS DAM or OS/VS BDAM), and Virtual Storage Access Method (VSAM). Fcr the remainder of this chapter, "DAM" will be used to refer both to DOS/VS DAM and OS/VS BDAM, and "ISAM" will refer to both DOS/VS ISAM and OS/VS BISAM.

The facilities provided by the standard access methods are extended in some cases by CICS/VS file control to provide additional support. For example, file ccntrcl supports the following data sets:

- Fixed-length and variable-length records
- Blocked and unblocked data sets
- ISAM, DAM, and VSAM

Extensions provided by CICS/DOS/VS file ccntrol enable the support of variable-length DOS/VS ISAM data sets, which are not part of standard support provided by DOS/VS ISAM. Similarly, file control provides support for blocked fixed-lergth or variable-length DAM data sets, which are not included in the standard support provided by DOS/VS DAM or OS/VS BDAM. The suppcrt cf blccked direct access data sets is particularly useful if those data sets are processed sequentially by CICS/VS programs, as discussed below in "Sequential Access (Browsing)." CICS/VS file ccntrcl allcws toth direct access and sequential access to ISAM, DAM, and VSAM data sets.

DIRECT ACCESS

Direct access, scmetimes referred to as random access, is supported by file control for ISAM, DAM, and VSAM data sets. The following services are provided by CICS/VS file control for DAM, ISAM, and VSAM:

- Randcm reccrd retrieval
- Random record update
- Randcm record additicn
- Random record deleticn (VSAM only)
- Logically open/close data sets
- Exclusive control of reccrds during update operations
- Variable-length ISAM reccrds (both DCS/VS and OS/VS)
- Blocked DAM records
- LOCATE mode, read-only retrieval (VSAM only)
- Mass record insertion (VSAM cnly)
- Segmented records
- Indirect access

These services enable CICS/VS file ccntrol to provide data management support that surpasses OS/VS or DOS/VS data management support in many areas.

Direct access to data sets is made on the basis of record identification of the particular logical record to be retrieved. The record identification may be either a record key in the case of ISAM or key-sequenced VSAM data sets, or a record location within the data set for DAM or entry-sequenced VSAM data sets. The use of record keys or locations for direct access is discussed in more detail under "Record Identification."

Based upon presentation of the appropriate record identification by the application program, CICS/VS file control will access the data set requested by the program to carry out the services listed above, and described in detail in the following sections.

Random Record Retrieval

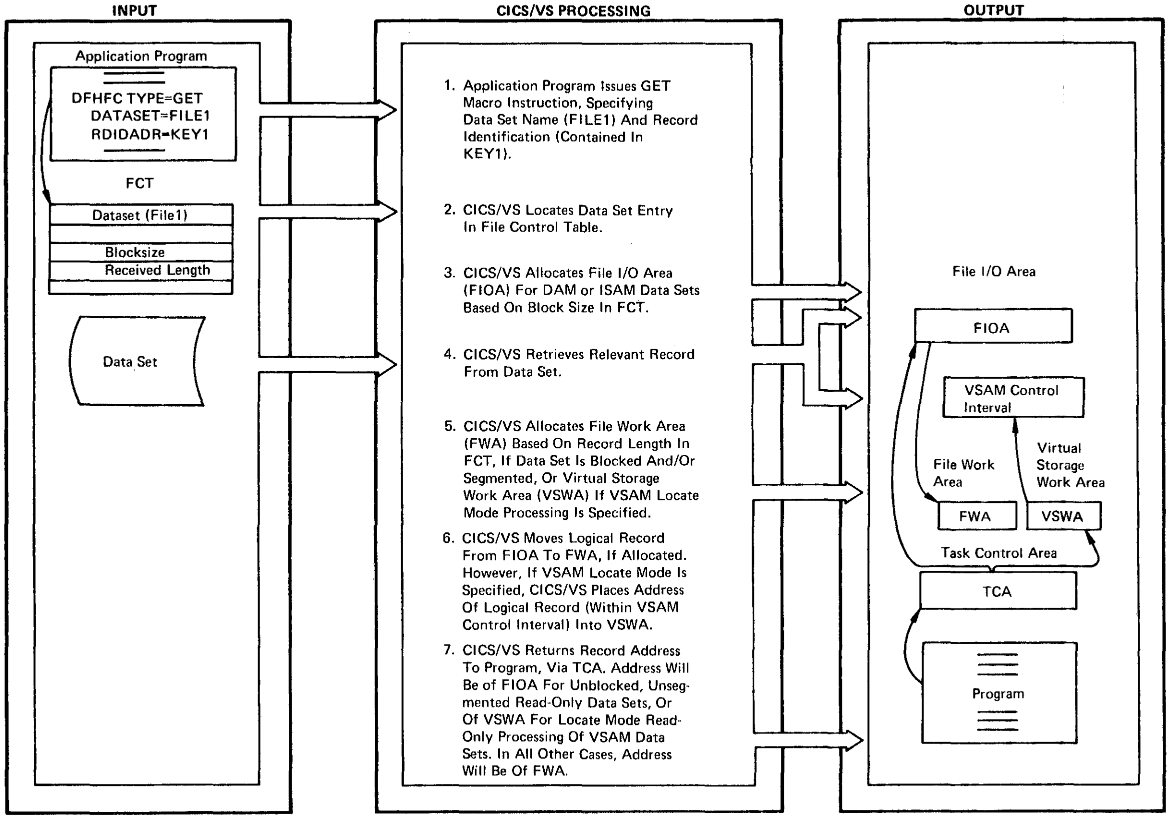
File control will directly access the record identified by the application program using either key or record location (depending upon the type of data set) from the specified data set. The application program issues a file control GET macro instruction, identifying by name the data set to be accessed, and indicating the location in the program which contains the record identification. The data set name is used by CICS/VS to locate the relevant entry for that data set in the file control table (FCT). This entry contains specifications for that data set, such as:

- Access method used
- Record length
- Block length
- Key length (if applicable)
- Key location (if applicable)

This information is not contained within the application program. In the event of a change to the data set, the relevant changes may be made to the FCT, without affecting the application program. This provides a limited degree of data independence.

When the application program issues a file control GET macro instruction, CICS/VS dynamically allocates storage to be used as an input area (file I/O area--FIOA), and a work area (file work area--FWA, or virtual storage work area--VSWA--for locate mode processing of VSAM data sets) if required. The input operation then begins. The application program waits until that requested operation is completed. Any I/O errors on completion, which cannot be recovered by the access method or by file control, are then returned to the application program for action. See Figure 5-2.

Although an application program does not continue processing while a requested I/O operation is being carried out, CICS/VS utilizes the available processing time during the I/O for other concurrently executing tasks. Consequently, all tasks are given an equal opportunity to process, based upon their respective task priorities, while I/O is in progress. The net result is improved overall performance of all concurrently executing tasks in the system, even though the full processing overlap potential of the single task issuing the I/O operation request is not utilized.



1. Application Program Issues GET Macro Instruction, Specifying Data Set Name (FILE1) And Record Identification (Contained In KEY1).
2. CICS/VS Locates Data Set Entry In File Control Table.
3. CICS/VS Allocates File I/O Area (FIOA) For DAM or ISAM Data Sets Based On Block Size In FCT.
4. CICS/VS Retrieves Relevant Record From Data Set.
5. CICS/VS Allocates File Work Area (FWA) Based On Record Length In FCT, If Data Set Is Blocked And/Or Segmented, Or Virtual Storage Work Area (VSWA) If VSAM Locate Mode Processing Is Specified.
6. CICS/VS Moves Logical Record From FIOA To FWA, If Allocated. However, If VSAM Locate Mode Is Specified, CICS/VS Places Address Of Logical Record (Within VSAM Control Interval) Into VSWA.
7. CICS/VS Returns Record Address To Program, Via TCA. Address Will Be of FIOA For Unblocked, Unsegmented Read-Only Data Sets, Or Of VSWA For Locate Mode Read-Only Processing Of VSAM Data Sets. In All Other Cases, Address Will Be Of FWA.

Figure 5-2. CICS/VS File Control Random Record Retrieval

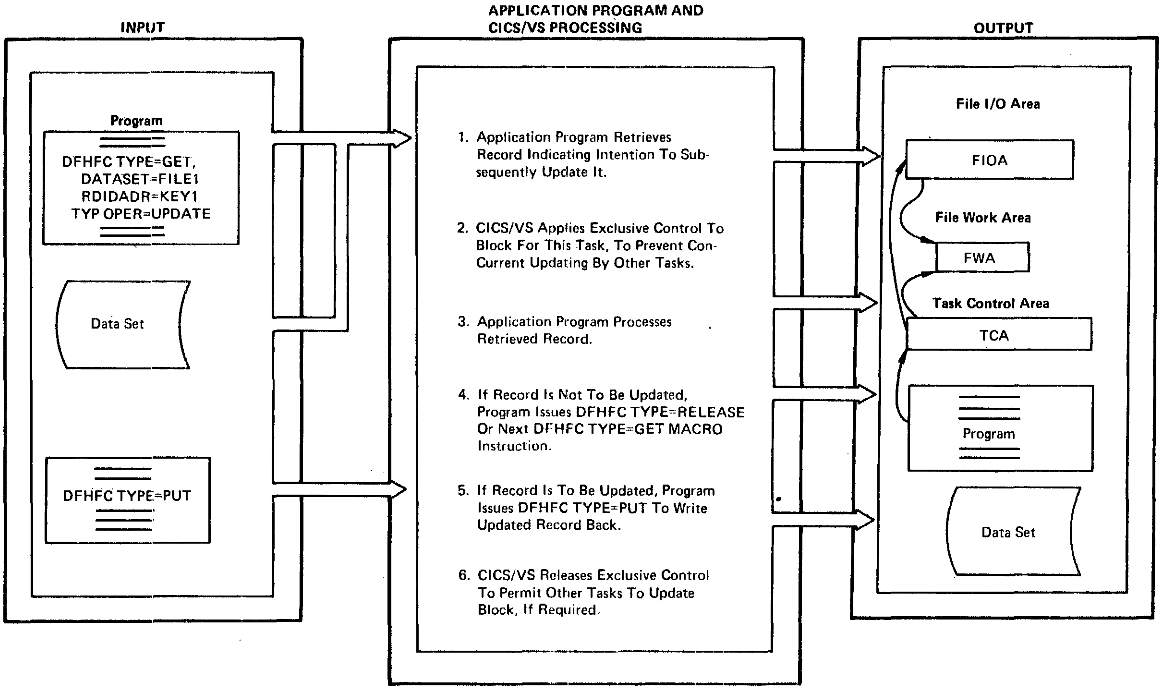
A file I/O area (FIOA) is dynamically allocated before the GET operation commences. On completion of that I/O operation without error, in the case of blocked and/or segmented data sets a file work area (FWA) is allocated of sufficient size to contain a logical record. The requested logical record is then located in the block in the FIOA, and transferred across to the FWA. For locate mode processing of VSAM data sets, a VSWA is allocated, and information identifying the record in the control interval is placed in the VSWA. This is discussed in more detail in "Locate Mode Processing (VSAM Read-Only)" later in this chapter. The application program is then presented with the address of the FWA or VSWA, or of the FIOA for an unblocked unsegmented read-only data set. The application program may then process the logical record.

When the record has been processed and is no longer required, the FIOA and FWA or VSWA (if allocated) can be dynamically released by the application program, and the storage utilized by these areas may be returned to the CICS/VS dynamic storage area for use in satisfying other storage requests. Figure 5-2 illustrates the function of CICS/VS file control random record retrieval.

Random Record Update

A record can be directly accessed using a file control GET macro instruction, as described above, for potential subsequent update. An indication that this record may subsequently be updated is made by the application program at the time that the GET macro instruction is issued, by indicating that the type of operation is a GET for UPDATE.

In this case, the record is retrieved as described above for the GET macro instruction. After the application program has updated the logical records in the FWA, it issues a PUT macro instruction, supplying to CICS/VS the address in storage of that FWA. CICS/VS file control determines the fact that this is a PUT of a record which was earlier retrieved for update. The logical record then replaces the original record on disk (see Figure 5-3).



1. Application Program Retrieves Record Indicating Intention To Subsequently Update It.
2. CICS/VS Applies Exclusive Control To Block For This Task, To Prevent Concurrent Updating By Other Tasks.
3. Application Program Processes Retrieved Record.
4. If Record Is Not To Be Updated, Program Issues DFHFC TYPE=RELEASE Or Next DFHFC TYPE=GET MACRO Instruction.
5. If Record Is To Be Updated, Program Issues DFHFC TYPE=PUT To Write Updated Record Back.
6. CICS/VS Releases Exclusive Control To Permit Other Tasks To Update Block, If Required.

Figure 5-3. CICS/VS File Control Random Record Update

If the application program does not wish to update the record which was retrieved, it does not issue a PUT macro instruction. The application program should issue a File Control RELEASE macro instruction.

Exclusive Control During Update

If the exclusive control feature was specified when the file control management routine was generated for the installation, file control will ensure that no other concurrently executed task is able to issue a GET with UPDATE macro instruction for the same logical record for ISAM data sets, physical record for DAM data sets, or control interval for VSAM data sets (referred to as the "physical record" below). This is necessary in a multitasking environment to avoid two or more concurrently executing tasks updating the same physical record on disk, with the possibility of losing information resulting from one or more concurrent updates. However, a GET request without update may be concurrently issued with a GET request for update, for the same physical record. Several tasks may read that record at the same time, but only one task is permitted to update it.

If it is not necessary to update the record retrieved, exclusive control on that physical record can be released by issuing a file control RELEASE macro instruction (see Figure 5-3). This will permit any other waiting task which also wishes to update the same physical record to commence its update, at an earlier time than it could if the application program did not issue a RELEASE macro instruction.

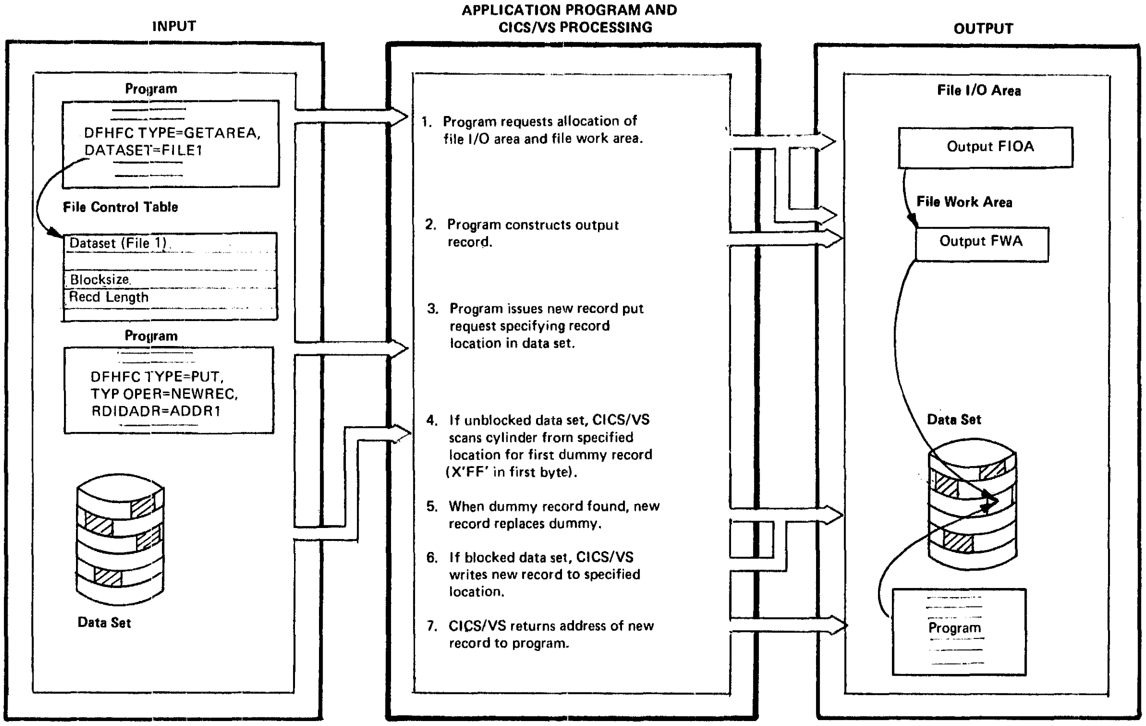
Random Record Addition

Records may be added to a data set through the use of a File Control PUT macro instruction indicating that the type of operation is a new record addition. In this case, the application program must first request that a file work area (FWA) be allocated to enable the new record to be constructed in main storage. The allocation of an FWA is achieved by issuing a file control GETAREA macro instruction specifying the data set name to which the record will be subsequently added. The data set name is used to locate the appropriate entry in the FCT and so determine the record length to be used by CICS/VS in allocating the FWA.

After constructing the new record in the allocated FWA, the application program issues a PUT macro instruction, specifying that the type of operation is the addition of a new record. The record identification supplied by the program is used to determine where the new record will be added.

If the record identification provided is a record key for addition of new records to ISAM or key-sequenced VSAM data sets, the record is placed in sequence in the data set based upon that key. For DAM data sets, the new record is inserted as close as possible to the specified record location as described below.

For fixed-length unblocked DAM records, such data sets must be initially generated with a number of dummy records interspersed throughout the data set. A dummy record is one containing hexadecimal FF in the first byte of the record. The record to be added is inserted in the first available dummy record location following the specified record location. If no dummy records are available in the same cylinder (for DOS/VS), the application program is notified; it may then reissue the PUT request for the new record to another part of the data set until a dummy record is found. When the new record replaces the dummy record, file control returns the record location where the new record is stored to the application program (see Figure 5-4).



1. Program requests allocation of file I/O area and file work area.
2. Program constructs output record.
3. Program issues new record put request specifying record location in data set.
4. If unblocked data set, CICS/VS scans cylinder from specified location for first dummy record (X'FF' in first byte).
5. When dummy record found, new record replaces dummy.
6. If blocked data set, CICS/VS writes new record to specified location.
7. CICS/VS returns address of new record to program.

Figure 5-4. CICS/VS File Control Addition to Fixed-Length DAM Data Set

For variable-length record DAM data sets, CICS/VS file control attempts to add the new record at the end of the specified track, for CICS/DOS/VS, providing there is sufficient space on that track to contain it. For CICS/OS/VS, a specified number of tracks may be searched to locate a track on which to add the record. If there is not sufficient space, the application program is notified, and may reissue the PUT request for the new record, indicating another track to be used. When the new record has been successfully written at the end of the specified track, its record location is returned to the application program (see Figure 5-5).

For entry-sequenced VSAM data sets, new records are always added to the end of the data set regardless of whether they are fixed or variable-length. The relative byte address of the added record in the data set is returned to the application program.

Random Record Deletion (VSAM Only)

The file control DELETE macro instruction is used to specify the deletion of records in a VSAM key-sequenced data set. The specified record is physically deleted. The space occupied by that record is reclaimed and added to the available free space in the particular control interval which contained that deleted record.

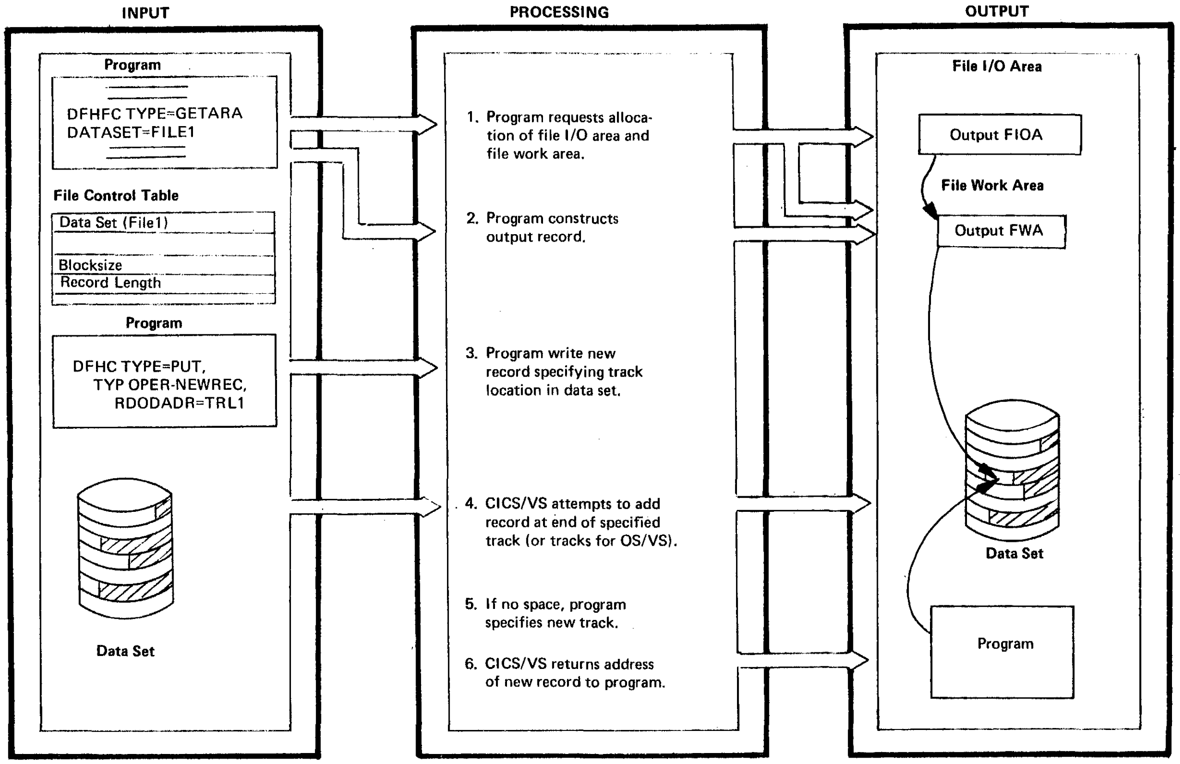

INPUT
PROCESSING
OUTPUT
Program
DFHFC TYPE=GETARA
DATASET=FILE1
File Control Table
Data Set (File1)
Blocksize
Record Length
Program
DFHC TYPE=PUT,
TYP OPER-NEWREC,
RDODADR=TRL1
Data Set
1. Program requests allocation of file I/O area and file work area.
2. Program constructs output record.
3. Program write new record specifying track location in data set.
4. CICS/VS attempts to add record at end of specified track (or tracks for OS/VS).
5. If no space, program specifies new track.
6. CICS/VS returns address of new record to program.
File I/O Area
Output FIOA
File Work Area
Output FWA
Data Set
Program

Figure 5-5. CICS/VS File Control Addition tc Variable-Length DAM Data Set

Locate Mode Processing (VSAM Read-Cnly)

The normal mode of processing for file control operations is move mode. With mode processing of blocked data sets, the logical record is moved from the blcck into a FWA, and the address of that FWA is presented to the applicaticn program.

For VSAM data sets, locate mode processing may be specified for read-only operations. With locate mcde processing, the address of the logical record in the control interval is stored in a virtual storage work area (VSWA). The additional CPU processing required to move the logical record from the control interval is therefcre avoided. However, lccate mcde is invalid if a read fcr update is specified and/or segmented records are being retrieved.

Blocked DAM Records

CICS/VS file control provides for the deblocking of logical records in a blocked direct access (DAM) data set. This service is provided for both fixed-length and variable-length records. When creating or adding to blocked DAM data sets, the application program must work with entire blocks.

The advantage in supporting blocked DAM records is to enable both direct and sequential access of the data set. The block size should be such that the physical reccrd retrieved for direct access is maintained as small as possible, while still providing sufficient blocking to enable satisfactory performance for sequential retrieval.

DOS/VS ISAM Variable-Length Records

CICS/DOS/VS supports the retrieval and static update (that is, no length variation) of variable-length records within a fixed-length block under ISAM organization. These pseudovariable blocks must contain the block length in the first four bytes in the standard form LLbb. Since all blocks are fixed-length, this value is the same for all blocks. Each logical record within the block must also reflect the length of the record in the first four bytes (LLbb). A logical record may not be continued into the next block. The first byte of any unused portion of a block must contain a hexadecimal FF.

The addition and deletion of records for a DOS/VS ISAM variable-length record data set must be handled by the user in an offline batch environment. When creating the data set, it must be defined as fixed unblocked, and the key for each block must be the same as the last logical record in that block. The block size must be an even number of bytes. All records must reside in the prime data area; no overflow records are allowed.

However, the use of key-sequenced VSAM data sets instead of ISAM allows the support of both fixed-length and variable-length records, with the added advantage that the record length can be either increased or reduced as a result of a record update, addition, or deletion.

Dynamic OPEN/CLOSE of Data Sets

When the CICS/VS system is initialized, data sets may be specified in the file control table (FCT) as either open or closed. Closed data sets may be dynamically opened for accessing at a later time, by means of a master terminal command. The design techiques described below utilize dynamically opened or closed data sets.

Data sets may be dynamically closed at certain times of the day, to prevent access to information from terminals, and may be dynamically opened when access is to be permitted. In this way, support for certain online applications may be provided only when desired. If an application program attempts to access a data set which has been closed, an error indication is returned to the program.

Mass Record Insertion (VSAM Only)

When adding records to a key-sequenced VSAM data set, significant performance advantages can be realized if many records have to be added, and if those records are added in the same sequence as the original data set. This is referred to as mass record insertion. The application program specifies the mass insert operation in a GETAREA macro instruction. This indicates to the file control program that the user intends to submit several successive PUT requests for new logical records with keys that are in ascending sequence. VSAM then performs the addition of the new records faster than if the additions were made in random sequence.

Each subsequent PUT macro instruction utilizes the same FWA as each record is added. The mass insert operation is terminated by issuing a file control RELEASE macro instruction.

VSAM Shared Resources (CICS/OS/VS Only)

VSAM shared resources enable a pool of I/O related blocks, channel programs, and buffers to be shared among several VSAM data sets. This permits efficient utilization of storage in an environment in which many VSAM data sets are open and it is difficult to predict the amount of activity against a given data set, or in a situation where each transaction may access several VSAM data sets.

The user indicates in the FCT which VSAM data sets are to share resources. CICS/VS calculates the maximum amount of resources required by using the number of strings specified in the FCT for each of the VSAM data sets that are to share resources and the control interval sizes for these data sets from the VSAM catalog. CICS/VS then requests VSAM to build a resource pool large enough for a certain percentage of maximum amount of resources required. The user can override this percentage and resource calculation if desired. For example, the user may wish to override the CICS/VS calculation to reflect specific data set activity known only to the user.

Storage utilization efficiency obtained by sharing VSAM resources must be evaluated against the effect on performance. If insufficient resources are available to satisfy a specific I/O request against a shared resource data set, the requesting task is placed in a CICS/VS wait until the necessary resources become available. CICS/VS provides statistics (number of strings, buffer sizes, and number of buffers of each size) to identify the resources allocated. Statistics are also provided to aid in the optimization of these resources to ensure that sufficient buffers and VSAM strings are available to avoid excessive task wait time. (See "CICS/VS Working Set" in Chapter 7.)

If the activity against specific data sets is higher than can be managed using shared resources, those data sets should be defined in the FCT as not sharing resources.

SEQUENTIAL ACCESS (BROWSING)

The operations discussed above refer to direct access. CICS/VS file control enables DAM, ISAM, and VSAM data sets to be sequentially as well as directly accessed. This sequential access is sometimes referred to as a "browse" operation. Data sets to be browsed may be either fixed-length or variable-length, blocked or unblocked data sets.

A browse operation using CICS/VS file control is analogous to the sequential retrieval of records from ISAM data sets, sometimes called SETL retrieval, in a batch environment. However, a batch program can only sequentially retrieve records from one logical section of a data set at a time. On the other hand, CICS/VS enables many browse operations to be concurrently executed on the same data set, either from the one task or several tasks. This is referred to as multiple browsing, and is discussed further below.

Browse Initiation

To specify a browse operation, the application programmer identifies the data set to be browsed, and provides the record identification of the logical starting point in the data set for the browse operation. This logical starting point can be either a specified record location, or key, or a generic key. For example, if it is desirable to browse an orders data set, containing orders for products placed by different branches, a generic key may indicate that browsing is to start with the first order recorded from a specified branch. The initiation of a browse operation is achieved by the application program issuing a file control SETL macro instruction.

Browse Retrieval

Each record is sequentially retrieved for the browse operation when the application program issues a GETNEXT macro instruction. Each GETNEXT macro instruction presents the next sequential logical record to the application program for processing. In the case of an ISAM data set or a key-sequenced VSAM data set, the records are presented in ascending key sequence (except for a browse operation using relative byte address (RBA) for a key-sequenced data set, when records may be presented in physical sequence). For a DAM data set, or an entry-sequenced VSAM data set, the records will be presented in the sequence in which they are physically stored on the data set.

Browse Termination

The browse operation continues with each subsequent GETNEXT macro instruction, until the end of the data set is reached or it is desired to terminate the browse operation. This termination is achieved by issuing a file control ESETL macro instruction.

When a browse operation is initiated by a SETL macro instruction, a file work area (FWA) and file I/O area (FIOA), or a virtual storage work area (VSWA) for VSAM data sets, is allocated for that browse. Each subsequent record read as the result of a GETNEXT macro instruction is presented to the application program in this FWA. When the ESETL macro instruction is issued to terminate the browse, the FWA and FIOA or VSWA are released.

Multiple Browsing

A task can issue one or more SETL macro instructions to initiate one or more browse operations. Each SETL macro instruction results in the allocation of an FWA and FIOA or VSWA for that browse operation, and this FWA or VSWA contains the current record identification of the logical point reached in the data set. When several SETL macro instructions are issued by the one task against the same data set, several FWAs and FIOAs, or VSWAs will be allocated, one FWA and FIOA or VSWA for each browse. The task may maintain several concurrent browse operations, by indicating the address of the FWA or VSWA for the logical section of the data set to be browsed next. The GETNEXT macro instruction issued will read a record from that logical section of the data set and present it to the application program in the relevant FWA or VSWA. Figure 5-6 illustrates multiple browse operations.

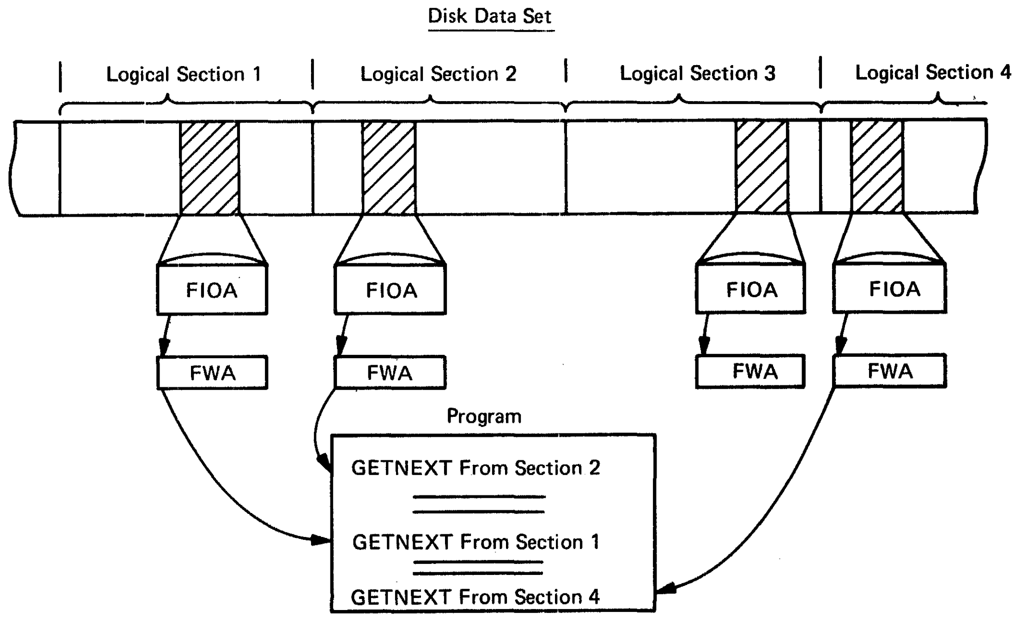


Figure 5-6. Multiple Browse Operations Using CICS/VS File Control

There is no logical limit to the number of browse operations that may be executed concurrently for the same data set, either from the same task or many tasks. The only limitation is the availability of dynamic storage to maintain an FIOA and FWA or VSWA for each concurrent or multiple browse. This is a factor of the block length, record length, degree of multitasking, and amount of dynamic storage allocated in the CICS/VS partition, and number of VSAM strings specified for a VSAM data set.

The multiple browse technique introduces a number of very useful system design solutions. For example, an orders data set may contain orders that are in sequence according to product number as placed from several branches within a company. If it is desired to retrieve all product orders from a specific branch, this can be achieved by issuing a browse operation, starting the browse at the first product number ordered from that branch. Subsequent GETNEXT macro instructions will retrieve the next product ordered from the branch, until the end of all products ordered from that branch is reached. At this time the browse operation may be terminated by an ESETL macro instruction.

However, if the product orders received from several branches are reported using a terminal, this may imply that the orders data set should be sorted into the sequence of branch number within product number.

Generally, online sorting is impractical. Records should be retrieved in the sequence of branch within product, while still maintaining the orders data set in the sequence of product within branch. This can be achieved by issuing multiple browse operations, having each browse initiated from the first product order record from each branch. Each browse operation in effect logically breaks up the orders data set into a number of separate order data sets, one for each branch.

A GETNEXT request can be issued for each branch browse operation to retrieve the orders placed for the first product number in the small logical data set for each branch. The second GETNEXT macro instruction issued for each browse operation then retrieves the next product order record for each branch.

This can continue, retrieving all the information from each branch relating to a specified product until the application program has constructed an entire terminal page. At this time, the browse operations for the products and branches contained on that terminal page may be terminated by issuing an ESETL macro instruction for each browse.

As previously stated, the technique of multiple browsing enables the sequential retrieval of information in a sequence different from that in which a data set is organized. This multiple browsing design technique may open up powerful data inquiry possibilities for online data sets.

Skip Sequential Browsing (VSAM Only)

Skip sequential refers to the ability to sequentially browse through a logical section of a data set, and then skip to another logical section of a data set to continue the same browse operation. In effect, it provides a direct access capability in the middle of a sequential retrieval operation. The record identification of the next logical section of the data set may be moved into the record identification field set up in the program, and another GETNEXT macro instruction can be issued. This will position the browse to the new section of the data set, thus effecting a skip sequential operation. This technique cannot be used for DAM or ISAM data sets.

Weighted Retrieval Function (VSAM Only)

This facility is provided as a built-in application function and can be used only for key-sequenced VSAM data sets. It enables a data set to be searched by CICS/VS and records to be extracted from that data set based upon selection criteria. These criteria may be specified either by the application program, or by the terminal operator, to be used by the application program.

This provides a powerful information retrieval capability to CICS/VS, so that records may be retrieved and fields within those records may be matched against information provided by the application program or terminal operator. Records may be selected based upon an exact match with the selection criteria, or a match within a specified range of the selection criteria. Figure 5-7 illustrates the concept of weighted retrieval.

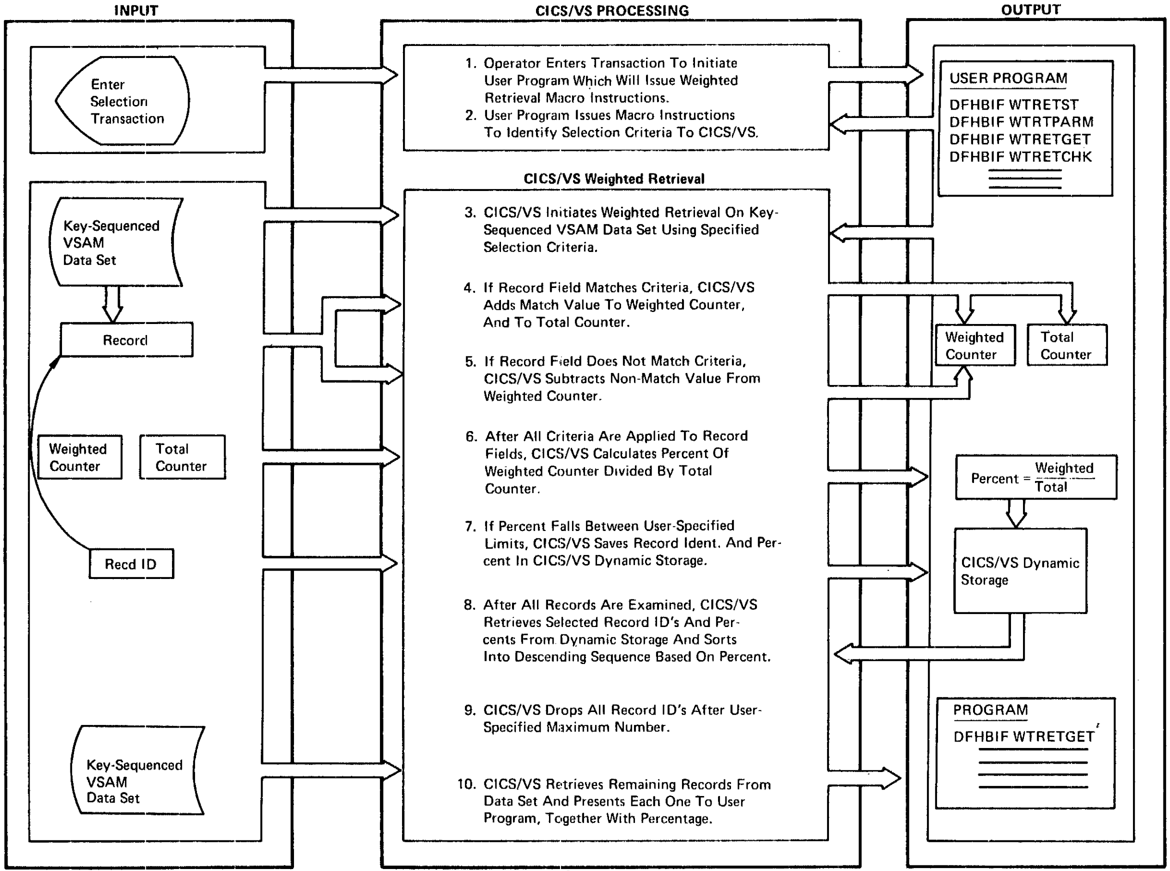

INPUT
CICS/VS PROCESSING
OUTPUT
Enter Selection Transaction
1. Operator Enters Transaction To Initiate User Program Which Will Issue Weighted Retrieval Macro Instructions.
2. User Program Issues Macro Instructions To Identify Selection Criteria To CICS/VS.
USER PROGRAM
DFHBIF WTRETST
DFHBIF WTRTPARM
DFHBIF WTRETGET
DFHBIF WTRETCHK
CICS/VS Weighted Retrieval
Key-Sequenced VSAM Data Set
3. CICS/VS Initiates Weighted Retrieval On Key-Sequenced VSAM Data Set Using Specified Selection Criteria.
4. If Record Field Matches Criteria, CICS/VS Adds Match Value To Weighted Counter, And To Total Counter.
Record
Weighted Counter
Total Counter
5. If Record Field Does Not Match Criteria, CICS/VS Subtracts Non-Match Value From Weighted Counter.
Weighted Counter
Total Counter
6. After All Criteria Are Applied To Record Fields, CICS/VS Calculates Percent Of Weighted Counter Divided By Total Counter.
Percent = Weighted / Total
7. If Percent Falls Between User-Specified Limits, CICS/VS Saves Record Ident. And Percent In CICS/VS Dynamic Storage.
Recd ID
CICS/VS Dynamic Storage
8. After All Records Are Examined, CICS/VS Retrieves Selected Record ID's And Percents From Dynamic Storage And Sorts Into Descending Sequence Based On Percent.
9. CICS/VS Drops All Record ID's After User-Specified Maximum Number.
PROGRAM
DFHBIF WTRETGET
Key-Sequenced VSAM Data Set
10. CICS/VS Retrieves Remaining Records From Data Set And Presents Each One To User Program, Together With Percentage.

Figure 5-7. CICS/VS Weighted Retrieval

Associated with each selection criterion are both a match value and a nonmatch value. In addition, counters are maintained for each selection criterion to accumulate statistics relating to the degree of matching achieved. In the event of an exact match, or a match within specified limits, a match value associated with that criterion is added to a weighted counter and to a current total counter. In the event of a match not being achieved, a nonmatch value is subtracted from the weighted counter, but not from the current total counter. The match and nonmatch values may be either positive or negative, but they must have the same sign.

If a particular criterion is matched, the weighted counter is increased appropriately. However, if another criterion is not matched, the weighted counter may be decreased. After all selection criteria have been applied to the necessary fields in the record, a percentage of acceptability is calculated by dividing the value of the total counter into that of the weighted counter. If this percentage falls within defined limits for the weighted retrieval operation, the full key identification of that record and the percentage are saved in CICS/VS dynamic storage by the built-in weighted retrieval function.

After all required records are examined, the keys and percentages are read back from dynamic storage and sorted into sequence based upon the percentage of compliance of each record with the selection criteria.

If the number of keys satisfying the selecticn criteria exceeds a maximum, say N, specified by the application program, then all keys having a percentage equal to or lower than that cf the N+1th key are dropped. Follcwing this, the remaining records are retrieved and made available to the application program one at a time in order of decreasing acceptability, together with each reccrd's percentage. Refer to the *CICS/VS Application Programmer's Reference Manual* for more detailed information about the use of weighted retrieval.

Application Uses for Weighted Retrieval Function

The weighted retrieval functicn uses the browsing capability of CICS/VS file control and is applicable only to key-sequenced VSAM data sets. Use of this weighted retrieval capability opens significant online application cpportunities in a number of industries. For example, in a manufacturing industry, weighted retrieval may be used to search a key-sequenced VSAM wcrk crder data set to identify specified part numbers, quantities, revenue, and completion periods for all work orders. Alternatively, a VSAM manufacturing planning data base can be searched to identify all wcrk orders planned to be manufactured by specified equipment cr with particular materials or parts.

In the banking industry, the weighted retrieval function can be a powerful tool. It enables all customer records cf various branches of the bank to be examined. Thcse with a current balance above or below a specified amount for certain types of accounts, can be selected. Alternatively, Weighted retrieval can be used to provide an exception report of all checking accounts with an overdraft greater than a specified amount.

Weighted retrieval can be used in the insurance industry to search key-sequenced VSAM policy data bases. It can identify those pclicies with claims exceeding a certain value for specified types of policies in particular geographic locations or owned by a particular class of policyholder.

In the medical industry, a patient information system can use the weighted retrieval function to identify all patients receiving particular medication in specified quantities over a particular period of time. Alternatively, all patients with a particular combination of symptoms may be selected.

Weighted retrieval can also be used in law enforcement agencies to search a key-sequenced VSAM criminal data base tc select all criminals with specified perscnal characteristics and *modus operandi* that compare with characteristics and *modus operandi* of participants in a specific crime. Alternatively, a key-sequenced VSAM crimes data base can be searched, selecting those crimes with the same *mcdus operandi*, and using this *modus operandi* to select those criminals known to use that *modus operandi* from a criminal data base. The reccrds of these identified criminals can be further searched to select criminals who satisfy cther criteria identified by the nature cf a particular crime. Using weighted retrieval in this application beccmes a powerful tool for crime analysis and the identification of criminals or suspects associated with thcse crimes.

As shcwn by previous examples, the application potentials offered by the use of the built-in weighted retrieval function can be significant. In fact, utilizing weighted retrieval may be an important consideration in determining the data base support tc be used for the particular informaticn tc be retrieved.

The weighted retrieval functicn can only be used with key-sequenced VSAM data sets. As will be seen later in the discussion of the DL/I products, DL/I DOS/VS uses VSAM. IMS/VS DL/I may use either VSAM or BISAM and BDAM. A DL/I VSAM data base utilizing root segments only can be accessed as a standard VSAM data set and operated upon by the weighted retrieval function.

For effective online performance, VSAM data sets (and, hence, weighted retrieval) should not be used with systems having less than 144K of real storage. This is discussed further in "Data Base Selection Criteria" at the end of this chapter.

RECORD IDENTIFICATION

As discussed above, data sets supported by CICS/VS file control can be accessed either directly or sequentially. Records are accessed based upon the record identification supplied by the application program. The record identification utilized depends upon the particular data set being accessed. There are two types of record identifications:

- Record key
- Record location

Record Key

Record identification based upon a key is used to access ISAM data sets and key-sequenced VSAM data sets. The key may be either a full key for retrieval of a particular logical record, or a partial (generic) key, to indicate a logical point in a data set from which a browse operation is to commence. This generic key contains sufficient information in the high-order bytes of the key to uniquely identify the logical section of the data set. The remaining low-order bytes of the key may be either binary zeros or blanks. For key-sequenced VSAM, a truncated generic key may be utilized, with the first byte of the key specifying (in binary) the number of significant bytes in the generic key which follows.

For instance, the orders data set discussed above for browsing can utilize a key containing a branch number in the high-order bytes of the key, and specific product numbers in the low-order bytes of the key. For example, orders for product number 1016 from branch number 12 may be contained in a record which utilizes the key of 121016. The generic key to enable the first product record to be accessed for branch number 12 would then be the generic key 120000 for ISAM, or 212 for VSAM. The "2" indicates (in binary) that a generic key of length two bytes follows, for branch number 12 in this example.

When a full record key is used to access an ISAM data set, it must locate a record on that data set with the identical key; otherwise, an error indication is returned to the application program.

However, when a full record key is used to access a key-sequenced VSAM data set, any search for relevant VSAM records must be specified as:

- <u>Full Key Equal</u> - indicates that the key provided by the application program is a full key, and failure to locate a record with this exact key will result in an error indication being returned to the application program.
- <u>Full Key Greater or Equal</u> - specifies that the record key is a full key, and that the first data record with a key equal to or greater than the supplied record key is to be retrieved. This is equivalent to using a generic key in ISAM.

- Generic Key Equal - indicates that the record key is a generic key with a specified generic length. A record whose key is equal to the supplied generic key for the number of bytes indicated is then retrieved. If one cannot be found, a "no record found" condition is returned to the program.

- Generic Key Greater or Equal - indicates that a generic key is provided, and the first data record with a key equal to or greater than this generic key for the number of bytes indicated is to be retrieved.

An additional advantage in the utilization of record keys is in the addition of records. When new records are added to the data set, they are inserted in sequence in the data set based upon their record key value.

Record Location

To facilitate retrieval from DAM or VSAM data sets, records are identified by their locations in the data set. VSAM record identification is based on relative byte address (RBA) within the data set. In the case of DAM, the physical block (record) identification can be on the basis of:

- Actual disk address (MBBCCHHR)

- Relative track and record within the data set

- Relative block number (for CICS/CS/VS only)

If a physical key is recorded for the physical record, it may be appended to each of the record identifications detailed above. Figure 5-8 shows some representative record identification field formats.

DAM data sets with or without physical keys can be accessed. If a physical key is recorded on disk preceding the data record, the record identification can indicate the relative track within the data set, and the key which is physically recorded with the data record to be retrieved.

Both blocked and unblocked DAM data sets are supported by CICS/VS file control. In the case of blocked DAM data sets, additional information may be provided to identify the logical record within the physical block. This logical record identification immediately follows the physical block identification (as detailed above) in the record identification field provided by the application program. Logical records may be selected from a physical block based upon:

- Record number within block

- Record key within block (as illustrated in Figure 5-8)

where the location of the record key within each logical record is defined in the file control table.

CICS/VS file control uses this logical record number or key to deblock the relevant logical record from the physical block and present it to the application program.

For VSAM data sets, the record location utilized is a relative byte address (RBA). VSAM data sets use this relative byte address to identify the location within the entire data set of information (such as a logical record) to be retrieved.

PHYSICAL RECORD (BLOCK) LOCATION	PHYSICAL KEY (IF PRESENT)	DEBLOCKING ARGUMENT (IF BLOCKED)	COMMENTS
Relative Block No.	-	Record No.	CICS/OS/VS Only
Relative Block No.	Key	Record Key	
TTR	-	Record No.	Relative Track and Record (binary)
TTR	-	Record Key	
TTR	Key	Record Key	
TTTTTTRR	-	Record No.	Relative Track and Record (zoned decimal)
TTTTTTRR	-	Record Key	
TTTTTTRR	Key	Record Key	
MBBCCHHR	-	Record No.	Actual Disk Address
MBBCCHHR	-	Record Key	

Block Reference	Physical Key (if present)	Deblocking Argument

Format of Record Identification Field in Program

Figure 5-8. DAM Data Set Record Location

Initially, this appears to restrict the size of the data set. However, the relative address is maintained in a fullword, enabling a data set to be maintained, containing 2 raised to the power of 32 bytes. This is equivalent to a data set of approximately 43 billion bytes, extending over more than forty-three 3330 disk drives.

Records which are written to an entry-sequenced VSAM data set are never moved until the data set is reorganized. Any additions to the data set are made at the end of the data set, and the relative byte address by which that added record may be subsequently retrieved is returned to the application program.

Using the relative byte address for record identification of a VSAM data set provides the following advantages:

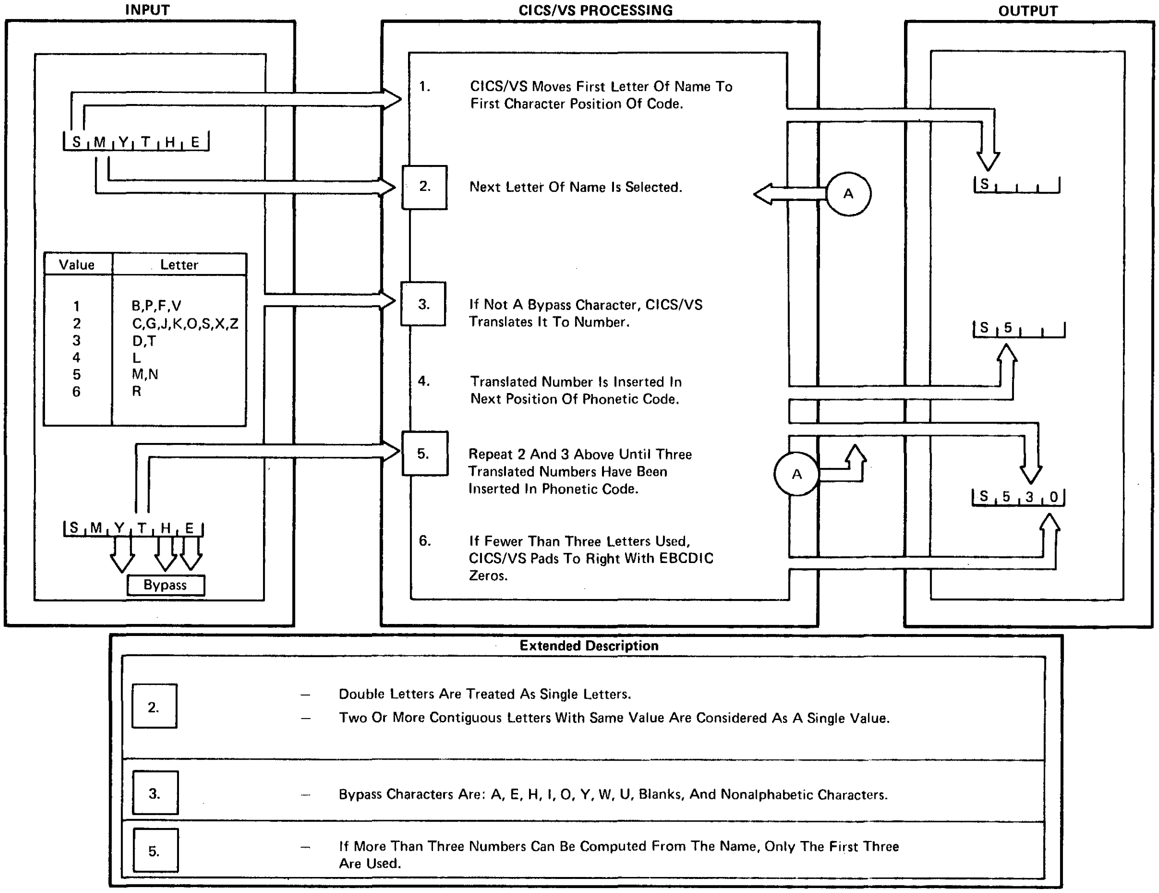
- Operates equally well with fixed-length or variable-length records
- Provides rapid file access, with full rotational position sensing support even when variable-length records are utilized

The record identification provided by an application program to access a VSAM data set by RBA is a four-byte relative byte address.

This may be calculated using techniques similar to that used to calculate a relative reccrd rumber or relative blcck number for DAM data sets, or the relative byte address may be stored as a pointer in logically related records.

Phonetic Conversion Functicn

Phonetic conversion is an cpticnal CICS/VS built-in function that can be utilized to develcp record keys based upon possible misunderstood or misspelled informaticn. The application program presents a 16-byte field to CICS/VS by issuing a built-in function (BIF) phonetic conversicn macro instructicn. The phcnetic ccnversion routine returns a four-byte phonetic equivalent cf the supplied field. This returned value consists of the first letter of the field, and three EBCDIC numbers which represent the letters in the rest cf the field.



Value	Letter
1	B,P,F,V
2	C,G,J,K,O,S,X,Z
3	D,T
4	L
5	M,N
6	R

1. CICS/VS Moves First Letter Of Name To First Character Position Of Code.

2. Next Letter Of Name Is Selected.

3. If Not A Bypass Character, CICS/VS Translates It To Number.

4. Translated Number Is Inserted In Next Position Of Phonetic Code.

5. Repeat 2 And 3 Above Until Three Translated Numbers Have Been Inserted In Phonetic Code.

6. If Fewer Than Three Letters Used, CICS/VS Pads To Right With EBCDIC Zeros.

Extended Description

2. – Double Letters Are Treated As Single Letters.
– Two Or More Contiguous Letters With Same Value Are Considered As A Single Value.

3. – Bypass Characters Are: A, E, H, I, O, Y, W, U, Blanks, And Nonalphabetic Characters.

5. – If More Than Three Numbers Can Be Computed From The Name, Only The First Three Are Used.

Figure 5-9. CICS/VS Phonetic Ccnversion

The key produced is based upon the phonetic sound of the name. Names which sound similar, but are spelled differently, will generally produce the same phonetic value (see Figure 5-9). For example, the names SMITH, SMYTH, SMYTHE, and SMITHS prcduce a phonetic key of S530. Likewise, the names ANDERSCN, ANDRESEN, and ANCRESENN produce a phonetic key of A336. This phonetic key is used as a partial key, which can then be used to access a name data base. Phonetically similar names will produce the same value, reducing errors caused by pronunciation or misunderstanding in spoken conversation. Misspelled names can be used in retrieving required data.

The built-in phonetic conversion function is based upon the phonetic conversion capability provided in the IBM Program Product FASTER (Filing And Selection Technique for Easy Retrieval: 5734-G21(OS), 5736-G24(DOS)). It is particularly useful in industries which require data sets to be accessed based upon names or product descriptions. It is also useful in a police information system, to identify all criminals and suspects with phonetically similar names. Phonetic conversion, together with the built-in weighted retrieval function, enables records to be retrieved based upon names, and records with phonetically similar names to be further identified based upon selection criteria through the use of the weighted retrieval function (see above).

For example, all criminals named Smith with specific personal characteristics can be identified. Criminals with a particular name and other identifying information, such as birth date and address, may be used to select the appropriate records.

A phonetic conversion subroutine is also provided by CICS/VS for use by batch programs which process online CICS/VS data sets in a batch environment.

INDIRECT ACCESS

CICS/VS file control enables data bases to be constructed. This is achieved by the use of the indirect access feature of file control, enabling various data sets to be constructed to identify logically related records in other data sets. The indirect access feature utilizes pointers from a record in the data set to logically related records in other data sets. The pointers can contain the actual disk address of a logically related record, the relative location of that record in its data set, or the key of that record. This enables identification and retrieval of information logically related to the record being processed.

Indirect Access Application Examples

Figure 5-10 illustrates a product record in a product data set and the supplier of that product through a supplier number. This supplier number is used as a pointer to access a separate supplier data set to obtain further information about the supplier of the product in question. The supplier number in the product record becomes a pointer to the supplier data set and can be indirectly accessed from the product data set.

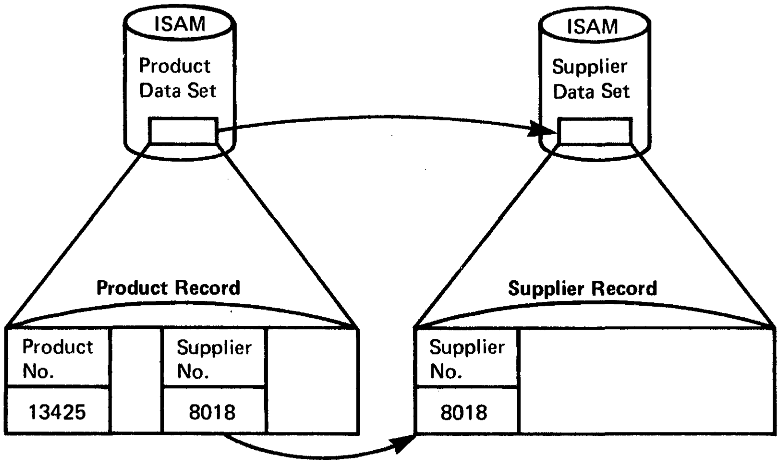


Figure 5-10. Product Data Set Indirect Access

Another example of indirect access is in an insurance policy information system. In this case, a policy record in a policy data set contains infcrmation relating tc the policyhclder (for example, customer number or name). If the customer data set is organized in customer name sequence as an ISAM data set, the name may be used as a key to retrieve the customer reccrd relating to that particular policy. In this way, the customer name in the policy record is used as a pointer for indirect access to further customer details in the customer data set (see Figure 5-11).

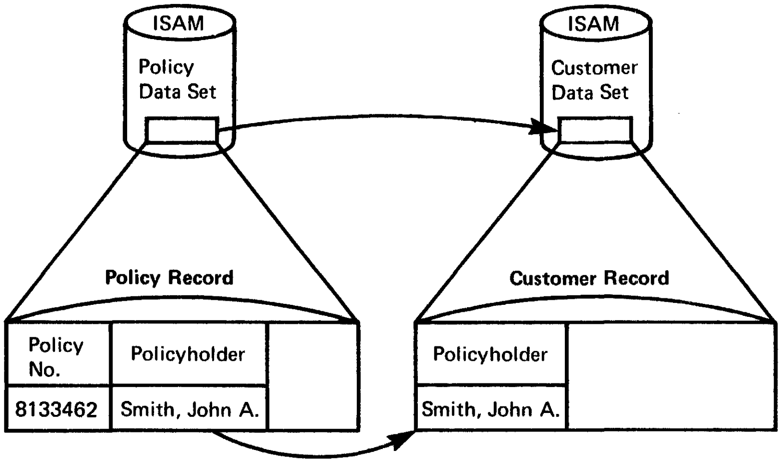


Figure 5-11. Policy Data Set Indirect Access

An indirectly accessed data set may also contain pointers to other logically related records in cther data sets. The customer record, indirectly accessed from the policy record, may in turn have a field which identifies that custcmer's insurance agent. The identification of this agent (agent number) may be used to access the related agent record in an agent data set (see Figure 5-12).

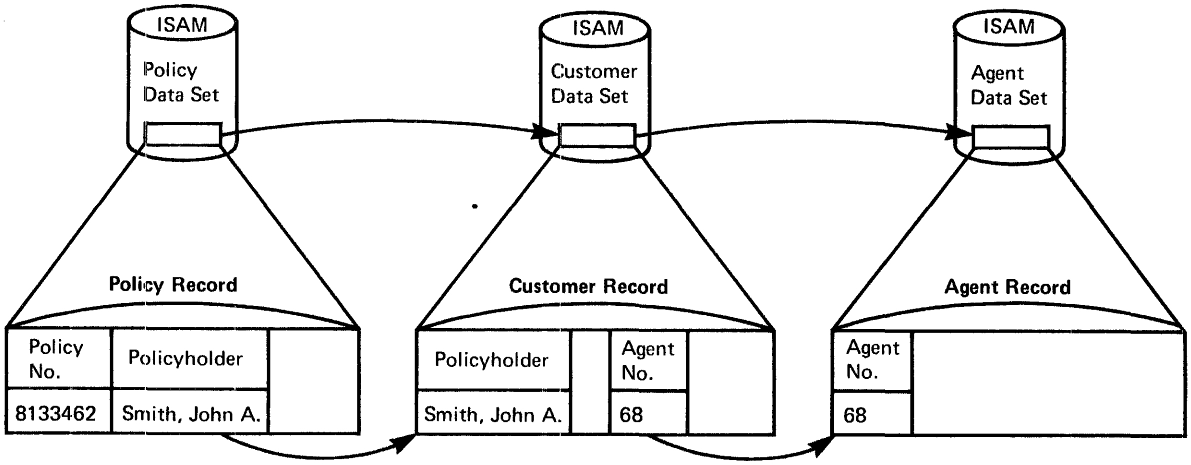


Figure 5-12. Indirect Access to Insurance Agent Data Set

Indirect Access Implementaticn

The CICS/VS file ccntrol indirect access feature enables fields in a record to be utilized as pcinters to lcgically related records in other data sets. There is nc limit to the number of indirect accesses to other data sets which may be made through the use of these pointers. Data sets may te indirectly accessed, regardless of whether they are fixed-length or variable-length ISAM data sets, DAM data sets, or VSAM data sets. Depending upcn the type cf data set, the pointer will be either a record location (in the case of DAM or VSAM data sets), or a record key (in the case of ISAM or key-sequenced VSAM data sets).

The data set which contains a pcinter field tc a logically related record in another data set is referred to as the index data set. The logically related data set is referred to as the cbject data set. An index data set may utilize several fields in a record to point to logically related records in several object data sets. These indirectly accessed object data sets may in turn utilize a field in their records to point to logically related records in cther data sets. These original object data sets tecome index data sets fcr the next level of indirectly accessed data set.

Indirect Access Initiaticn

Indirect access enables a chain tc be constructed through logically related records in many different data sets. Figure 5-13 illustrates a parts data set, which may te organized in part name sequence as an ISAM data set. This data set is accessed by means of a part name. The part name record is utilized as an index to a part number record in the parts data set. This part number record may in turn contain a supplier number utilized as a pointer to the supplier data set. In turn, the supplier record may contain a disk address pointer to an associated accounts payable record for that supplier in a DAM accounts payable data set.

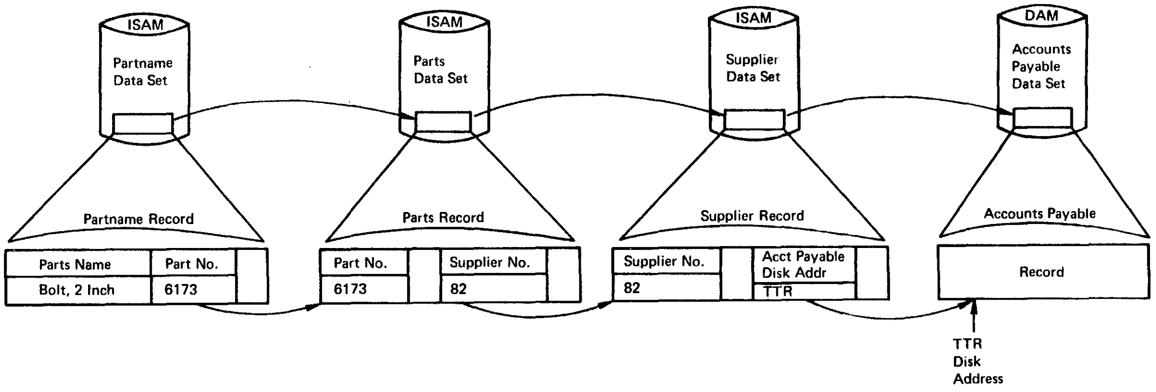
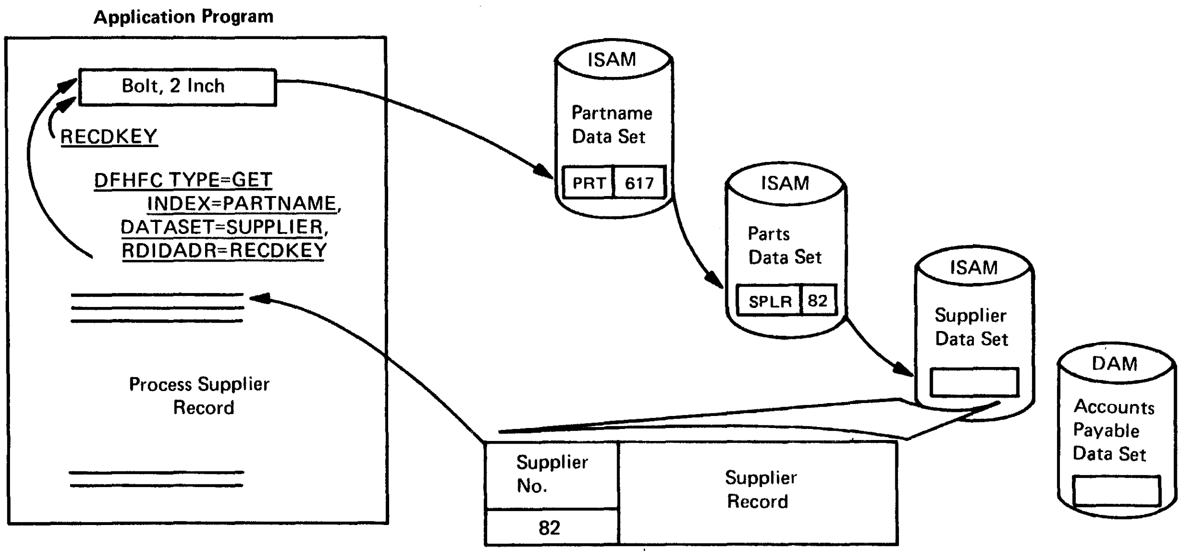


Figure 5-13. Indirect Access Chain in a Parts Data Base

To initiate an indirect access retrieval, the application program issues a file control GET macro instruction indicating the name of the data set to be utilized as an index data set, and the name of the data set from which a logically related record is to be retrieved. In the example illustrated in Figure 5-13, the application program may provide a part name key, indicate the part name data set as the index data set, and specify that a record is to be retrieved from the supplier data set, to obtain further information about the supplier of that part name. This is shown in Figure 5-14.



```
DFHFC TYPE=GET
      INDEX=PARTNAME,
    DATASET=SUPPLIER,
    RDIDADR=RECDKEY
```

Figure 5-14. Indirect Access Operation

CICS/VS file control automatically retrieves the part name record for the part name key provided by the program, extracts the part number, and uses it as a key to retrieve the part number record from the parts data set. Also, the supplier number is extracted from that parts record and is used as a key by file control to retrieve the related supplier record from the supplier data set. This supplier record is the record requested by the application program and is returned to it for processing.

In one GET request, CICS/VS file control follows the necessary indirect access chain, accessing as many data sets as required, to retrieve the record requested and present it to the application program. However, the application program is not aware of the number of data sets indirectly accessed. It appears to the program as if the supplier data set in the above example is in fact organized in part name sequence, rather than in supplier number sequence.

By following an indirect access chain in this way, file control must be aware of the logical relationship between data sets. This is achieved at system generation when the file control table (FCT) is generated.

Specification of Indirect Access Logical Relationships

As characteristics of each data set are specified in the FCT entry for that data set during FCT generation, an indirect access relationship between that data set and another data set may be specified. Information required to define an indirect access relationship is:

- Location of the field in the record to be used as a pointer to the indirectly accessed data set

- Length of that field or pointer

- Name of the object data set

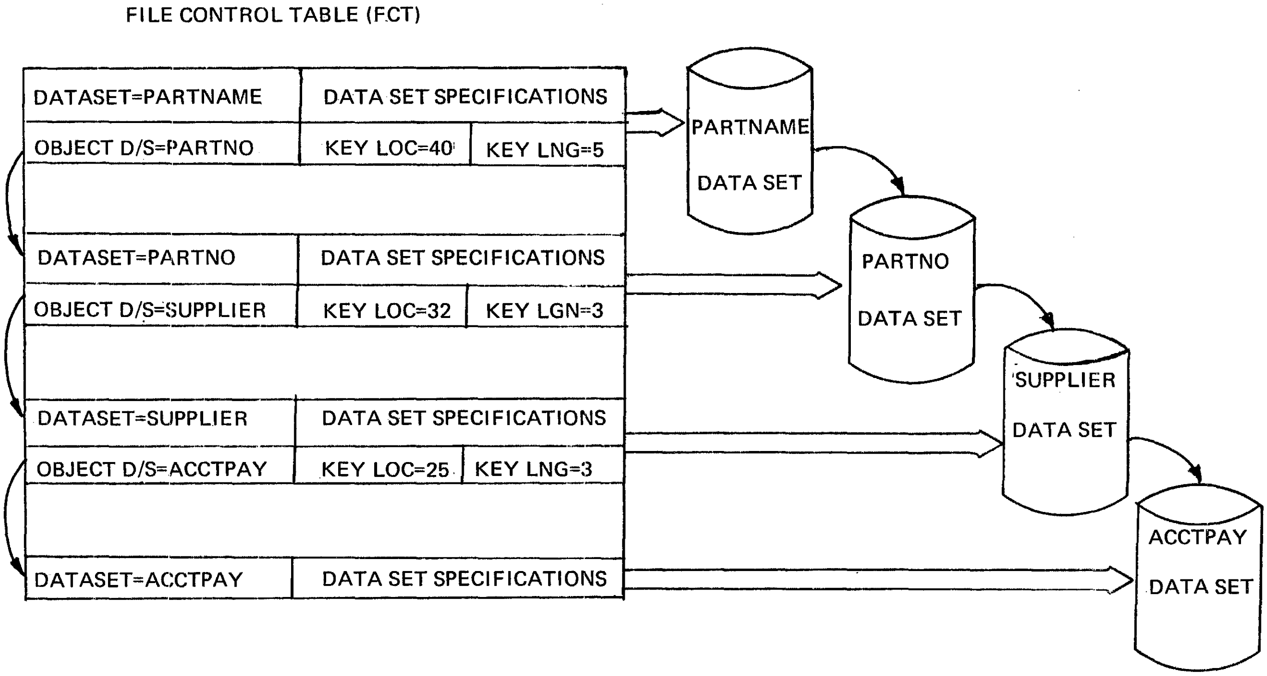


Figure 5-15. Specification of Indirect Access Logical Relationships

All fields in the reccrd which are to be utilized as pointers to indirectly accessed data sets are identified, together with the data sets to which they refer, as shown in Figure 5-15. This information defines the data set in question as an index data set and identifies the indirectly accessed data set as an object data set. These data sets, when they are subsequertly defined in the FCT, are also identified as index data sets which refer tc cther object data sets. This is done by defining the record fields which are to be used as pointers in those data sets, and the names cf the cbject data sets tc which they refer.

A chain of logically related data sets is defined in the FCT during system generation. When an application program requests indirect access retrieval, file control identifies a chain on which the object data is located. It then retrieves each related reccrd in the data sets on the identified chain in the sequence specified by the FCT, until the related record in the cbject data set is retrieved. It is then presented to the application prcgram for processing.

Indirect access retrieval enatles data bases to be constructed utilizing logically related information in a numter of data sets. One file control GET macro instruction causes all the required indirect accesses to be carried out until the requested record is retrieved for presentation to the task. This indirect accessing is carried out asynchronously, enabling cther ccncurrently executing tasks to continue processing.

Updating Indirectly Accessed Reccrds

Indirect access retrieval may be carried out with the intention of subsequently updating the cbject data set record, if required. This is indicated by the application prcgram specifying that this indirect access retrieval is also part of an update operaticn. When the object record is retrieved, exclusive ccntrcl is placed on that logical record for ISAM data sets, physical record for DAM data sets, or control interval for VSAM data sets. The application program issues either a PUT macro instruction to write the updated record tack, cr a RELEASE macrc instructicn to indicate that the record is nct to te updated, but that exclusive control is to te released.

Duplicates Data Set

In following a direct access chain through several data sets, the pointer field in an index data set reccrd can identify a number of separate records in its relevant object data set. As shown in Figure 5-12, a policy record identifies the pclicyholder by name. The customer data set in this case may be organized in custcmer name sequence, with the name used as a key tc access relevant records. However, there may be several customers with the same name, such as Jones. To develop a unique key for each policyholder with the name of Jones, additional informaticn covering first and seccnd names, birth date, or address must be added to the key. Hcwever, adding extra information to the record key reduces the amcunt of disk storage available for an ISAM data set, and wastes disk stcrage when additional identifying information is not used.

To overcome this problem, CICS/VS file control provides an additicnal capability with the indirect access feature, to enable duplicate records to be identified. This is achieved by utilizing the first byte of a pointer field in an index data set record. This first byte contains a unique code which cannct otherwise occur as part of the key. In the case of customer Jones, the first byte of the custcmer name field in the policy record can contain a unique code, for example, hexadecimal FF, as shown in Figure 5-16. This is immediately followed by the customer name (Jones in this case). The hexadecimal code FF identifies this as a pointer to several records with the same key. The key is utilized to access a separate duplicates data set, rather than the normal object data set. In this example, the key Jones is used to access a "duplicate customer" data set that contains one record with the name key of Jones. This duplicate record may in turn contain information enabling further identification of customers with the name Jones.

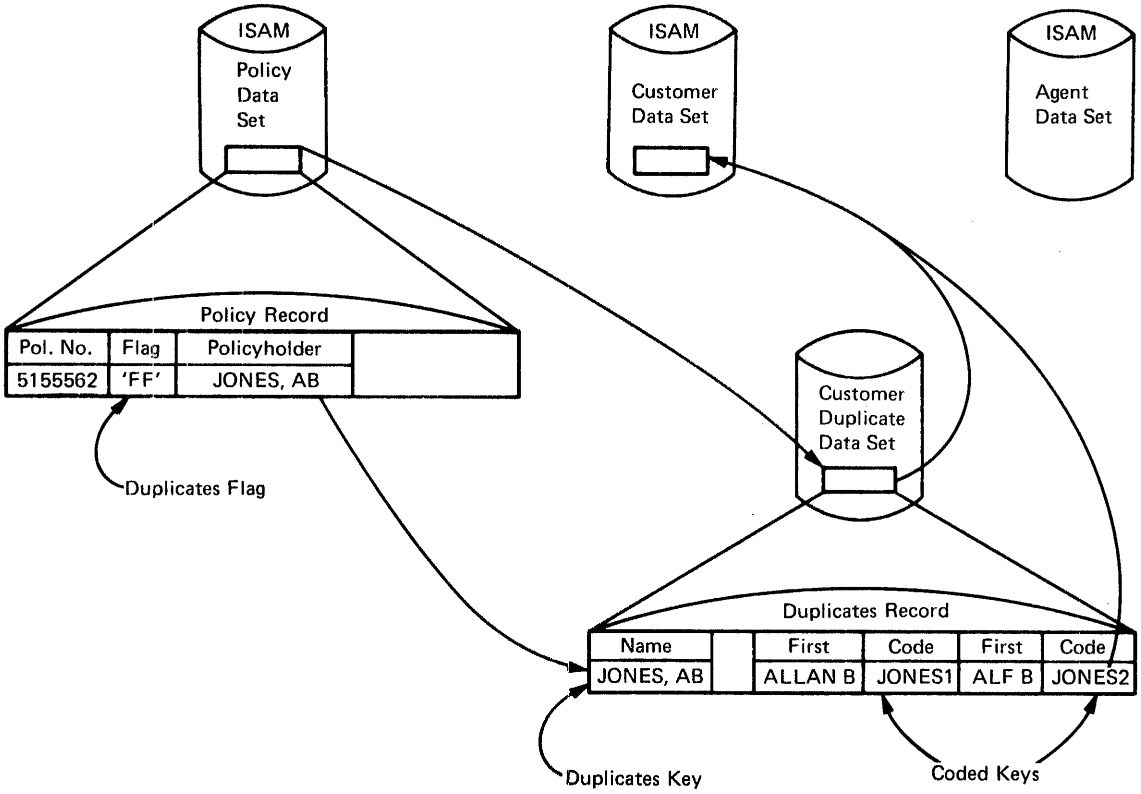


Figure 5-16. Duplicates Data Set for Indirect Access

Duplicates Data Set Implementation

The definition of a duplicates data set associated with a particular index data set is specified in the FCT as part of the index data set FCT entry. This indirect access FCT entry identifies the location and length of the pointer field in the index record and the name of the object data set. In addition, the user-specified duplicates code in the first byte of the pointer is defined, together with the name of the duplicates data set.

Even though an indirect access chain is followed through several data sets, the first byte of the pointer field to be used in a record is examined by file control to determine whether it is a duplicates code defined for that index data set. If it is, the duplicates data set is accessed using that pointer, instead of the defined object data set for that index record. The indirect access chain is broken at that point, and the duplicates record is returned to a user-supplied duplicates routine in the application program for further processing, instead of returning the requested object data set record. The function of the user-supplied duplicates routine is to examine the information presented in that duplicates reccrd, and uniquely identify the pointer to be used to retrieve the next record in the indirect access chain. This identification can be made either by the application program itself, cr by a reequest fcr further informaticn from the terminal operator if necessary (see Figure 5-16).

The duplicates data set feature becomes a powerful capability, enabling unique record identificaticn problems to be resolved so that an indirect access chain can be maintained through the various logically related data sets.

Additions to Indirect Access Data Sets

Records can be retrieved using the indirect access data set, either for processing only (GET), or for subsequent update (GET with UPDATE). The updated object record can be written back with a PUT macro instruction, or ignored by issuing a RELEASE macro instruction.

However, when adding new records to data sets which are indirectly accessed, care must be exercised tc ensure that the indirect access chains are correctly maintained.

Records can be added to data sets which are part of an indirect chain, using the procedures described in the topic "Direct Access." When constructing new records, the indirect access pointer fields must be correctly positioned, and must contain the correct information for record identificaticn tc access the object data set referenced by that particular index pointer.

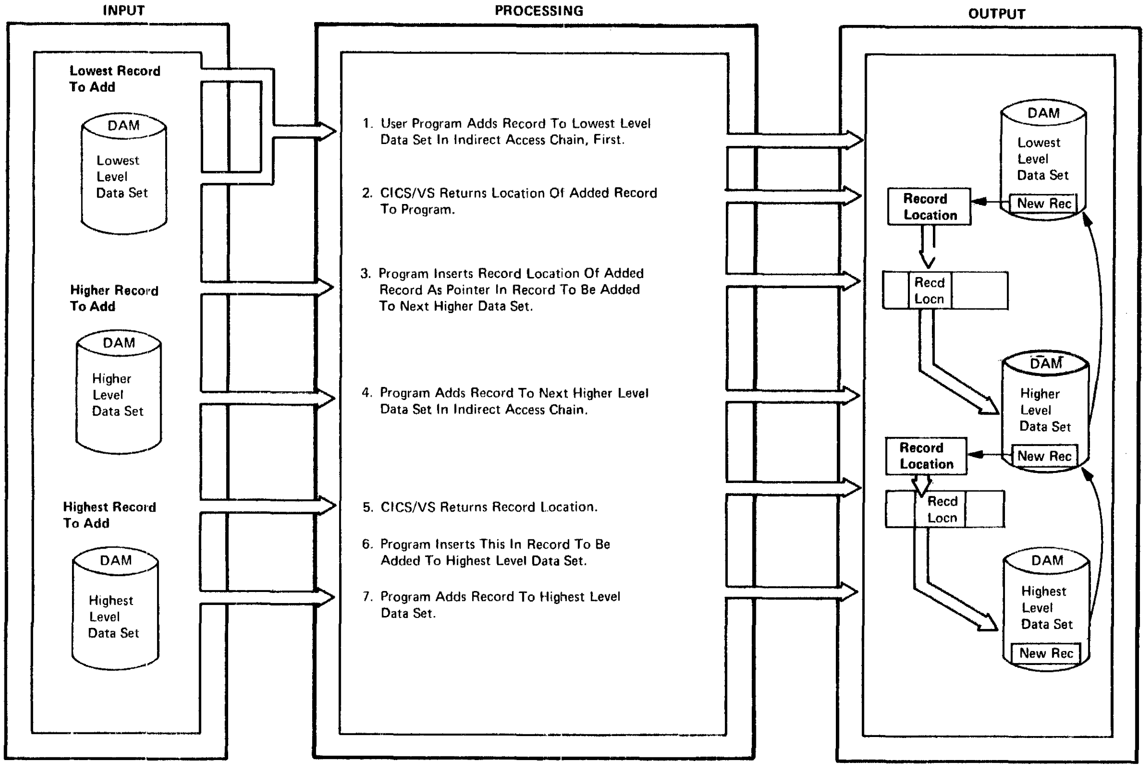
The order in which new records are added to indirect data sets is significant, depending upcn whether a record key or a record location pointer is utilized. If all the indirect access pcinters are record key pointers (for example, tc be used for accessing ISAM or key-sequenced VSAM data sets), the order in which additions should be made to the indirect data sets is not particularly significant. However, if some or all of the pcinters are record location fields for use with DAM or VSAM data sets, the crder becomes significant, because the actual locations of the added records are not known until the addition is completed.

As discussed above, and illustrated in Figures 5-4 and 5-5, records added to DAM data sets are written at a specified locaticn or near a specified locaticn depending upon the type of data set involved. In the case of DAM data sets with fixed-length records, an addition is made utilizing the first available dummy reccrd after the location specified by the applicaticn program where the addition shculd be made. In the case of DAM variable-length data sets, the addition is made at the end of the specified track, for CICS/DOS/VS, provided there is sufficient room to insert that new record. For CICS/OS/VS, a specified number of tracks can be searched tc locate a track which can contain the record.

On adding a new record to a DAM data set, the actual record location cccupied by the new record is returned to the application program. This record location can be used as a pointer to that new record from an indirect access index reccrd. By inserting this record location pointer in the appropriate field of the record used to index the added record, the indirect chain between these two data sets is maintained. Similarly, if the index record containing this record location pointer is a new record to be added, the lccation of that index record is returned to the application program and can be utilized as a pointer to the higher level index record for that data set.

In this way, new records may be added to a series of data sets, constructing any indirectly accessed chains in the process. This is achieved by adding records to the lowest level data sets first, and then progressing upward through higher level index data sets until the highest level data set record is added. The record location occupied by each new record is returned to the application program for use in constructing the next higher level index record.

Figure 5-17 illustrates the addition of records to indirectly accessed DAM data sets.



1. User Program Adds Record To Lowest Level Data Set In Indirect Access Chain, First.
2. CICS/VS Returns Location Of Added Record To Program.
3. Program Inserts Record Location Of Added Record As Pointer In Record To Be Added To Next Higher Data Set.
4. Program Adds Record To Next Higher Level Data Set In Indirect Access Chain.
5. CICS/VS Returns Record Location.
6. Program Inserts This In Record To Be Added To Highest Level Data Set.
7. Program Adds Record To Highest Level Data Set.

Figure 5-17. Addition of Records to Indirect Accessed Data Base

Additions to an entry-sequenced VSAM data set are made at the end of the data set, as discussed under the topic "Direct Access." The record location (RBA) of the added record is returned to the application program. This RBA may be utilized as a pointer from an index record to that new added record. Indirect access chains for entry-sequenced VSAM data sets may be built up (see Figure 5-17) in similar fashion to the method used for DAM data sets. Records are added at the lowest indirect access level first, and the RBA pointers returned to the application program are utilized as indirect access pointers in the next higher indirect access level. This progresses upward through successively higher indirect access levels until the highest indirect access level record has been added to its associated data set.

An indirect access chain can utilize any combination of DAM, ISAM, entry-sequenced VSAM, or key-sequenced VSAM data sets. If an indirect access chain utilizes any direct access data sets (DAM or entry-sequenced VSAM), additions must proceed frcm the lowest level data set upward to the highest level data set. It is a good technique to always add records at the lowest level and then at subsequently higher levels regardless of whether DAM, ISAM, or VSAM data sets are used.

Indirect Access Chain Integrity

Special consideration should be given to the possibility of system failure during the addition of an indirect access chain. If a system failure cccurs before all the logically related records are added to their relevant data sets, an incomplete direct access chain will result. Accordingly, techniques discussed in Chapter 8 should be utilized for journaling any indirect access additions, to enable incomplete indirect access chains tc be backed out on restart if necessary.

Normally, new indirect access chain records should be added offline to the data sets. However, adding these indirect access record chains online using the preceding technique is required to reduce the vulnerability of those data sets tc system failure, and it is advisable to ensure that only one direct access chain be added at a time. This may be achieved by all applicaticn programs that generate new indirect access chains enqueuing on the same single user resource prior to commencing an indirect access chain addition. On successful completion of the entire indirect access chain addition, the application program can dequeue itself from the single user resource. By utilizing CICS/VS task control ENQ/DEQ macro instructions, only one task at a time will be able to carry out an indirect access chain addition. While this may have an effect cn online perfcrmance, depending upon the frequency of adding new chains, it will result in a definite improvement in the integrity and safety of the varicus data sets in the event of a system failure.

SEGMENTED RECORDS

The CICS/VS file ccntrol segmented record feature enables data sets to be constructed for efficient use cn disk and dynamic storage space, including those data sets containing a considerable amount of variable-length information and which have various fields that are either present or absent in specific records.

The segmented record feature ccnsiders a record to be comprised of a number of segments - each segment containing one or more related fields. For example, in a customer data set (see Figure 5-18), the customer name may be defined as one segment and each address line as another segment. A number of customer account history fields containing the balance outstanding (current, cne month, two mcnths, three months, and over three months) may comprise ancther single segment.

Customer Number	Credit Limit	Customer Name	Postal Address				Ship-To-Address				Customer History	Account History — Arrears				
			Line 1	Line 2	Line 3	Line 4	Line 1	Line 2	Line 3	Line 4		Current Balance	One Month	Two Months	Three Months	Over Three Months
6 bytes	6 bytes	20 bytes	20 bytes	20 bytes	20 bytes	20 bytes	20 bytes	20 bytes	20 bytes	20 bytes	60 bytes	6 bytes	6 bytes	6 bytes	6 bytes	6 bytes

- Record Format: Fixed-Length
- Record Length: 282 Bytes

Figure 5-18. Typical Customer Record Format

Another example of segmented records is shown in Figure 5-19, which illustrates a savings acccunt data set used in the banking industry. The account master informaticn may te defined as cne segment. Each deposit or withdrawal transaction made previously against the account may also be defined as a segment.

Account Master Information				Previous Transactions				Against This Account			
Account Number	Account Type	Current Balance	Other Account Details	Passbook Withdrawal	Passbook Deposit	No Book Deposit	Passb With	Passbook Withdrawal	No Book Withdrawal	Passbook Deposit	Unused

Figure 5-19. Typical Savings Acccunt Record Format

Segment Design

Generally, fields are grouped within one segment if they have some similar relationship, such as similar information which will be operated upon together by various programs, or fields which are either all present or all absent for a particular record, or single fields which may contain variable-length information such as name or address lines.

A segment may contain any number of fields up tc a total segment length of 255 bytes. Segments may be further defined as either fixed-length segments, or variable-length segments whose length is indicated by a one-byte binary field at the start of the segment.

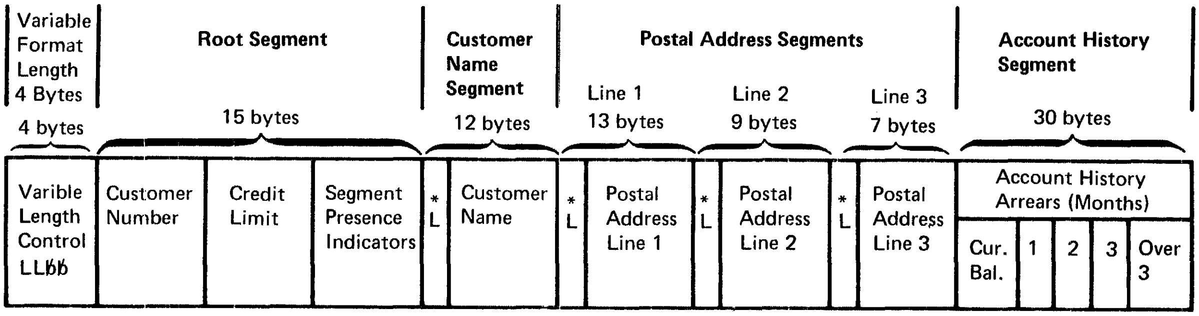
Consider the storage of a customer name and address in the record format shown in Figure 5-18. How many bytes should be allocated for the name field? Should each name field be allocated 20 bytes or 30 bytes or 15 bytes? Depending upon the characteristics of various customer names, a 30-byte name field could contain every possible name, whereas a 20-byte name field may contain 95% of customer names, and a 15-byte name field only 80% of names. Using the 30-byte name field will avoid the necessity of abbreviating or reducing the lengths of names, as would be necessary using 20-byte or 15-byte fields. However, the great majority of names may be less than 15 bytes. In this case, 15 bytes or more of disk space in each customer record is wasted if a 30-byte name field is allocated.

How many bytes should be allocated to each address line? Allocation of 30 bytes for each address line may enable every possible address line to be stored in full, while 20-byte or 15-byte address lines may require some form of abbreviation. Again, the use of 30-byte address line fields may result in wasting approximately 15 bytes or more per address line, because the majority of address lines may fit within 15 bytes.

Another consideration is the number of address lines to allocate for a customer record. Many customer addresses have two or three address lines, while some may require six or more address lines. Should six address lines be allocated for each customer record? Should instead three address lines be allocated, reducing longer addresses to three lines, but wasting an address line field in the case of a two-line address?

The segmented record feature enables record format definition problems such as these to be easily resolved. The customer name and each customer address line may be defined as separate segments. Furthermore, they may be defined as variable-length segments such that for one additional length byte at the start of each segment, the exact length of that segment can be indicated. Thus, a five-character customer name would occupy five bytes plus one byte for the length indication - a total of six bytes. However, a 25-byte customer name would occupy a total of 25 plus one or 26 bytes, while a 14-character name would occupy a total of 15 bytes. Only the amount of storage required for each individual name need be allocated, for more efficient disk storage utilization.

Figure 5-20 shows that each address line segment may be defined as variable-length, with the actual length of each address line occupying only that many bytes, together with an additional byte per address line as a length indication. Names and addresses need not be abbreviated, but may occupy as little or as much disk storage as required.



* L = 1-Byte Length Field For
Variable-Length Segments

**LLbb = 4-Byte Variable-Length
Record Control

- Record Format: Variable-Length
- Length Of This
Customer Record: 90 Bytes
- Original Record Length
If Not Segmented: 282 Bytes

Figure 5-20. Segmented Custcmer Record Format

Presence or Absence of Segments

A further advantage of variable-length segments is that segments may te defined as being either present or absent on an individual record basis. In Figure 5-18, eight address line segments are defined for each customer record. However, if only three address lines are required, only three address line segments need be present for that record (see Figure 5-20). The remaining five possible address line segments are not present, and do not occupy any disk space.

Root Segment

Fields which are always present in every record, such as a customer number, credit rating, and other required information for each record, are generally grcuped together into one segment. This is called the root segment and precedes all other segments in the record. This segment is always present in a segmented record.

Segment Indicator Flags

The presence cr absence of other segments in the record is indicated by segment indicator flags. These indicator flags are contained within the root segment, and may comprise either bit indicators or halfword indicators.

Bit indicators use a separate tit tc indicate the presence or absence of each specific segment. If the segment associated with a bit is present, that bit is turned cn. However if the segment associated with the tit is absent, that tit is turned off.

Halfword indicators utilize two bytes to indicate the presence or absence of each segment. The twc tytes are used as a zero/nonzero switch. If the twc bytes are nonzerc, the associated segment is present; if the two bytes are binary zero, the associated segment is absent.

COBOL programs must use halfwcrd indicators, unless the bit manipulation built-in functicn supplied by CICS/VS is utilized (see "Application Built-in Functicns" in Chapter 2). Figure 5-21 illustrates the use of segment indicator flags, and relates to the customer record shown in Figure 5-20.

Segment Definiticn in FCT

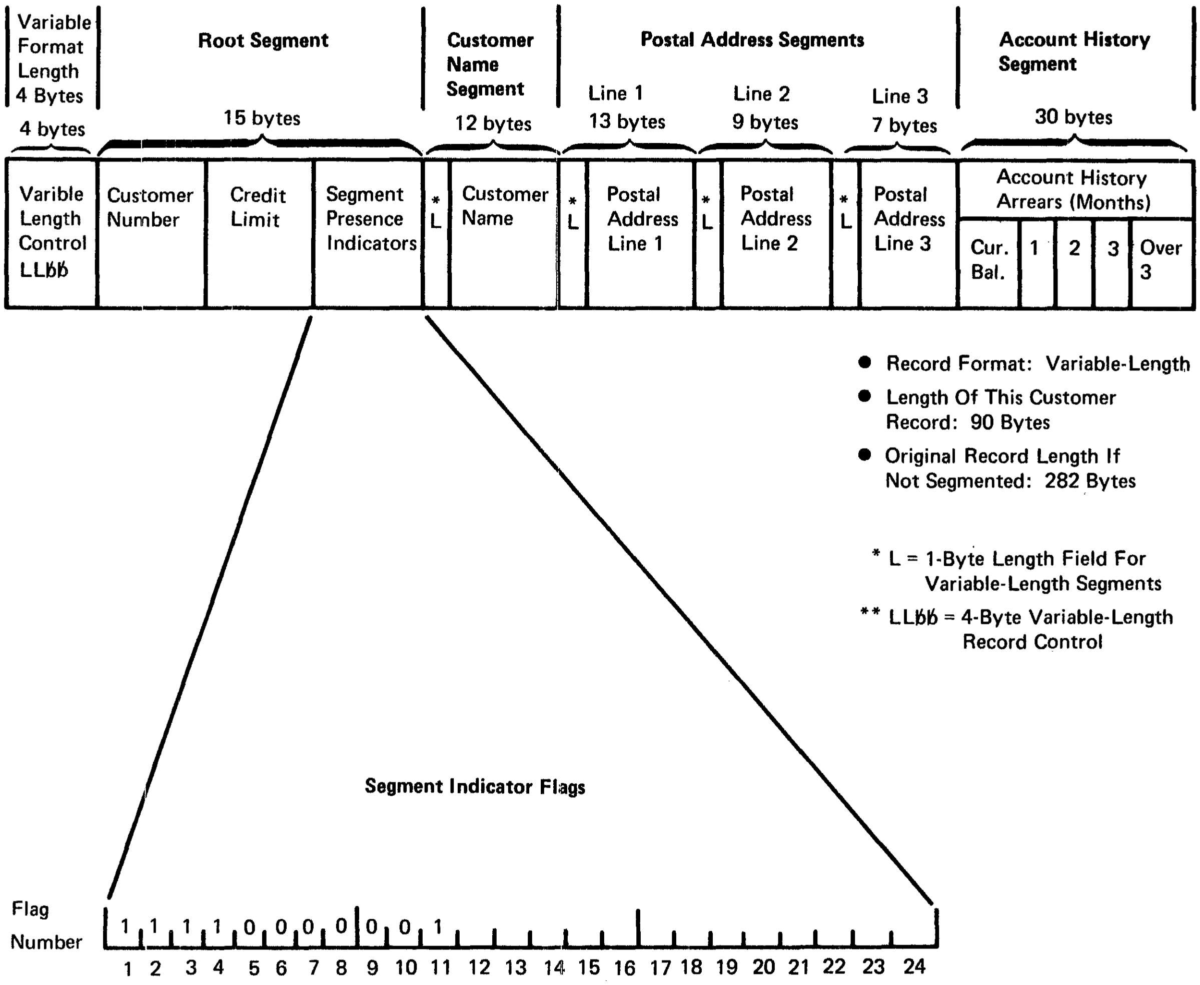
While it is the responsibility of the system programmer to specify segments in the FCT, it is important that the system designer have a general understanding of how segments are defined, to better understand the operation of the CICS/VS segmented record feature. This will enable him to take advantage of facilities cffered by this feature when initially designing the varicus data bases which will be utilized by the application.

Either bit indicators or halfword indicatcrs may be used with a particular segmented record data set, tut not both. The selected type of indicators is specified in the file ccntrcl table (FCT) entry for that segmented record data set. As many bytes as required to contain all of the necessary segment indicatcr flags (up tc a maximum of 99 segment flags) must te defined within the root segment. The starting location, and length in bytes, of the segment indicator flags field in the root segment is also defined in the FCT entry for a segmented record data set. See Figure 5-21.

Each separate segment is then defined in the FCT entry (see Figure 5-22). Every segment is allccated a segment name up to 8 characters long, and is specified as a fixed-length segment or a variable-length segment of a defined maximum length. In addition, any required boundary alignment (byte, halfword, fullwcrd, or doubleword) necessary when the segment is read into storage is identified. As each segment is presented to the applicaticn prcgram for processing, the segment is aligned on the specified bcundary. However, the extra bytes required to ensure specific boundary alignment are not recorded on disk; they are inserted when the record has teen read from disk and before the segment is passed to the applicaticn program.

An application program may utilize several segments in a record for processing. These segments may te grouped together and called a segment set. By identifying a segment set by name, all the segments comprising that segment set are also identified.

Consider a custcmer data set in the utilities industry, as shown in Figure 5-23. An applicaticn program which requires access only to the customer name and address may request a segment set comprised only of the customer name segment and each of the postal address line segments. This may be uniquely identified as a segment set which is given a specific name, such as NAMEADDR.



- Record Format: Variable-Length
- Length Of This Customer Record: 90 Bytes
- Original Record Length If Not Segmented: 282 Bytes

* L = 1-Byte Length Field For Variable-Length Segments

** LLb̸b̸ = 4-Byte Variable-Length Record Control

- Segment Indicator Flags Indicate The Presence Or Absence Of Segments.
- 1 Bit or 2 Bytes May Be Used To Indicate The Presence Or Absence Of Each Segment. Here, Bit Indicators Are Used.
- Flag 1 Refers To The First Segment After The Root, That Is, Customer Name. Flag 2 Refers To Postal Address Line 1, Flag 3 Is Postal Address Line 2, And So On, Up To Flag 11, Which Is Account History.
- A Bit ON Indicates The Relevant Segment Is Present; A Bit OFF Indicates The Segment Is Absent.
- The Example Here Shows That Customer Name, A 3-Line Postal Address, And Account History Are Present In This Particular Customer Record.
- Bits 12 Through 26, In This Example, Are Not Used. They Have Been Allocated To Enable Another 13 Segments To Be Added To This Record Later, For Other Applications, Without Necessitating Any Modification To Existing Application Programs.

Figure 5-21. Segment Indicator Flags

6 bytes	6 bytes	20 bytes										60 bytes	6 bytes				
Customer Number	Credit Limit	Customer Name	Postal Address					Ship-To-Address			Customer History	Account History — Arrears				
			Line 1	Line 2	Line 3	Line 4	Line 1	Line 2	Line 3	Line 4		Current Balance	One Month	Two Months	Three Months	Over Three Months

Data Set = Customer

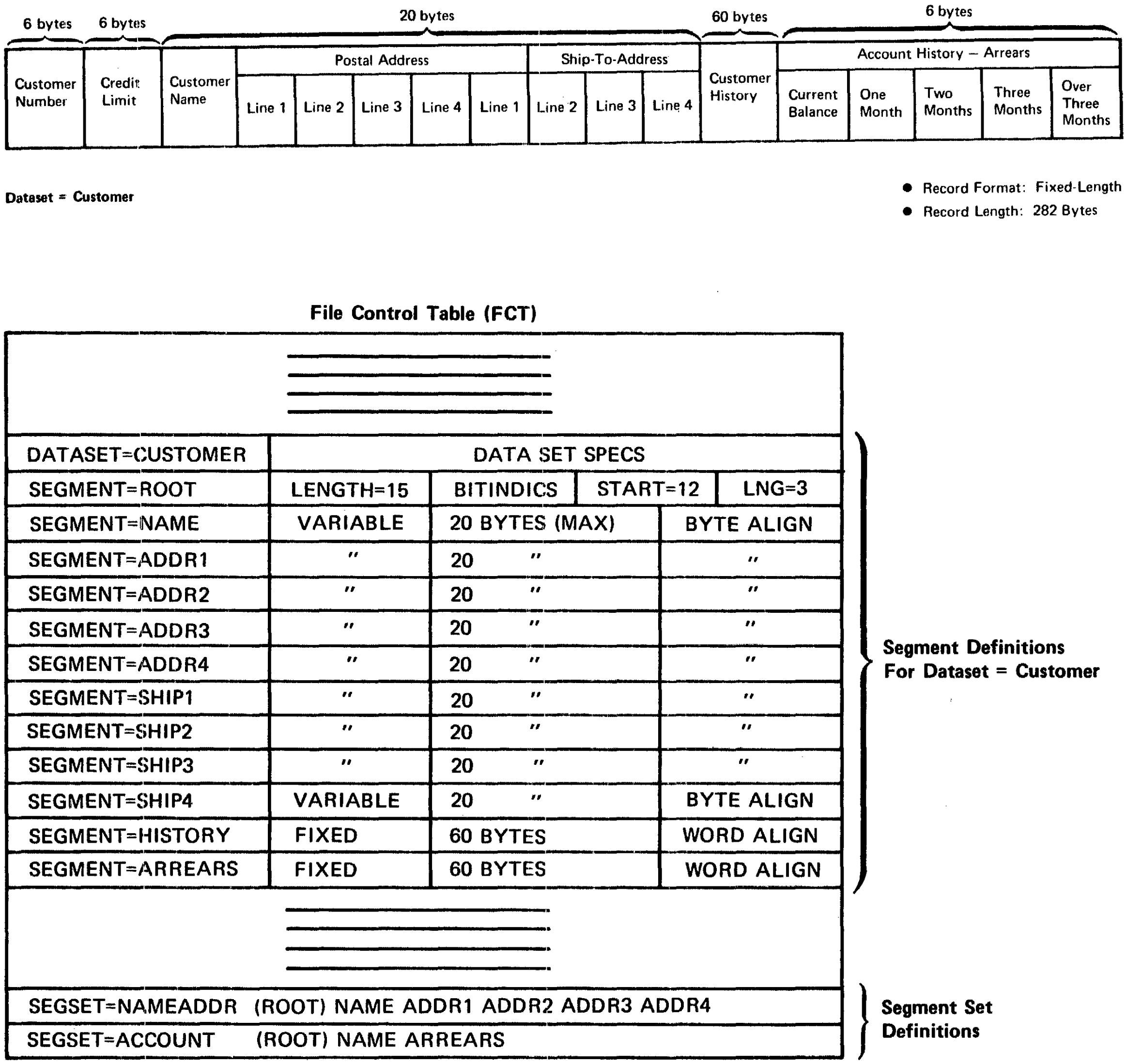
- ● Record Format: Fixed-Length
- ● Record Length: 282 Bytes

File Control Table (FCT)

DATASET=CUSTOMER	DATA SET SPECS			
SEGMENT=ROOT	LENGTH=15	BITINDICS	START=12	LNG=3
SEGMENT=NAME	VARIABLE	20 BYTES (MAX)	BYTE ALIGN	
SEGMENT=ADDR1	″	20 ″	″	
SEGMENT=ADDR2	″	20 ″	″	
SEGMENT=ADDR3	″	20 ″	″	
SEGMENT=ADDR4	″	20 ″	″	
SEGMENT=SHIP1	″	20 ″	″	
SEGMENT=SHIP2	″	20 ″	″	
SEGMENT=SHIP3	″	20 ″	″	
SEGMENT=SHIP4	VARIABLE	20 ″	BYTE ALIGN	
SEGMENT=HISTORY	FIXED	60 BYTES	WORD ALIGN	
SEGMENT=ARREARS	FIXED	60 BYTES	WORD ALIGN	

Segment Definitions For Dataset = Customer (See Record Format Above)

Figure 5-22. Segment Definition in FCT



6 bytes	6 bytes	20 bytes									60 bytes	6 bytes				
			Postal Address					Ship-To-Address				Account History – Arrears				
Customer Number	Credit Limit	Customer Name	Line 1	Line 2	Line 3	Line 4	Line 1	Line 2	Line 3	Line 4	Customer History	Current Balance	One Month	Two Months	Three Months	Over Three Months

Dataset = Customer

- Record Format: Fixed-Length
- Record Length: 282 Bytes

File Control Table (FCT)

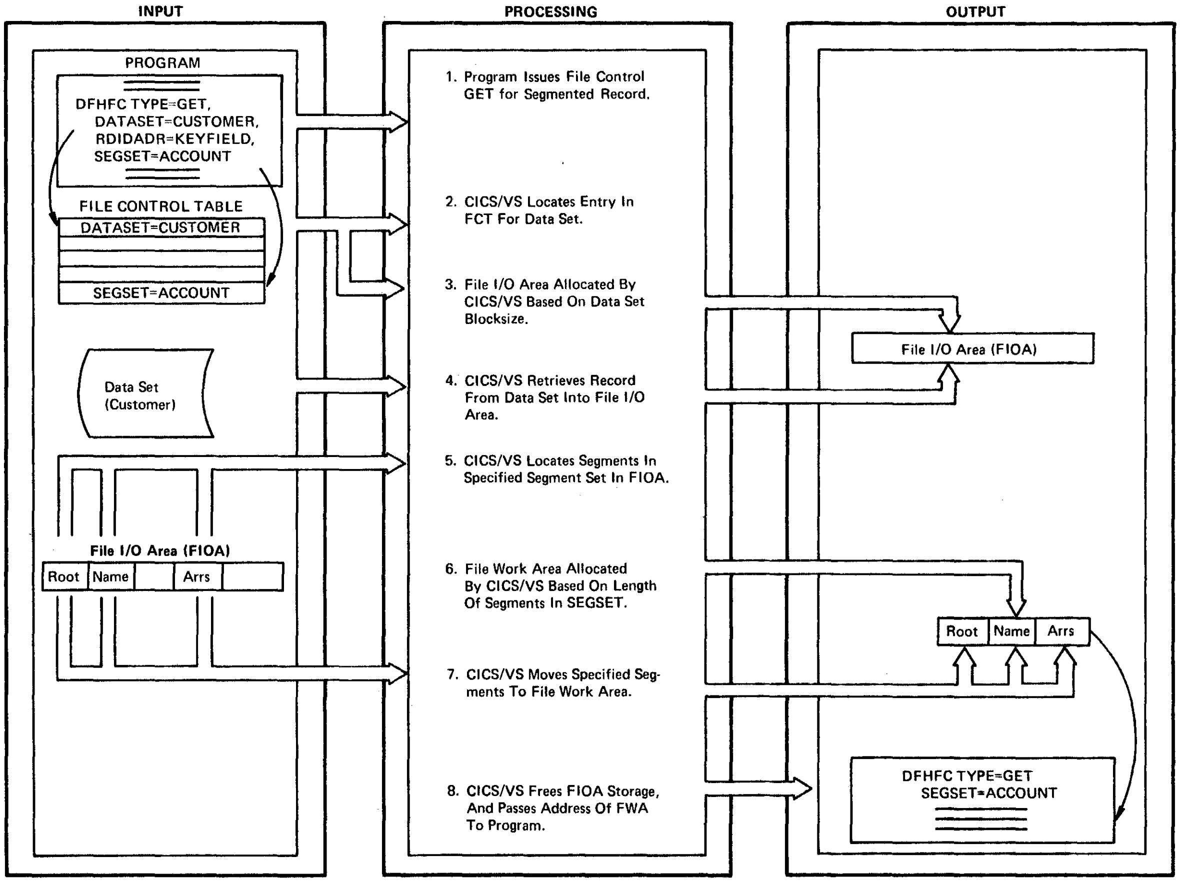
DATASET=CUSTOMER	DATA SET SPECS			
SEGMENT=ROOT	LENGTH=15	BITINDICS	START=12	LNG=3
SEGMENT=NAME	VARIABLE	20 BYTES (MAX)	BYTE ALIGN	
SEGMENT=ADDR1	"	20 "	"	
SEGMENT=ADDR2	"	20 "	"	
SEGMENT=ADDR3	"	20 "	"	
SEGMENT=ADDR4	"	20 "	"	
SEGMENT=SHIP1	"	20 "	"	
SEGMENT=SHIP2	"	20 "	"	
SEGMENT=SHIP3	"	20 "	"	
SEGMENT=SHIP4	VARIABLE	20 "	BYTE ALIGN	
SEGMENT=HISTORY	FIXED	60 BYTES	WORD ALIGN	
SEGMENT=ARREARS	FIXED	60 BYTES	WORD ALIGN	
SEGSET=NAMEADDR	(ROOT) NAME ADDR1 ADDR2 ADDR3 ADDR4			
SEGSET=ACCOUNT	(ROOT) NAME ARREARS			

Figure 5-23. Segment Set Definition in FCT

Another application program may require access cnly to the customer name and acccunt history segments in Figure 5-23. These segments may be defined in a separate segment set which may be given a unique name such as ACCOUNT. The application prcgram in this way indicates that it is sensitive cnly to segments in the specified segment set.

Segment Retrieval

When the application program wishes to retrieve a specified segment set from a segmented reccrd, it prcvides information for the identification of that reccrd, such as a record key or record location field, identifies the data set tc be accessed by name, and then identifies the segment set by name to be presented to the application program (see Figure 5-24).



1. Program Issues File Control GET for Segmented Record.
2. CICS/VS Locates Entry In FCT For Data Set.
3. File I/O Area Allocated By CICS/VS Based On Data Set Blocksize.
4. CICS/VS Retrieves Record From Data Set Into File I/O Area.
5. CICS/VS Locates Segments In Specified Segment Set In FIOA.
6. File Work Area Allocated By CICS/VS Based On Length Of Segments In SEGSET.
7. CICS/VS Moves Specified Segments To File Work Area.
8. CICS/VS Frees FIOA Storage, And Passes Address Of FWA To Program.

Figure 5-24. Segment Retrieval

File ccntrol uses the reccrd identification field to access the required record from the specified data set. The segment set name is then used to identify the particular segments making up that set. Each of the segment names in the segment set is defined in the FCT entry as being fixed-length or variable-length, of a specified maximum length, and requiring specified boundary alignment. Segments are assumed to be in the same sequence in the record as the sequence in which they are defined in the FCT.

The segments in the segment set are extracted from the record by file control. These segments are presented to the application program together with the root segment, as if only those segments were present in the record on disk (see Figure 5-24). Other segments which are not part of the segment set are ignored, and are not presented to the application program.

The presence or absence of segments in the record passed to the program is indicated by the status of the segment indicator flags (see Figure 5-21). These segment indicator flags are located within the root segment, and each flag is tested by the application program to determine whether it is on or off, and hence whether its associated segment is present or absent. The first segment indicator flag (which may be a bit flag, for example) indicates the presence or absence of the first segment in the record following the root segment. The second indicator flag specifies presence or absence of the second segment in the record following the root segment, and so on. These indicator flags specify the presence or absence of all segments within the record, not only those segments in the segment set passed to the program.

The use of DSECTs or structures in application programs can simplify the testing and manipulation of segment indicator flags. If the appropriate DSECT or structure defining all indicators is copied into the application program when it is compiled, the use of installation-controlled labels for each indicator may be achieved. The application programs may then be less sensitive to changes in segments, a segment change being made to the DSECT or structure and the programs then recompiled.

Segment Updating (Key-Sequenced VSAM)

Application programs may replace segments, delete segments, or add new segments. However, the extent to which new segments may be added depends upon the particular access method used for the data set. If DAM, ISAM, or entry-sequenced VSAM data sets are used for the segmented record data set, the total segmented record length may not be increased, but may be reduced if required. (An exception is with OS/VS variable-length ISAM records, where the segmented record length may be increased if necessary.) With key-sequenced VSAM data sets, the segmented record length may be increased, either through the addition of new segments to the record or by a change in the length of existing segments. The increased record length is regarded by file control as a replacement of the previous segmented record and is handled in much the same way as the addition of a new record to the data set. That is, records which follow the lengthened record in the same control interval are shifted to the right to enable the increased-length record to be inserted.

If it is necessary to increase the length of DAM, ISAM, or entry-sequenced VSAM segmented records, a different technique, segment updating, must be utilized.

Segment Updating (DAM, ISAM, and Entry-Sequenced VSAM)

When it is determined that a segmented record in a DAM, ISAM, or entry-sequenced VSAM data set will be increased in length because of the addition of a new segment, or the increase in length of an existing segment, the original segmented reccrd retrieved must be updated by the application program to indicate that that record is now obsolete. This may be specified by the program placing a logical delete flag in the root segment as shown in Figure 5-25.

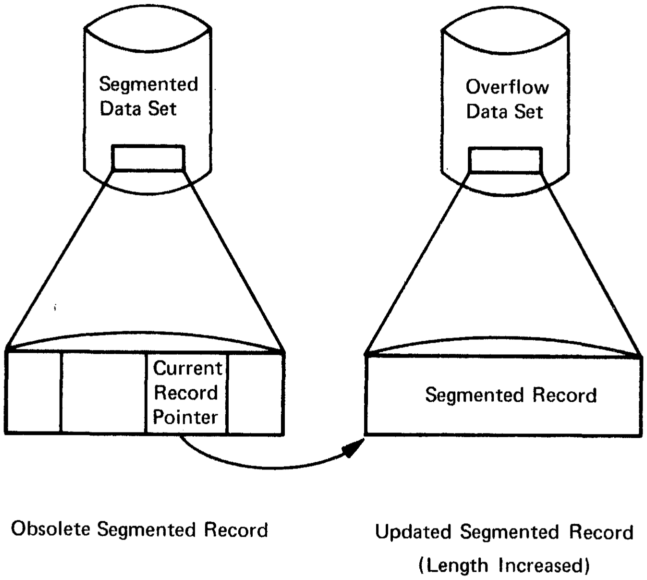


Figure 5-25. Segment Updating, with Length Increase in DAM, ISAM, and Entry-Sequenced VSAM Data Sets

The increased-length record may be written by the user program to a separate user-defined "overflow" data set. The location of that increased-length reccrd in the overflow data set may be inserted as a record pointer field in the root segment of the criginal record (see Figure 5-25). The logical delete flag in the root segment may be utilized as the record pcinter field. If this reccrd pointer field is zero, it indicates that the segmented record is current. However, if the record pointer field is ncnzero, it indicates that the segmented record has been logically deleted and replaced by ancther record in the overflow data set. The record pcinter may be utilized by the user program as a record key, or record location pointer (depending upon the particular data set it pcints to). It directs the application program to the new current segment in the overflcw data set.

With this technique, an increase in length of a segmented record may be logically supported using DAM, ISAM, or entry-sequenced VSAM data sets. The increased-length reccrd is first written by the user program to the overflow data set. The record location field returned by CICS/VS as a result of that addition, or the record key for ISAM, is then inserted into the record pcinter field in the root segment of the record in the segmented data set (See Figure 5-25).

On subsequent retrieval, the application prcgram first tests the record pcinter field in the rcot segment. If the pointer field is nonzero, the application program uses that pointer as a record identification tc the more current segmented record in the overflow data set, and retrieves that current segmented reccrd for prccessing. The overflow data set must be specified during FCT generation as having the same segments, and segment sets, as the criginal segmented data set.

If a segment is to be updated with no change in length, that update is effected and the segments are presented by the program to CICS/VS to be written back to the data set by means of a PUT macro instruction. If the length of a variable-length segment is to be reduced, the length byte is modified to reflect the new length of the updated segment. This segment is then written back to the data set by issuing a file control PUT macro instruction. File control blocks up each segment in the segmented record, removes any boundary alignment bytes, and uses the length byte in variable-length segments to allocate a sufficient number of bytes to contain contents of that segment.

To allow for a potential increase in length after a GET with UPDATE macro instruction of variable-length segments, or the addition of a new segment which was missing in the segmented record, CICS/VS file control will present the application program with a segmented record containing space set aside for each segment, whether present or absent, based upon either the fixed length of that segment, or the maximum variable length of that segment.

Segment Deletion

A segment may be deleted if the application program turns off the relevant segment indicator in the root segment. File control then physically deletes that segment when it writes the segmented record back to disk.

Fixed- and Variable-Length Segmented Records

CICS/VS segmented records may be either fixed-length or variable-length. For fixed-length records, the actual segmented record may be less than the allocated fixed-length record size. Slack bytes are therefore used at the end of the segmented record to make up the specified fixed-length record. Segmented records which are variable in length because of variable-length segments, or the absence of particular segments, may be specified as variable-length records (see Figure 5-26). In this case, variable-length records enable more efficient disk storage utilization.

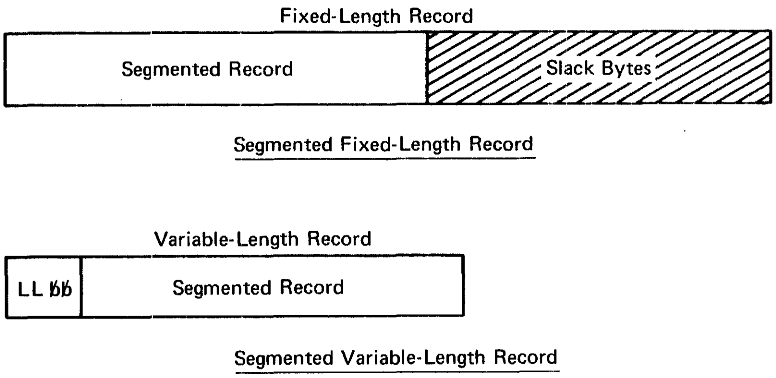


Figure 5-26. Segmented Record Disk Utilization

Creation and Maintenance of Segmented Records

CICS/VS file control provides no facilities fcr the creation or maintenance of segmented reccrds. These reccrds must be created offline, with the various segments (either fixed- or variable-length) blccked by user prcgrams. The offline user program must set the relevant indicator flags, to specify the presence cr absence of each segment in that segmented record. Once segmented records have been created offline, they may be retrieved online using the CICS/VS segmented record feature. In the event of addition of new segments or deletion of existing segments online, the online application program is responsible fcr determining that the appropriate bit or byte indicator flag is cn or cff, and turning on the appropriate indicator for a segment which has been added online, or turning off the indicator for a segment which is to be deleted online. CICS/VS will examine these indicator flags when the record is to be written back to disk, and take the appropriate acticn tc either delete the segment or increase the segmented record length if necessary to allow for an added segment.

With the availability of the built-in bit manipulation function (see Chapter 2), Assembler, PL/I, and American National Standard (ANS) COBOL application programs may test bit indicator flags and set bit indicator flags on or off. Consequently, the use of halfword indicator flags for ANS COBOL programs is not necessary, and all indicator flags may be maintained as bit flags.

ADVANTAGES OF SEGMENTED RECOFDS

The principal advantages in the use of the segmented record feature are described belcw.

Disk Storage Utilization

Through the use of variable-length records, variable-length segments, and omitting unused segments, only that amount of disk storage required for storage of the information relevant to each record need be allocated.

Dynamic Storage Efficiency

On an application prcgram GET macro instructicn, the entire segmented record is read into the allocated file input/output area (FIOA), or, for VSAM data sets, into the VSAM buffer. The segments making up the segment sets requested by the prcgram are extracted by CICS/VS from the record in the FICA, buffer, cr VSWA and are moved into the file work area (FWA). At this time, if the applicaticn program did not indicate that the record will subsequently be updated and written back, CICS/VS file control releases the FICA or VSWA dynamic storage, and transfers only the requested set of segments to the FWA. Consequently, the amount of main storage used in prccessing those segments is only as large as the segments themselves. No additioral dynamic storage is utilized for segments which are not being processed.

Limited Data Independence

Because an application prcgram identifies only that set of segments which are significant to it, the mcdification of segments in the data set which are not relevant tc the program concerned requires no modifications to that program. Limited data independence is thus achieved with a consequent reducticn in the amount of program modification following a change to the record format or data set organization.

A further consideration of data independence is that additional segment indicator flags may be allccated in the roct segment to allow for the addition of future segments. If bit indicators are used, each additional byte of bit flags will enable up to eight segments to be added to the end of the segmented record. When these segments are eventually added to the records, scme reorganization of the data set will of course be necessary. However, applicaticn programs which utilize segment sets which dc nct contain the new added segments are not affected.

CICS/VS FILE CONTROL DESIGN CONSIDERATIONS

As can be seen frcm the atcve discussion cf the indirect access feature and segmented record feature of CICS/VS file control, data bases may be constructed. A number of factors should be taken into account in deciding the apprcpriate data base support technique, either indirect access, or segmented records, or both.

Access To Online Data Base by Offline Programs

If data sets utilized online in a data base must also be processed offline, the indirect access feature may be found to be more suitable than segmented records. Use of this feature requires each offline application program to provide its own support for the extraction of segment sets from segmented records, and the insertion of segment sets back into segmented records for updating. Alternatively, the code supporting the segmented reccrd feature in the CICS/VS file control program may be extracted by the user and modified for use by batch programs.

If the batch processing requirements of the installation dictate that standard access methcd support be used for batch programs accessing online data sets, the indirect access feature rather than the segmented record feature should be used. This enables logically related information to be maintained acrcss a series of different data sets. However, this additional file accessing will have an effect on the perfcrmance of the online applications, and consequently on response time. This is a further consideration in the design of a CICS/VS file control data base.

Provided that suitable support can be developed to enable batch application programs to access segmented records in online data sets, the segmented record feature is superior to the indirect access feature when one considers the reduced file accessing necessary and the efficient utilization of disk storage. With one file access, a segmented record with all its related segments may be retrieved. With indirect access, these related segments may have been defined in separate indirectly accessed data sets, each requiring a separate file access. Furthermore, only the object or target data set is returned to the user.

Segmented Records

A consideration with segmented records is the extent to which additions to segments may be made. If segment additions result in an increase in the segmented record length, then the segmented record data set should ideally be defined as a key-sequenced VSAM data set, or a variable-length OS/VS ISAM data set. Alternatively, DAM, ISAM, or entry-sequenced VSAM data sets may be utilized using the dummy segment and overflow data set techniques described above. However, this will result in less efficient disk storage utilization, more user programming, and more file accesses.

A further consideration is the extent to which information may be added to data set records at a future time. If there is a strong possibility of this occurring, it may be advisable to use the segmented record feature, allocating additional segment indicator flags in the root segment to allow for a specified number of additional future segments. At that time, additional segments may be added, without requiring modification of programs which utilize segment sets not associated with the newly added segments.

Multiple Occurrence of Segments

The use of the segmented record feature enables a large number of multiple segments to be stored for a particular logical record within a block, to a maximum of 99 segments in each logical record. The number of multiple occurrences of segments may be further limited by user data set creation and maintenance factors.

Real Storage Availability

While the segmented record feature does not provide the capability of the DL/I products, and introduces some difficulties regarding batch program access to segmented records, this segmented record feature may be considered for installations with real storage of less than 144K bytes, which are unable to use the DL/I products because of insufficient real storage. See "Data Base Selection Criteria" at the end of this chapter for more detail.

RECOVERY CONSIDERATIONS

Recovery of information in the event of system or program failure must be considered in data base design. This subject is discussed in more detail in Chapter 8, but is introduced here to identify those factors which are relevant to the recovery of CICS/VS file control data sets. The optional journaling feature of CICS/VS permits a record to be automatically logged by file control to the system log and/or to be journaled to a user journal data set. The subset option of CICS/DOS/VS does not support automatic logging or automatic journaling.

Automatic Logging

Automatic logging is required if data set backout is to be supported by CICS/VS on emergency restart following an uncontrolled shutdown. Automatic logging is specified in the file control table entry of each data set for which backout is to be supported by specifying LOG=YES. (See *CICS/VS System Programmer's Reference Manual*.) On any update, deletion, or addition of a new record, the "before" image is automatically recorded on the CICS/VS system log if automatic logging is specified. The system log is journal number 1.

Automatic Journaling

Automatic journaling allows the user to specify data set activity to be recorded on any CICS/VS journal. For example, automatic logging records the "before" image of a record to be updated and goes to the system log. Automatic journaling may specify that the "after" image is to be recorded also. Additional journaling activity is identified in the FCT entry for each data set through use of the JREQ parameter, and may be directed to a user journal data set or the system log through use of the FCT JID parameter. This may be desired if a chronological record of all data set activity is to be maintained. It permits implementation of a user-written recovery program to recover a data set from a previous backup copy if an unrecoverable I/O error is detected.

The information recorded on automatic logging, and on automatic journaling, contains the identification of the task which carried out the update, deletion, or addition, the transaction code and terminal identification involved with that task, the time of day, other information required by the CICS/VS journal control program, and an exact image of the record which was updated or deleted, or the record identification of an added record. These records are written to the CICS/VS system log and/or relevant journal data set in chronological sequence.

These logged updates, deletions, and additions to the CICS/VS system log are utilized by CICS/VS on emergency restart following uncontrolled shutdown to back out the data set activity of in-flight tasks. Additional restart information can be found in Chapter 8.

DL/I PRODUCTS

This section introduces the concepts, user design considerations, and advantages of the DL/I products. It is strongly recommended that all CICS/VS users read the section in its entirety, unless they already have prior DL/I experience.

This section is intended only as an introduction to some of the significant features of DL/I, as used in the CICS/VS environment. It should not be considered as a substitute for the DL/I product documentation. No attempt is made to describe the operation of the various DL/I products in depth, nor to describe in detail the design of DL/I data bases. DL/I is discussed only in sufficient detail to enable the CICS/VS system designer to evaluate the various DL/I products and the CICS/VS File Control facilities as data base support for online applications to run under control of CICS/VS. Refer to the appropriate DL/I General Information Manual, System/Application Design Guide, and other DL/I manuals listed in the preface of this publication for further information about these DL/I products and the design of DL/I data bases.

This chapter on data base design concludes with a discussion of various selection criteria which can assist the system designer in his choice of the appropriate data base support to be used for the online applications being designed.

CICS/VS enables data bases created and maintained by the following DL/I products to be accessed by CICS/VS application programs:

- DL/I ENTRY

- DL/I DOS/VS

- IMS/VS DL/I

DL/I ENTRY

DL/I ENTRY provides a subset of the data base facilities offered by DL/I DOS/VS. It utilizes DOS/VS SAM and VSAM on which DL/I access methods HSAM and HISAM are organized. Data bases may be created, maintained, and operated upon by batch processing programs. In addition, CICS/VS application prcgrams may retrieve, update, add or delete information in DL/I ENTRY HISAM data bases cnline. However, DL/I ENTFY dces not support logging of DL/I activity, and does not provide data base recovery utilities that are available with DL/I DOS/VS and IMS/VS.

DL/I DOS/VS

DL/I DOS/VS provides a subset of the data base facilities cffered by IMS/VS and utilizes DCS/VS SAM and VSAM as its standard access methods, on which are organized the CL/I access methods, HSAM, HISAM, HIDAM, and HDAM. These access methods are descrited later in this chapter. Data bases may be created, maintained, and operated upon by batch processing programs. In addition, CICS/VS application programs may retrieve, update, add, and delete informaticn in DL/I DOS/VS data bases online. Data base reccvery utilities are supplied for data base backout in the event of an uncontrolled shutdown and for data recovery following an unrecoverable I/O errcr.

IMS/VS DL/I

IMS/VS DL/I operates under ccntrol cf OS/VS1 cr OS/VS2. It utilizes VSAM as its standard access methcd (and also BISAM and a EDAM access method called OSAM). DL/I access methods HSAM, HISAM, HIDAM, and HDAM are organized on VSAM (see later in this chapter). Data bases may be created, maintained, and operated upon by batch prccessing programs. In addition, CICS/VS application programs may retrieve, update, add, and delete information in IMS/VS DL/I data bases online. Data base recovery utilities are supplied for data base backout in the event of an unccntrolled shutdown and for data recovery following an unrecoverable I/O error.

Unless specifically stated otherwise, the following discussion of DL/I support applies to each of the CL/I products previously described.

DL/I ACCESS FROM CICS/VS

Access to DL/I data bases online from CICS/VS applicaticn programs is achieved using the multitasking facilities of CICS/VS. Any CICS/VS application programs may ccncurrently access the same data base, up to the maximum number of active DL/I tasks specified for the CICS/VS partition, or the maximum number of concurrent DL/I tasks specified during initialization of the relevant DL/I product with CICS/VS.

DL/I ENTRY permits concurrent data base access up tc a maximum of 8 tasks, DL/I DOS/VS up to 2£5 tasks, and IMS/VS DL/I up to 15 tasks.

DL/I utilities are used tc describe the physical organization of data bases and the way in which programs will logically access the data bases defined. In addition, a number of DL/I utilities are provided to allow recovery of data bases in the event of I/O errors or system failures.

INTRODUCTION TO DL/I

DL/I is a general-purpose data base control system that executes in a virtual storage environment under DOS/VS, OS/VS1, or OS/VS2. It has been designed to simplify the user's task of creating and maintaining large common data bases to be accessed by various applications. Its design is open-ended, which allows future DL/I functions to be added without affecting existing functicns. DL/I also allows growth to online applications through an interface with CICS/VS, and through the data communications feature cf IMS/VS.

DL/I has been developed by IBM to serve two application areas:

- Batch processing
- Online processing

In batch processing, single data base transactions requested by applications are accumulated and processed periodically against the data base. Because of the elapsed time, data in the data base is not always current. The use of batch processing should depend on how current the user's information must be, viewed in relaticn to the cost of other methods of processing data.

For online processing, IMS/VS DL/I provides a data communications feature, and also a CICS/VS DL/I interface facility. With DL/I ENTRY and DL/I DOS/VS, data communications support is provided only by a CICS/VS DL/I interface facility. The use of online processing, as opposed to batch processing, enables a response to be generated for each transaction as it is requested. This reduces the elapsed time inherent in batch processing systems and allows the user to maintain current data for his applications.

Traditionally, data used by application programs is organized in data sets. Each data set is physically structured to present data in the physical sequence and format required by the particular application program,

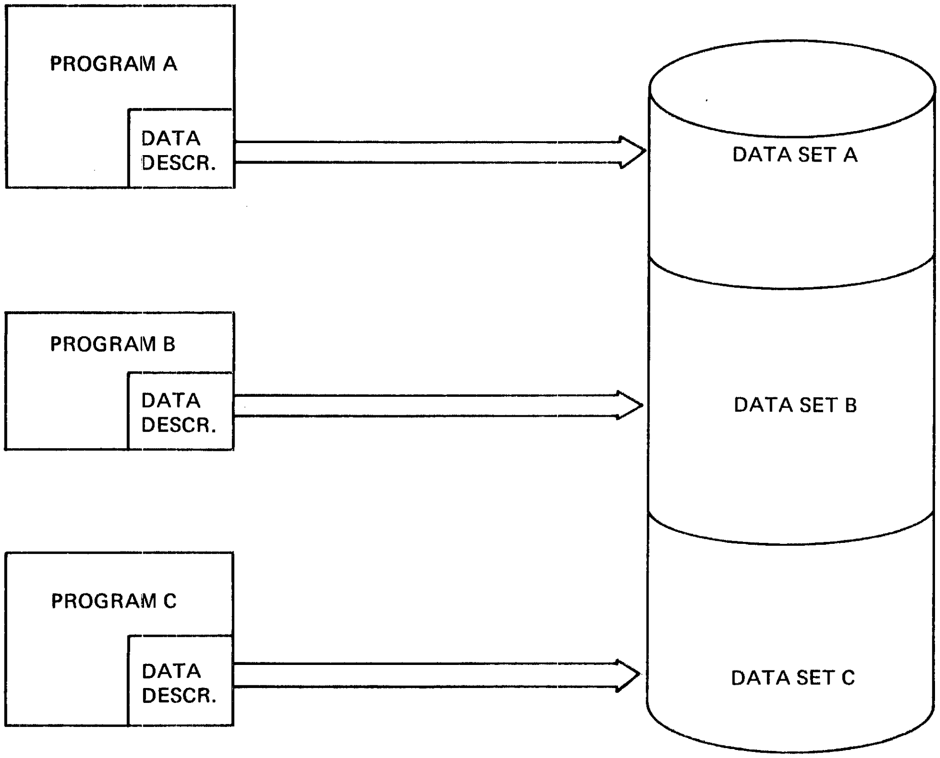


Figure 5-27. Traditional Data Set Approach

and each program contains a description of the data set organization and record format as an integral part of the program (see Figure 5-27). When the same data is shared by many applications (common data), the data is duplicated on different data sets so that it can be presented to each application program in the physical sequence and format required. This duplication uses additional storage space and results in increased maintenance time and cost, since the same data has to be maintained simultaneously in many locations. Futhermore, when the data set organization or record format must be changed, each program which accesses that data set must be modified to reflect the changes. This traditional data set approach is illustrated in Figure 5-27.

DL/I enables programs to be freed from their dependence on data set organization and record format. The description of the physical organization of a data base is removed from programs and contained in a separate data description table. Each program utilizes this data description. DL/I extracts the requested information from the data base to present to the program. This is illustrated in Figure 5-28. Because programs using DL/I are no longer dependent upon the physical organization of data, when the data base organization must be changed, generally only the data description table need be changed. Programs which access the data base using the data description in most cases need not be aware that the data base has changed, and generally need no modification.

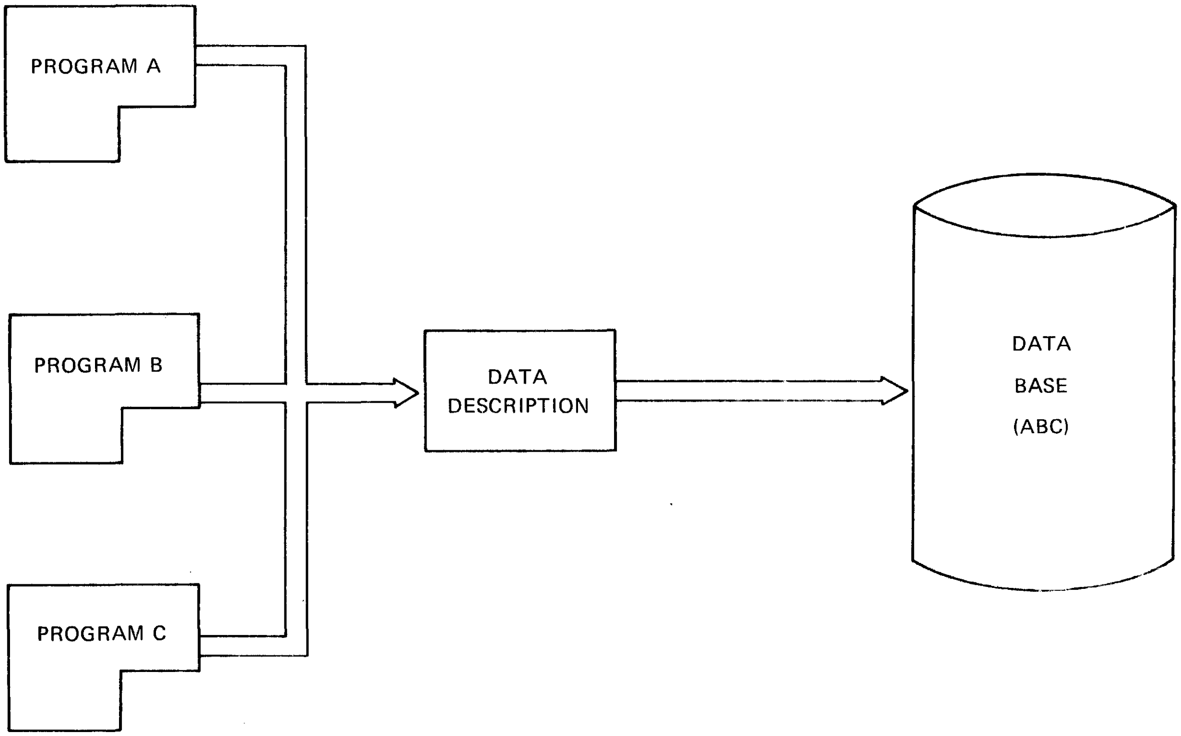


Figure 5-28. DL/I Data Base Approach

All application data is stored in one or more data bases in a hierarchical fashion; that is, the most significant data resides on hierarchically higher levels, while less significant but related data (dependent data) appears on hierarchically lower levels, as illustrated in Figure 5-29. This hierarchical approach enables programs to view data in a data base apart from its physical organization. Through the use of a concept called "sensitivity," each application program views only that data in the data base which it uses. (Sensitivity is discussed further, in "Logical Data Structures," later in this chapter.)

DL/I accesses data in the data base and presents only the information requested by the program. The data presented to the program by DL/I is called a "segment." A program requests a segment from DL/I by issuing a DL/I call.

In practice, a system designer reviews the data requirements of all applications (as illustrated at the start of this chapter), then defines the data base or bases. To create a data base, the user defines to DL/I a common data structure and format that serve his applications and loads his application data into that data base.

The definition of the data base is provided by a data base description (DBD), and a DBD is required for each data base (see Figure 5-29). This is generated prior to the loading of data into the data base, by assembling a set of DBD macro instructions which define the data base.

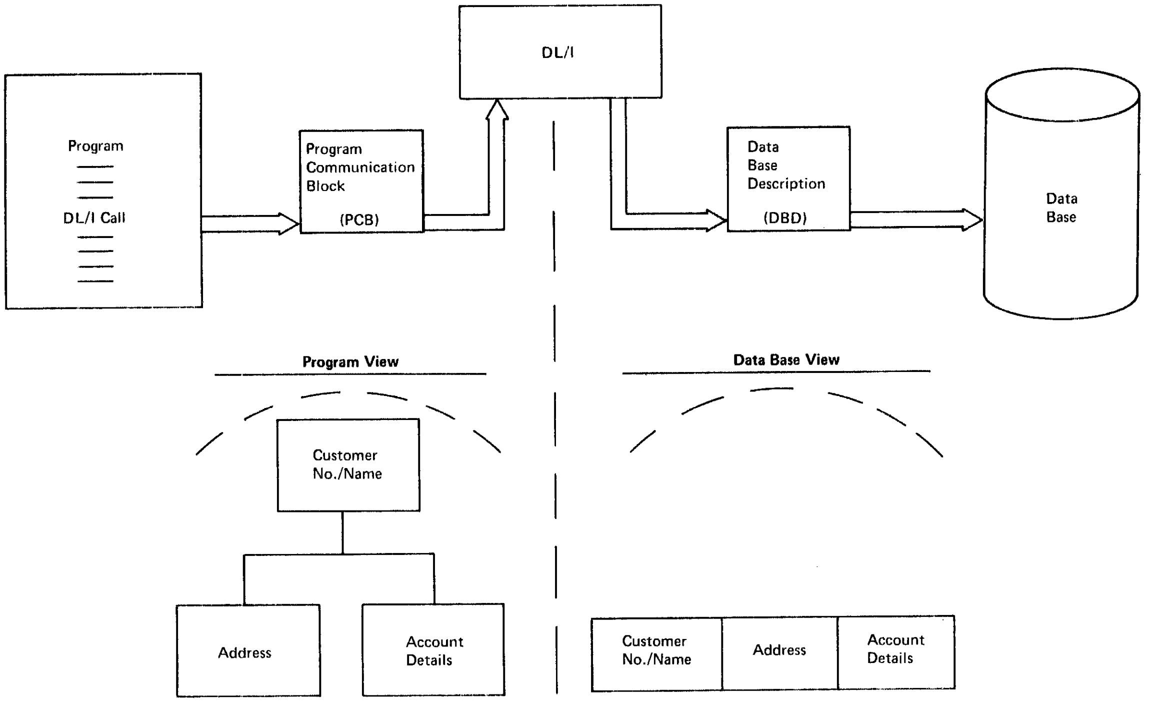


Figure 5-29. DL/I Data Base Access

The second definition required is the program specification block (PSB). There is one PSB for each application program. The PSB defines to DL/I:

- The data bases accessed by an application program. Associated with each data base used by an application is a PCB. One or more PCBs exist within a PSB.

- Each PCB identifies the sensitive data (segments) in the data base, available to the application program which uses the PSB.

- Each PCB identifies the way in which the program views the data base.

- More than one PCB defined in a PSB allows the application program to process multiple data bases.

The PSB is generated by assembling a set of PSB macro instructions which define the above factors.

An application program specifies its associated PSB to be used to access data bases. Each PCB is used to pass information to DL/I identifying:

- The name of the DBD describing the particular data base

- The type of calls (or processing options) which the program will use to access the data base

- The number of segments to which this program is sensitive

In addition, fields are provided in the PCB for DL/I to pass information back to the program, such as:

- The level number in the hierarchical structure of the last retrieved segment

- The DL/I status code indicating the result of the last DL/I call issued by the program

- The name of the last segment retrieved by DL/I

- The key of the last segment retrieved

Through DL/I's use of the DBD and PSB, application programmers can write their programs without much regard to the physical structure of data. Instead, they refer only to segments of data as needed by the program, without consideration for the physical location of that data in the data base.

ADVANTAGES OF DL/I

DL/I provides application independence from access methods, from physical storage organization, and from the characteristics of the devices on which the data of the application is stored. This independence is provided by a common symbolic program linkage and by data base descriptions external to the application program. A reduction in application program maintenance is generally realized.

DL/I provides for the reduction, and possible elimination of, redundant data or sharing of common data. The majority of the data utilized by any company has many interrelationships that can cause significant redundant storage of data if conventional organization and access methods are used. For example, manufacturing and engineering departments work with subset data which is also useful to quality control.

The storage organization and access methods employed by DL/I facilitate data integration with a minimum of data redundancy. However, if an analysis of a company's data shows that all of the data cannot be placed in a single common data base, DL/I allows the user the additional capability of physically structuring the data across more than one data base. Before DL/I, application programmers frequently did not have the time or ability to integrate other data with their own data to eliminate redundancies without the necessity of a major rewrite of the application programs involved.

SEGMENT DESIGN

The smallest element of data that may be retrieved by DL/I is a segment. A segment contains one or more logically related data fields, and is fixed-length for DL/I ENTRY, and fixed- or variable-length for DL/I DOS/VS and IMS/VS DL/I.

The design of segments to be utilized in DL/I data bases is dependent upon a number of factors. Some of these are:

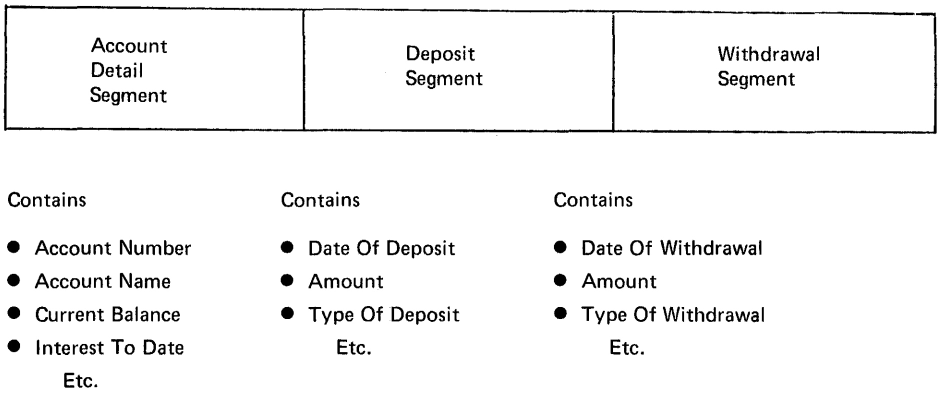
- The amount of data required at a time by the application program

  A segment is the smallest element of data which may be operated upon in a DL/I data base and may comprise one or several related fields. The allocation of fields to segments is generally based upon common usage of the data contained within those fields. For example, if a particular group of fields may be required by one application program, it may be desirable to group these into a segment, if other characteristics of those fields are similar (see below).

- The multiple occurrence of information

  The multiple occurrence of information may require each separate occurrence to be defined as a segment. For example, in a savings bank system there may be many deposits or withdrawals against each savings account (see Figure 5-30). The savings account details such as the account number, account name, current balance, and interest-to-date may comprise one segment which, because it may be required by many applications processing those accounts, may be defined as a "root segment." A root segment is at the highest level and generally contains information required by most application programs.

  Each deposit transaction may contain fields such as date of deposit, amount of deposit, and type of deposit. These fields may be defined together as being a deposit segment. Similarly, a withdrawal transaction may contain fields such as date of withdrawal, withdrawal amount, and type of withdrawal. These fields make up a withdrawal segment. Because there may be many deposit and withdrawal segments for a savings account root segment, defining each transaction as a separate segment will enable individual transaction details to be readily accessed. Figure 5-30 illustrates the multiple occurrence of deposit and withdrawal segments based upon a savings account root segment.



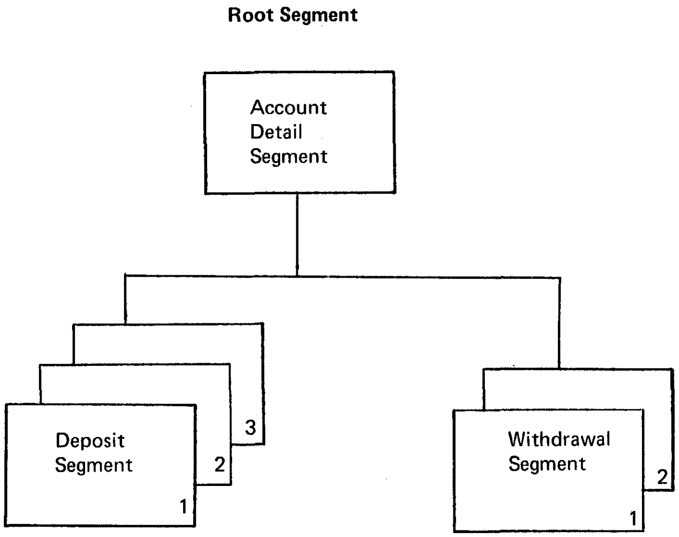


Figure 5-30. Multiple Occurrence of Segments

- The sensitivity to data by various programs

  The requirement for certain application programs to have access to certain infcrmation (that is, be sensitive tc certain segments), and not be able to access (or be sensitive tc) other information or segments, assists in the segment design. For example, related fields of information relevant to a particular applicaticn program may be designed as a segment. That applicaticn program may then be made sensitive to this segment.

- Variable-length segments

  In the case of IMS/VS DL/I, the ability to support variable-length segments may dictate those fields which should be defined as comprising a segment. Fcr example, a customer data set contains customer name and address. A customer name is normally variable in length, as are custcmer address lines. In addition there may be a variable number of address lines. In this case, it may be desirable to define the customer name as one segment, called, for example, the name segment, and to define each address line as a separate address segment. There may be multiple address segments to allow a variable number of lines in a customer address.

- Application-dependent requirements for information grouping

  The design of segments, as well as the fields which make up those segments, is dependent upon the application. An application-dependent requirement may be the extent to which information in the data base might be expected to change in the future. If there is a high likelihood that certain fields within a segment may change, it may be desirable to define these potentially changeable fields in a single segment, or a small number of segments. The extent of program maintenance because of a change in the segment contents is therefore reduced. DL/I enables new segments to be defined very readily, without affecting application programs which do not need to access those segments.

To assist in the design of segments, consult the appropriate _System/Application Design Guide_ for DL/I ENTRY, DL/I DOS/VS, or IMS/VS DL/I.

DATA INDEPENDENCE

Under DL/I, the data base concept gives the user data independence between the physical storage of data and the application programs which access the data in various ways. Physical data storage is accomplished through the use of two unique DL/I storage organizations:

- Hierarchical Sequential

- Hierarchical Direct

The Hierarchical Sequential organization is supported by two DL/I access methods: the Hierarchical Sequential Access Method (HSAM) and the Hierarchical Indexed Sequential Access Method (HISAM). The Hierarchical Direct organization is supported by two DL/I access methods: the Hierarchical Direct Access Method (HDAM) and the Hierarchical Indexed Direct Access Method (HIDAM). The application program interface with these two organization types and four access methods is totally symbolic. The application is typically insensitive to the physical data organization or the access method used. The various data base organizations and access methods are discussed in more detail in the _DL/I DOS/VS System/Application Design Guide_ and the _IMS/VS System/Application Design Guide_.

To provide this _independence_, three definitions are required prior to the use of the data base by a program:

- The segments within a logical data base record to which a program wishes to be sensitive

- The logical data base record structure represented by one or more segments from one or more physical records

- The data base organization and access methods

These definitions are made through the use of the following data base definition functions:

- Data Base Description (DBD) Generation

- Program Specification Elcck (PSB) Generation

- Application Control Elcck (ACB) Generation

The use of the DBD and PSE was described above. The generated PSBs and DBDs to be used in an cnline environment are processed offline by an application ccntrcl blcck (ACE) utility program before the first use, to combine their separate specifications into an application control tlcck (ACB) for use cnline. The DED is changed by the ACB utility into a data management block (DMB) describing the specific use of the physical data sets representing a data base. The PSB is changed by the ACB to describe the specific use of the lcgical structures available frcm one cr more data tases to the asscciated application prcgram for prccessing. It is thrcugh the use of the DBD, PSB, and ACB that DL/I achieves its data independence.

20 bytes		20 bytes									60 bytes	6 bytes				
Customer Number	Credit Limit	Customer Name	Postal Address				Ship-To-Address				Customer History	Account History – Arrears				
			Line 1	Line 2	Line 3	Line 4	Line 1	Line 2	Line 3	Line 4		Current Balance	One Month	Two Months	Three Months	Over Three Months

- Record Format: Fixed-Length
- Record Length: 282 Bytes

Figure 5-31. Custcmer Acccunt Reccrd Format

LOGICAL DATA STRUCTURES

Application prcgrams written tc use DL/I deal with logical data structures. A logical structure refers to the manner in which the application program sees the data. The logical data structure is always a hierarchical structure cf segments. Programs written tc process logical data structures can te independent of the physical data structure. A physical structure refers to the manner in which the data is stored on a tape or on a direct access storage device. An application program never deals directly with a physical data structure. Most data processing informaticn, regardless of industry, can and should be viewed as a logical data structure. Customer account information is chosen here fcr explanatory purposes because of its commonality.

Figure 5-31 illustrates a typical customer account record and describes the physical structure of the record as it might appear on tape or on a direct access storage device. Each of the field groups (name, postal address, ship-to address, customer history, and account history) is referred to as a segment. These segments usually contain more basic data elements. For example, one of the data elements typically included in the name segment is a customer number. In addition, the record might contain multiple address lines and account fields. This is typical if address and account history are part of the customer account information.

This same data record appears in Figure 5-32 as a DL/I logical data structure. The name, postal address, ship-to address, and customer and account history fields are considered DL/I segments. Each segment of information may be made up of several fields.

The logical data structure in Figure 5-32 represents a hierarchical relationship. Data relationships described by this hierarchical structure have only one segment at the first level, level zero, and may have multiple segments at subordinate levels (for example, level 1) in the hierarchical structure. Each dependent segment in the hierarchy has only one parent or immediate superior segment. The logical representation is sometimes called a tree structure. In Figure 5-32, the name segment with its associated address and account segments constitutes a logical data base record.

Through the concept of program sensitivity, DL/I allows a program to be structured so that only those segments of information that are relevant to the processing being performed are presented to it. For example, an account program could be written for only the name and account segments of the logical data base structure, as shown cross-hatched in Figure 5-32. The program need not be aware of the existence of the address segments.

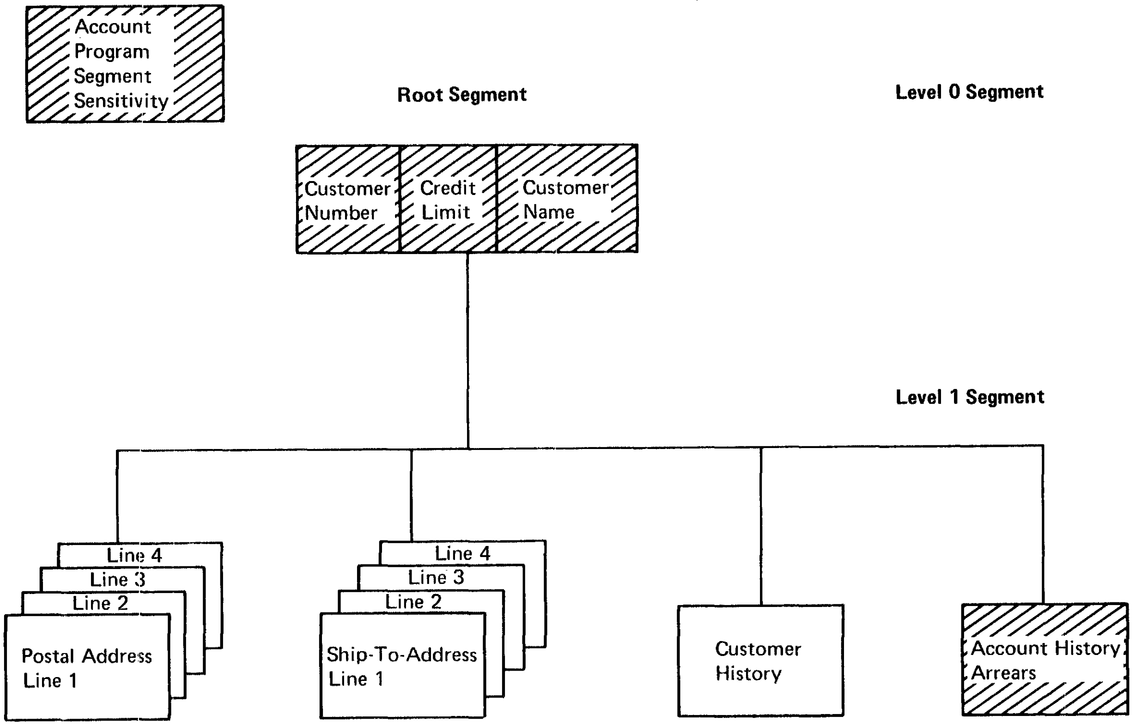


Figure 5-32. Customer Account Logical Structure

The DL/I data base capabilities allow for handling hierarchically related logical data structures of considerable variance. The maximum number of segment types is limited for DL/I to 255 per logical data base record (63 for DL/I ENTRY). A maximum of 15 segment levels can be defined in a logical data base record.

Figure 5-33 represents an example of a logical data structure for a banking savings account. The structure consists of four segment types: account detail, account name, deposit, and withdrawal.

Note: To illustrate DL/I terminology, the account name segment has been specified in the logical structure on a level lower than the account detail segment, but higher than the deposit and withdrawal segments. This implies that a relationship of deposits and withdrawals to each person operating on the account is significant to the application. In actual practice, it is generally not necessary to identify the person who made a deposit, or withdrawal, but only the fact that such a transaction was made against the account. In this case, the account name segment would be placed on the same level as the deposit and withdrawal segments. It has been shown at a higher level than these segments in Figure 5-33, only to illustrate DL/I terminology.

The logical structure in Figure 5-33 shows that account detail is a root segment. The account name segment is a dependent segment of the account detail segment, and the deposit and withdrawal segments are dependents of the account names.

Account Detail			Account Names		Deposits			Withdrawals	
Acct. No.	Acct. Type	Curr Bal.	Name1	Name2	Deposit 1	Deposit 2	Deposit 3	Withdrl 1	Withdrl 2
67321			Smith, John	Smith, Mary	$600	$200	$500	$20	$50

Savings Account Record

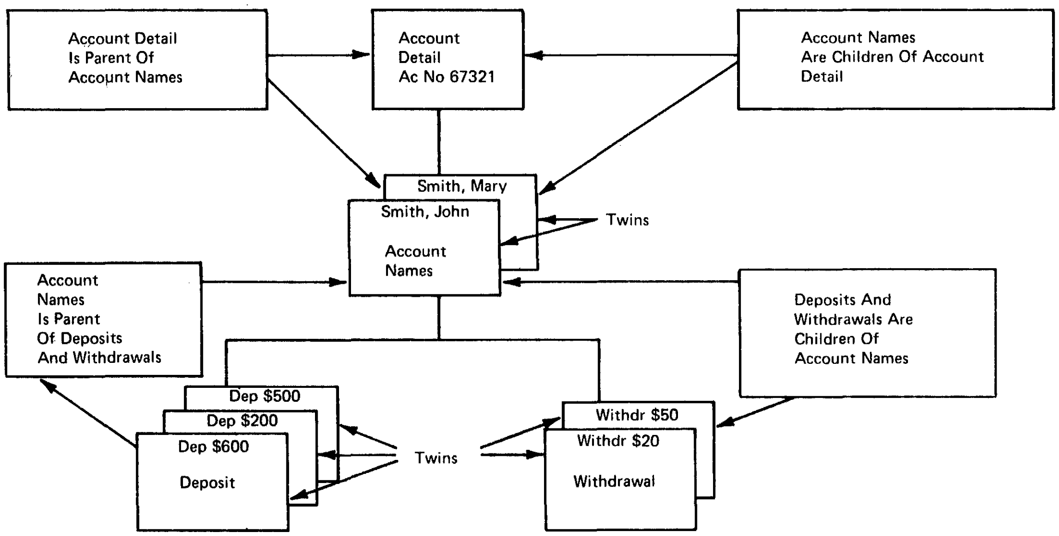

Account Detail Is Parent Of Account Names
Account Detail Ac No 67321
Account Names Are Children Of Account Detail
Smith, Mary
Smith, John
Account Names
Twins
Account Names Is Parent Of Deposits And Withdrawals
Deposits And Withdrawals Are Children Of Account Names
Dep $500
Dep $200
Dep $600
Deposit
Twins
Withdr $50
Withdr $20
Withdrawal

Figure 5-33. Savings Account Logical Structure

A dependent segment relies on some higher level segment for its full meaning or identification. Since the number of account name segments may vary from one savings account to the next, logical data base records will vary in length according to the number of segments occurring in the hierarchy of the data structure.

The root segment in Figure 5-33 contains the specific account details for account number 67321. There are multiple account name segments for persons using this account: John Smith and Mary Smith.

Also, for this account, there are two withdrawal and three deposit segments. The data base has four segment types, and this particular data base record consists of eight segments because of the multiple occurrences of the account name, deposit, and withdrawal segments.

The segments immediately above and below a given segment are called the "parent" and "child" segments respectively. In Figure 5-33, the account detail segment for account 67321 is a parent of the two name segments. Each account name segment is a child of the account detail segment. The deposit and withdrawal segments subordinate to the account name segments are their children. All occurrences of a particular segment type dependent on a particular parent segment are called "twin" segments. The account name segments for John Smith and Mary Smith are twins of each other.

Refer to the System/Application Design Guide for the relevant DL/I product for further discussion of logical structures.

DATA BASE ACCESS CALLS

A common symbolic program linkage and data base description allow batch and online application programs to request DL/I to:

- Retrieve a unique segment (GET UNIQUE)

- Retrieve the next sequential segment (GET NEXT)

- Replace the contents of an existing segment (REPLACE)

- Delete the data in an existing segment (DELETE)

- Insert a new segment (INSERT)

Additional data base access calls are provided to enable all dependent segments based on a particular parent segment to be retrieved sequentially. The retrieval access calls may also specify that a record being retrieved is to be held, because it will subsequently be updated, replaced, or deleted.

Application programs written in American National Standard (ANS) COBOL, PL/I, and Assembler Language utilize the CALL statement facility in these languages to perform the input/output functions listed above. The external data base descriptions (DBDs) and program specification block (PSB) describe the physical data organization and logical data structure of the data base to DL/I. Because of this approach to data reference, input/output operations and associated system control blocks are not compiled into the application program. This avoids program dependence upon currently available access methods and physical storage organizations. Other application data can be added to this data base without necessitating changes to application programs that use the data.

DATA BASE ORGANIZATION AND ACCESS METHODS

DL/I supports two basic physical storage organizations as discussed briefly above. The first organization, Hierarchical Sequential, provides the base for both the Hierarchical Sequential Access Method (HSAM) and the Hierarchical Indexed Sequential Access Method (HISAM). The second organization, Hierarchical Direct, provides the base for two more accessing techniques, the Hierarchical Direct Access Method (HDAM) and the Hierarchical Indexed Direct Access Method (HIDAM). HDAM uses an addressing algorithm for direct access support of the hierarchical direct organization. Standard addressing algorithms are provided by DL/I, or may be supplied by the user installation for each HDAM data base. HIDAM is an indexed access support of this hierarchical direct organization.

Each DL/I access method and data base organization will now be briefly introduced. Refer to the System/Application Design Guide for the relevant DL/I product for a more detailed description of each data base access method.

Hierarchical Sequential Organization

The primary differences between hierarchical direct and hierarchical sequential organizations are the manner in which segments are related, and the techniques of data storage and access. Segments in the hierarchical sequential organization that represent one data base record (that is, a physical hierarchical tree structure) are related by physical adjacency. This requires that segments which represent one data base record be contained in a variable number of segment blocks unique to that data base record.

HSAM Record Format

HSAM is used for sequential storage and access on tape or direct access storage. The DOS/VS and OS/VS sequential access methods (SAM and BSAM respectively) provide the data management services for HSAM.

Physical blocks (a variable number) required to contain a particular data base record are accessed by physical adjacency (HSAM).

HISAM Record Format

HISAM is used for indexed sequential access to the hierarchical sequential organization. data base record may be directly accessed through an index. Segments within a data base record are related by physical adjacency. All physical blocks used to store the segments of a single data base are related by direct address pointers. The virtual storage access method (VSAM) provides data management services for DL/I ENTRY, DL/I DOS/VS, and IMS/VS DL/I. IMS/VS can optionally use BISAM and a BDAM-like access method called OSAM for data management services. (However, DL/I facilities such as variable-length segments, or secondary indexing (see later in this chapter) are only available to data bases which utilize VSAM.) Generally, a HISAM data base is comprised of one VSAM key-sequenced data set and one entry-sequenced data set.

Note: DL/I ENTRY only uses one VSAM key-sequenced data set to support a HISAM data base.

Hierarchical Direct Organization

Segments in the hierarchical direct organization that represent one data base record (that is, a physical hierarchical tree structure) are stored in one or more physical blocks. However, all segments in that data base record, rather than the physical blocks containing the data base record, are related by direct addresses. Each segment in a data base record refers to segments of the same type, as well as to adjacent segment types, through direct addressing. Physical blocks that contain segments of the data base record are not related by direct addressing.

In the hierarchical direct organization, the space requirement for each segment is increased from that required in the hierarchical sequential organization. This space is required to accommodate direct addresses. However, the following advantages are gained:

- More rapid direct access to segments within a data base record.
- The ability to share space in a direct access storage block across multiple data base records. One physical block may contain segments from different data base records. This may result in a considerable saving of data base storage space.
- The ability to reuse space occupied by deleted segments through the maintenance of free space addresses.

HDAM Record Format

HDAM is used for direct algorithmic randomized access to records in the hierarchical direct organization. Direct access is available to each data base record. VSAM provides the data management services for HDAM. OSAM provides optional data management services for IMS/VS DL/I, but (as discussed previously for HISAM) facilities such as variable-length segments or secondary indexing cannot be utilized with OSAM. DL/I ENTRY does not support HDAM.

HIDAM Record Format

HIDAM is used for indexed access to the hierarchical direct organization. Indexed access is available to each data base record. VSAM provides the data management services for HIDAM for DL/I DOS/VS and IMS/VS DL/I. BISAM and OSAM provide optional data management services in HIDAM for IMS/VS DL/I. (See reference to BISAM and OSAM limitations above for HISAM and HDAM.) DL/I ENTRY does not support HIDAM.

Refer to the *System/Application Design Guide* for the appropriate DL/I product for further information concerning data base organization and access methods.

LOGICAL STRUCTURE DESIGN

The preceding discussion has established the concept and organization of DL/I data bases. As discussed previously, the smallest data element which can be accessed by an application program is a segment. The design of segments is a key factor in the design of DL/I data bases. Following this segment design, the design of logical data base structures should be considered.

Design Factors

Logical structure design depends upon the application. However, some of the factcrs which shculd be considered in the design of logical structures are:

- The utilization of dependent segments

  A prcgram may wish tc access the infcrmation in the root segment of a data base record. In the savings account example in Figure 5-33, the account details will be contained in the root segment.

  Follcwing the accessing cf this root segment, the application requirements may dictate that segments dependent on this root must te accessed. In the savings acccunt example in Figure 5-33, the account name segments, then the deposit segments and the withdrawal segments may be accessed either individually or in their crder of occurrence.

- The sensitivity cf programs to segments

  An account inquiry prcgram, fcr example, may wish to access only the account detail (rcct segment) and the acccunt name segments. In this case, the program wculd te defined as teing sensitive only to these segments.

  Similarly, an account processing program may need to access only the account detail (roct segment) and the depcsit and withdrawal segments, withcut requiring reference to the account name segments. This enables depcsit and withdrawal transacticns to be retrieved, updated to reflect pcssitle ccrrection, cr added as new deposit or withdrawal transacticns are received against an account. In this case, the account prccessing program would be defined as being sensitive only tc the account detail root segment, and the withdrawal and deposit segments, but not sensitive tc the account name segment.

- Application factors

  In a savings account example, cnce an account is opened, information such as the name segment may nct be changed or added to significantly. In fact, if there is only one name for each account, it may be logical tc place the name within the root segment itself. However, because there may be more than cne rame associated with an account, the pcssible multiple cccurrence of names may dictate that these be separate segments which are dependent upon the account detail root segment and further describe that account. The deposit and withdrawal segments reflect activity of transacticns against the account. Because there may be multiple deposit and withdrawal transactions, it is logical to make them separate segments which are also dependent (lcgical children) upon the account detail root segment.

  Similarly, in an insurance pclicy information system, a policy data tase may contain policy details in the root segment, policy beneficiaries in name segments dependent upon the root segment, and policy claims and renewals transactions dependent upon the name segments (see Figure 11-19).

  In a manufacturing wcrk crder system, the manufacturing wcrk crder data base may contain all cf the manufacturing wcrk crder information in a root segment, with multiple status information recorded at different stages cf manufacture in status segments dependent upon the root segment (see Figure 11-5).

In a medical patient information system, the patient's name, birth date, and other details would be contained in the root segment, with his address in separate address line segments dependent upon the root segment. Dependent upon these address segments would then be visit segments, diagnosis segments, and treatment segments of which there may be multiple occurrences for a patient (see Figure 11-23).

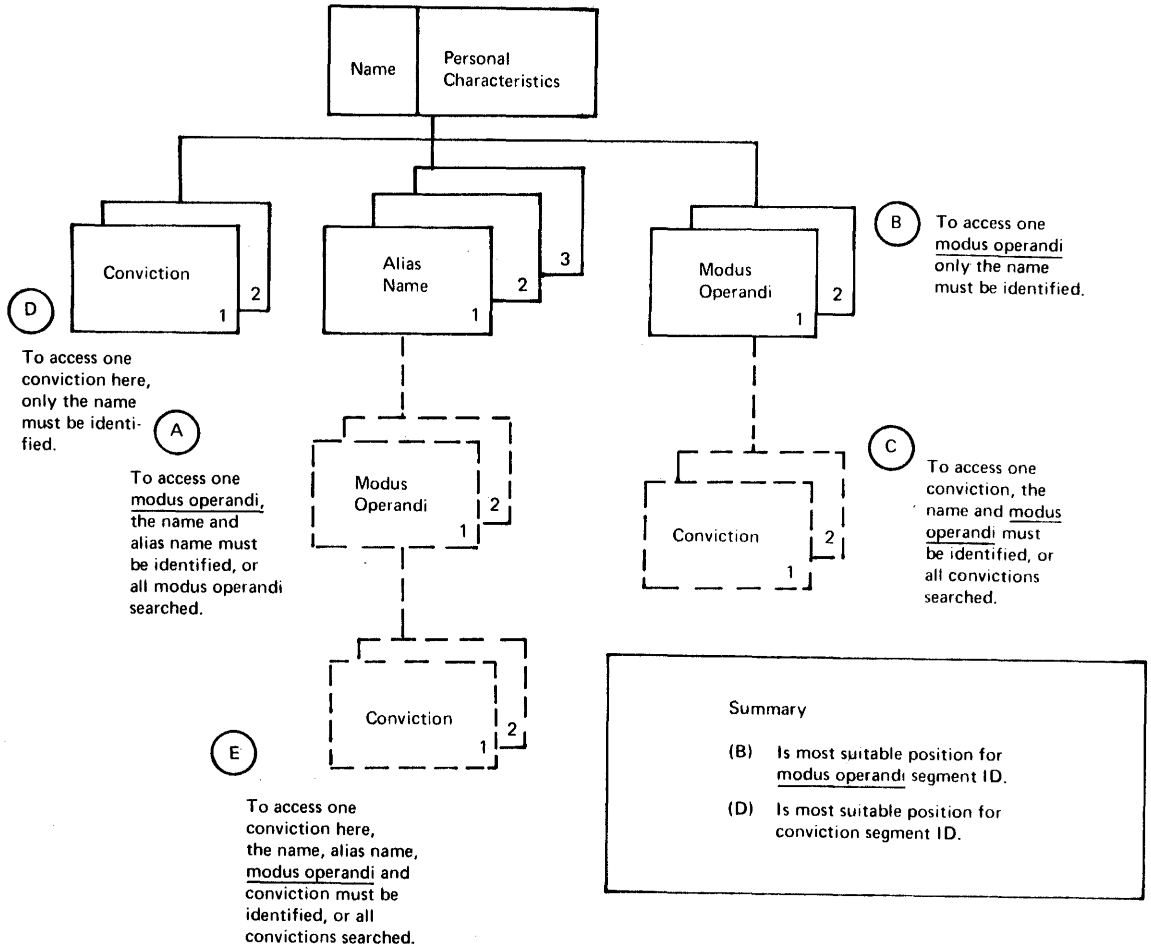
Segment Reference Design Factors

In a police information system, the logical structure may be designed for a criminal data base by placing the personal characteristics of the criminal in a root segment together with his true name. This is illustrated in Figure 5-34 for reference during the discussion in the following paragraphs.

Alias names may be dependent segments of the root segment. The particular mode of operation for various crimes may be each stored in a separate *modus operandi* segment. These *modus operandi* segments may be dependent upon the alias segments (see (A) in Figure 5-34), but more logically are further descriptions of the criminal who is described by the root segment. Thus, the alias segments and *modus operandi* segments are regarded as twins if both are children of the root segment (see (B) in Figure 5-34).

Following the *modus operandi* segments may be various convictions recorded against this criminal (C). Each conviction is contained within a conviction segment, and typically identifies the crime to which the conviction is related, the particular punishment carried out, and the extent to which that punishment has been carried out. Alternatively, the conviction segments may be children of the root segment and twins of the *modus operandi* and alias segments (see (D) in Figure 5-34).

The decision whether to define the logical structure in such a manner that these dependent segments are twins, or children, depends upon the information which will be available for subsequent accessing of this data base record. To access a child segment efficiently requires identification of the parent.



(B) To access one modus operandi only the name must be identified.

(D) To access one conviction here, only the name must be identified.

(A) To access one modus operandi, the name and alias name must be identified, or all modus operandi searched.

(C) To access one conviction, the name and modus operandi must be identified, or all convictions searched.

(E) To access one conviction here, the name, alias name, modus operandi and conviction must be identified, or all convictions searched.

Summary

(B) Is most suitable position for modus operandi segment ID.

(D) Is most suitable position for conviction segment ID.

Figure 5-34. Police Data Base Logical Structure

In this example, if *modus operandi* were a child of alias (A), and convictions were children of *modus operandi* (E), to retrieve information relating to particular convictions for a criminal would require knowledge possibly of his true name, his alias name, and various *modus operandi* (see (E) in Figure 5-34). However, in this application, that identification may not be available to access information relating to convictions. In this case, it would better to define convictions as children of the root segment (D).

If the convictions, *modus operandi*, and alias segments are twins, any segment may be accessed knowing only the true name of the criminal. Knowledge only of a particular alias name, or of a particular *modus operandi* for a criminal, in some cases, may require the use of additional DL/I features, such as logical relationships, or secondary indexing, which are described below.

Alternatively, instead of using secondary indexing, a separate data base may be organized based on alias names, which provides a cross-reference to the original criminal names, and another data base may be organized based upon *modus operandi* which again identifies the criminal's true name. Thus, by accessing the separate data bases first, the criminal may be identified and then further information related to that criminal may be obtained from the criminal data base.

Obviously, the factors which may be considered in the design of logical structures are often very dependent upon application. Refer to the *System/Application Design Guide* for the relevant DL/I product for further discussion of logical structure design.

UPDATES, ADDITIONS, AND DELETIONS

In order to update (REPLACE) segments in DL/I data bases, it is necessary to indicate when the original record is retrieved that there is a possibility that the record may be updated. This is indicated by placing a "hold" on that segment with a GET call, such as GET HOLD UNIQUE, GET HOLD NEXT, or GET HOLD NEXT WITHIN PARENT. The segment presented on a GET HOLD call is then updated by the application program and written back with a REPLACE call. Alternatively, if the segment is not to be updated, this is indicated by the next GET call, without an intervening REPLACE or DELETE. In effect, the HOLD used with the GET call implies that "exclusive control" is to be placed on that segment to prevent the same segment from being concurrently updated by another task. Only when the REPLACE or DELETE call (or the next GET) is issued is the exclusive control released.

When DL/I DOS/VS or IMS/VS DL/I is used with CICS/VS, concurrent updating will be prevented at the segment-type level with the scheduling-by-intent feature. The DL/I task will not be scheduled if it has conflicting intent with another active DL/I task on the same segment type. This may have performance implications, depending upon the frequency of reference to the same segment type by concurrently executing tasks. However, it provides the ability to back out programs independently in the event of an uncontrolled shutdown. VANDL-1 and DL/I ENTRY, used with CICS/DOS/VS, does not perform checks at scheduling time and does not provide specific backout support following an uncontrolled shutdown.

To illustrate the significance of segment intent scheduling on data base design and performance using DL/I DOS/VS or IMS/VS DL/I, consider the following order entry and inventory control application in the distribution industry.

Each store carries its own inventory and enters orders from a terminal located in the store. These orders update the inventory maintained in a central data base for all stores. If the inventory data base is designed with only one segment type for inventory details across all stores, only one task (that is, store transaction) would be scheduled to update that segment type at a time. This may have significant performance implications if several stores were concurrently entering orders, and may result in single thread access to the data base.

However, as each store carries its own inventory, a separate inventory details segment type may be defined for each store (subject to the maximum of 255 different segment types, imposed by DL/I). In this way, each store transaction would be scheduled on its own inventory details segment type, with no conflict with other stores inventory segment types. This will permit multithread access to each stores inventory details, with consequent improvement in performance without any loss in data integrity. However, it will be done at the expense of additional HDAM data base pointers, control block storage, and application program logic.

In the case of an addition (INSERT), the segment to be added is constructed according to the application requirements, and then added to the appropriate part of the data base by identifying the keys of various higher level segments (using a GET HOLD UNIQUE call, for example) and hence the location within the data base.

For a deletion (DELETE), the segment to be deleted may be identified by providing the keys of various higher level segments, and hence the location of the segment within the data base by using a GET HOLD UNIQUE call. Depending upon the hierarchical access method used, the segment may be either physically deleted immediately (HIDAM or HDAM), or logically deleted, and not physically deleted until the next data base reorganization (HISAM).

LOGICAL RELATIONSHIPS

IMS/VS and DL/I DOS/VS provide enhanced data base support through the use of logical relationships. Consider the police information example discussed in Figure 5-34 above. Two data bases will now be considered, a criminal data base and a crimes data base. If each data base were completely separate, criminal identification would be provided within the crimes data base, and crimes identification would be provided within the criminal data base. While this may be quite satisfactory, the use of logical relationships enables redundancy of information to be avoided. For example, the crimes data base, in identifying various criminals associated with a particular crime, is duplicating information which is already on the criminal data base. Similarly, the criminal data base, in identifying information relating to particular crimes, is duplicating information already in the crimes data base.

As described in "DL/I Access from CICS/VS," each DL/I product permits multithread access to DL/I data bases by several concurrently executing CICS/VS tasks. DL/I DOS/VS and IMS/VS DL/I use segment intent scheduling to ensure data integrity; DL/I ENTRY issues I/O requests to CICS/VS file control. File control maintains data integrity through exclusive control and protected resources. In addition, DL/I data bases specified for DL/I ENTRY as data sets in the file control table may also specify automatic logging for implementation of backout support. (See Chapter 8.)

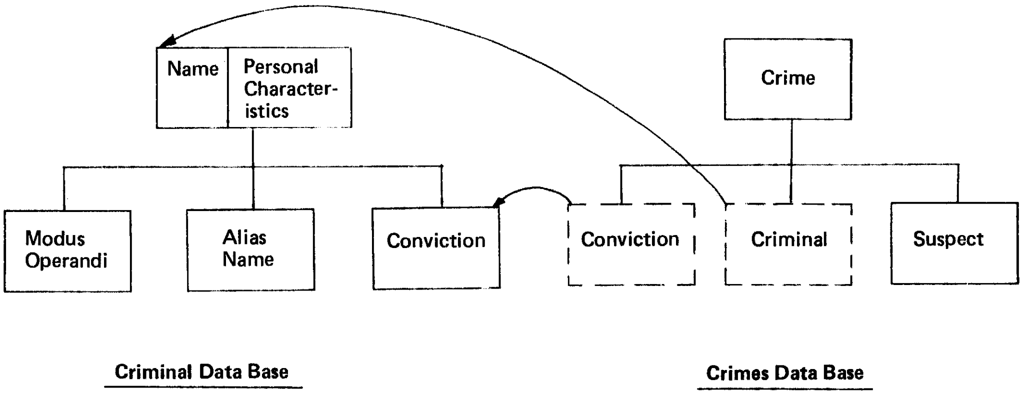


Figure 5-35. Police Data Base Logical Relationships

To avoid this duplication, logical relationships utilize pointers in the criminal data base conviction segments to identify the relevan crimes information in the crimes data base (see Figure 5-35). Similarly, pointers in the crimes data base enable relevant criminal information to be identified in the criminal data base. Crimes information and criminal information need appear only once, each in its own data base.

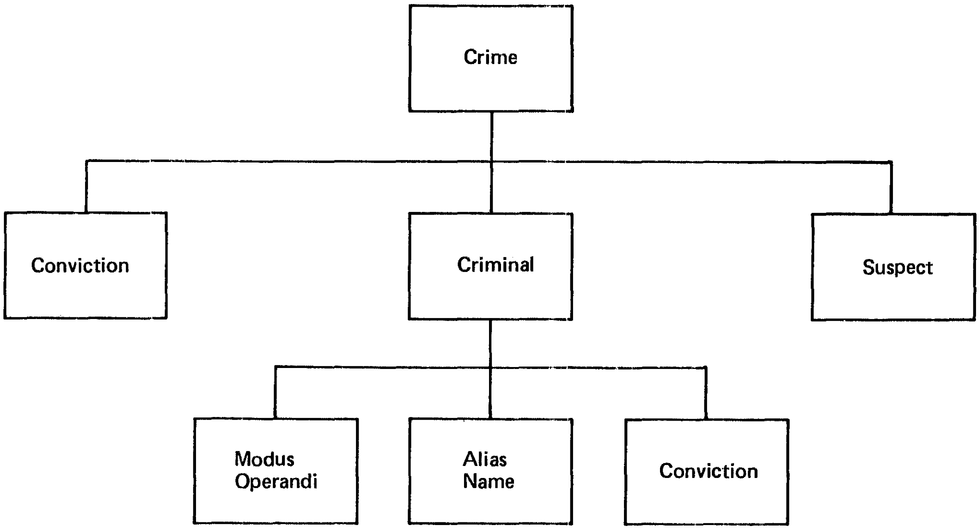


Figure 5-36. Logical Data Base Viewed from Crimes Data Base

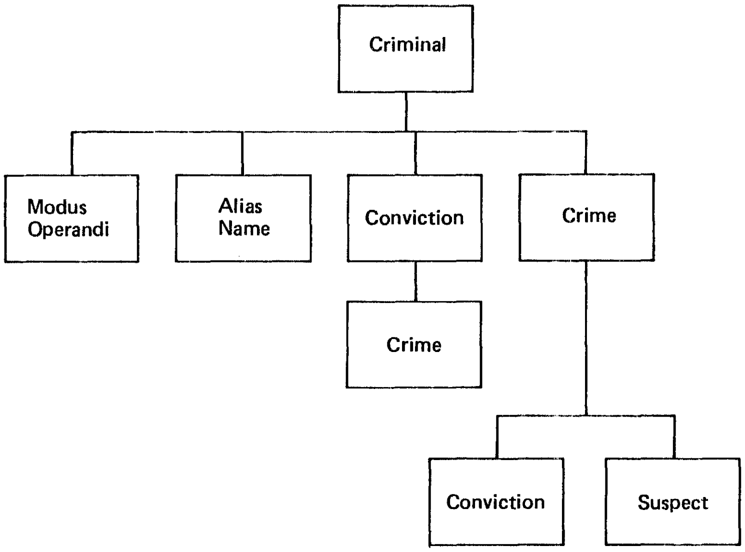


Figure 5-37. Logical Data Base Viewed from Criminal Data Base

Consequently, that information is accessible tc the other data base through the use cf lcgical relaticnshifs.

Through the concept of logical relationships, expanded data structures can be created. These expanded data structures are then available to applicaticn frograms. See Figures 5-36 and 5-37.

Once a logical data base is defined through one or more physical data bases, the logical data base can be accessed by an application program through a PCB.

Logical relationships and logical data bases are available with IMS/VS and DL/I DOS/VS, and limited logical relaticnships are available with DL/I ENTRY.

SECONDARY INDEXING

Secondary indexing is a feature offered by IMS/VS and DL/I DOS/VS, and to a limited extent by DI/I ENTRY, for data bases which utilize VSAM. Effectively, it enables separate indexes tc be developed based upon the data as key or data fields cf particular segments. For example, in the police information system described previcusly, and shown in Figures 5-34 and 5-35, it may be necessary to access information in the criminal data base given the *mcdus operandi*, or given the personal characteristics of a criminal. This can be achieved by establishing a data base crganized by *modus operandi* and another data base organized by personal characteristics.

Seccndary indexing, however, enables separate indexes to be developed by DL/I for segments at any level in a logical structure of a data base. These indexes may be used tc directly retrieve segments at that level. Rapid access to specific infcrmation within a data base can be achieved, as well as the ability to consider a data base as being organized in different sequences. In addition, secondary indexing allows a data base structure to be inverted. Consider the criminal data base in Figure 5-35, with a seccndary index based on a field within the *modus operandi* segment. The structure may then be referenced in an inverted form, as shcwn in Figure 5-38.

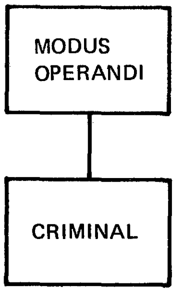


Figure 5-38. Inverted Police Data Base Logical Structure

If secondary indexing is nct to be used, an alternative approach may be adopted. By defining additional small data bases containing the relevant segment key infcrmaticn, and identifying the data base record key information as described above, a similar effect may be achieved, but with pcssibly more physical disk accesses and more user programming.

DATA BASE UTILITIES

A number of data base utilities are provided for DL/I. These facilitate the specification of data bases, and the recovery of the data bases in the event of failure.

Program Specification Block (PSB) Generation

The program specification block (PSB) macro instructions generate the control blocks that define to DL/I the physical or logical data bases used, the logical data structures within each physical or logical data base, and the operations allowed on each data base by a particular application. The sensitivity of an application program to particular information or segments within the data base is also defined in the PSB generation assembly.

Data Base Description (DBD) Generation

The data base description (DBD) macro instructions generate control blocks that define to DL/I for each data base, the data base name, its data structure, its data format, any logical relationships, any secondary indexes, and the DL/I access method used.

Application Control Blocks (ACB) Creation and Maintenance Utility

Before the program and data base descriptions created by the PSB and DBD generation utilities can be used with DL/I DOS/VS or IMS/VS DL/I, they must be merged and expanded to an internal format. Instruction execution and direct access I/O time, required to prepare the program for execution, are minimized by prebuilding the required application control blocks by means of the application control block creation and maintenance utility. Then, when an application program is to be run, its control blocks are read in directly (if they are not already in dynamic storage), and control is passed to the application program. With DL/I DOS/VS, the control blocks are brought into the CICS/DOS/VS address space at initialization of CICS/VS.

Data Base Reorganization Unload and Reload

When the data in the data base is updated, the physical structure of the data may change, increasing access time. Also, the space occupied by obsolete data is not in all cases reclaimed and reused. The data base reorganization unload and reload utilities may be used to unload, reorganize, then reload HISAM, HIDAM, and HDAM data bases to eliminate these problems. Additional information on DL/I data base reorganization and recovery may be found in the Utilities Reference Manual for the relevant DL/I product.

DATA BASE RECOVERY

The data base recovery system provided by DL/I DOS/VS and IMS/VS comprises four utility programs and is designed to provide a rapid, accurate, and easy-to-employ means of restoring the contents of the physical data base after destruction through an I/O error or abnormal task termination. The utilities are:

- Data base data set image copy

- Data base change accumulation

- Data base data set reccvery

- Data base backout

These utilities are introduced here, and discussed in more detail in Chapter 8.

DL/I ENTRY does not provide any support fcr data tase recovery.

DL/I LOG

All updates or additions to a data tase are autcmatically logged on a DL/I log tape in a batch envircnment, or optionally on the CICS/VS system log when CL/I is used on cnline with CICS/VS. These changes may be utilized, together with the data tase reccvery utilities described below, to restcre the data base to its data state at a particular point in time, in the event of the destruction of the data base through I/O errors, program errcrs, or system failure.

Data Base Backup

The data base data set image ccpy utility dumps individual data sets on tape or disk in a format suitable for use by the data base data set recovery utility. It is run periodically for data base backup.

Periodic Data Base Change Accumulaticn

The data base change accumulaticn utility sorts records from the data base log tape and combires all records that update the same segment. The result is a sequential data set that contains a condensed description of all changes tc the data base, and results in faster data base recovery.

Data Base I/O Error Recovery

The objective of the data base data set recovery utility is to recover from I/O error destructicn of the data base. This utility reconstructs individual data sets cf the data base by first obtaining from the data base backup run an image of each data set at a point in time when it was knewn to te valid, and then merging the accumulated application data from the most recent data base change accumulation run. Finally, any DL/I system lcg tapes which were not included in the accumulated change input are applied until the data set contains the desired data. The net effect is tc update the backup data base to its status at the time of failure caused by an I/O error.

Data Base Backout (Batch)

The data tase backout utility is designed tc recover frcm data base "pollutien" caused by abnormal prcgram terminaticn in a batch environment. It reads the DI/I system log tape tackward and removes (backs out) changes made to the data base frcm the pcint at which the CL/I system terminated, tc the pcint at which the program was scheduled. The resulting data base is ncw restcred to its data state at the time the application program was criginally scheduled, except for changes made by cther ncn tacked-out programs that terminated afterward. This utility also creates a log tape that reflects backcut changes.

Data Base Backout (Online)

In the online environment with CICS/VS, DL/I log activity is optionally directed to the CICS/VS system log. During emergency restart following an uncontrolled shutdown, the CICS/VS recovery utility program (DFHRUP) identifies in-flight tasks and collects in-flight tasks to the restart data set. (This identification may include in-flight DL/I activity.) CICS/VS backs out in-flight DL/I activity directly during emergency restart. The batch DL/I backout utility need not be used. See "Online DL/I Data Base Recovery" in Chapter 8 for additional information.

DL/I DATA BASE DESIGN

The information presented above has been given to enable the CICS/VS system designer to develop a conceptual understanding of the capabilities of the various DL/I products. Guidelines were given for the design of segments and logical structures, which make up part of the design of a data base. The following discussion identifies some additional factors which should be considered in data base design.

- Physical disk storage available

  Root-segment-only logical structures may enable an access method to be chosen which utilizes little or no prefix information, but contains only the data base record. However, a more complex logical structure, such as one involving logical relationships, contains considerable additional information in the prefix to define the pointers to various related segments.

  A further consideration is the reduction of redundant storage of information through the use of integrated data bases and logical relationships.

- Number of disk accesses

  The number of accesses which will be necessary to retrieve particular information is a factor to be considered in the design of data bases and of logical structures. These are influenced by factors which are the responsibility of the data base administrator, such as logical record lengths, block lengths, access method chosen, logical relationships, and secondary indexing.

- DL/I product to be used

  Another factor in data base design is the selection of the appropriate DL/I product. For example, in the case of DOS/VS installations, DL/I DOS/VS would normally be utilized with CICS/VS for installations having at least 160K of real storage or greater. For satisfactory performance, 192K of real storage, or greater, is recommended. support available with the particular product chosen may influence the data base design decision.

For further information on data base design, refer to the System/Application Design Guide for the relevant DL/I product.

DATA BASE SELECTION CRITERIA

Having broadly defined the data sets and programs required for the various online applications, a decision must be made as to the appropriate data base support for these applications. Various characteristics of the application and of the data sets required, as discussed at the beginning of this chapter, are used to make this decision. The support for CICS/VS applications should be selected from the following alternatives:

- CICS/VS file control support
- Data Language/I (DL/I) support

Several factors should be considered when making this decision. The most important of these factors are detailed in the following paragraphs.

- EXISTING RECORD FORMATS

  The particular record format for each data set should be considered. If the records are best suited to fixed format, any of the data base support products may be selected, depending on other criteria.

  However, if the data set can take advantage of variable record formats, the record contents should be examined in more detail. If the record is of variable-length because it contains a variable number of fixed-length sections of information (segments or logically related fields), then one of the DL/I products should be seriously considered.

  Each of the DL/I products supports the use of fixed-length segments. An effectively infinite number of segments can be present for each record, so resulting in a variable-length record.

  On the other hand, if there may be up to a certain maximum number of segments within each record, CICS/VS file control, using the segmented record feature, may be considered.

  If the variable record is made up of variable-length fields such as customer name and address, they may be supported by DL/I ENTRY or DL/I DOS/VS only if these variable-length fields are contained in fixed-length segments.

  If the record is variable because of the presence or absence of different sections or segments of the record, the record may lend itself either to CICS/VS file control segmented record support, or to DL/I support depending upon whether the segments are fixed-length or variable-length. If variable-length segments *must* be used, select CICS/VS file control, or if CICS/OS/VS is used, then IMS/VS DL/I may be selected.

- DATA BASE PERFORMANCE

  ### CICS/VS File Control Accessing

  Prime consideration in online application design should be given to the access time for retrieval of information from a data base. Depending upon the performance requirements of the application, this may dictate the selection of data base support. For example, if a data set may need to be accessed through other data sets, it may lend itself to the use of the CICS/VS file control indirect accessing feature. However, if several data sets have to be indirectly accessed to obtain the required information, these additional file accesses could have an adverse effect on online performance.

  The particular access method selected with CICS/VS file control for the application may affect the performance. For example, the direct access method (DAM) generally provides excellent online performance. However, DAM support requires that records be identified by either their physical location on disk or their relative location in a data set. The application, on the other hand, may require that a reccrd be accessed by a key. In this case, the indexed sequential access method (ISAM) may be suitable, but its use will involve at least two file accesses to retrieve each record. Furthermcre, if many additions are made to an ISAM data set, the access time for a specific reccrd may increase.

To overcome some of the above limitations, the virtual storage access method (VSAM) may be best suited. This enables records to be retrieved directly, based on relative location in the data set, or by key. It also enables rapid retrieval of information for applications with a high percentage of additions to the data set. However, VSAM should nct be used on CPUs with less than 144K of real storage, if satisfactory performance is to be achieved.

Another factor which should be considered is the serial scheduling of concurrently executing tasks, several of which may wish to update the same reccrd in a data set at the same time (see "Exclusive Control During Update" in this chapter). CICS/VS will permit only one task to update a reccrd at a time, and other tasks wishing to update that same record must wait for completion of the first update. (However, other reccrds in the data set may be concurrently updated, if required.) This serialization of updates may affect performance, if application factors may cause concurrent updating of individual data set records to be attempted.

DL/I Accessing

DL/I provides a number of access methods which may be used for satisfactory performance depending upon the requirements of the application. These access methods are the: Hierarchical Sequential Access Method (HSAM), Hierarchical Index Sequential Access Method (HISAM), Hierarchical Direct Access Method (HDAM), and Hierarchical Index Direct Access Method (HIDAM). Refer to the System/Application Design Guide for the relevant DL/I product for further information about DL/I access methcd selection.

The CICS/VS-DL/I interfaces all scheduled DL/I CALLs from CICS/VS application programs on a multithread basis. Several CICS/VS application programs (tasks) may concurrently access the same, or different, data bases up to a maximum of 255 concurrent tasks for DL/I DOS/VS, 32 for DL/I ENTRY, cr 15 for IMS/VS DL/I. CICS/VS tasks may concurrently access different segment types in a data base. However, if two or more tasks attempt to concurrently access the same segment type, DL/I DOS/VS and IMS/VS DL/I determine the type of accessing requested. If both tasks cnly wish to read the segment type, they are permitted concurrent access. However, if the tasks wish to update the segment type concurrently, for example, they are not scheduled concurrently by DL/I. The second task must wait until the first task has terminated its use of DL/I. This is called "Scheduling by Intent." It may affect performance if several tasks attempt concurrent update (for example) of specific segment types in the data base. (See "Updates, Additions, and Deletions" with DL/I in this chapter for an example of data base design to reduce the effect of segment intent scheduling in performance.)

The CICS/DOS/VS-DL/I ENTRY interface permits multithread access to DL/I data bases, up to 32 tasks for DL/I ENTRY. In order to prevent double updating of a segment, DL/I ENTRY uses CICS/VS facilities to enqueue (between the GET HOLD and the REPLACE calls) on the logical record that contains the segment to be replaced.

- EATCH PROGRAM ACCESS

  If the online application data sets require further processing in a batch environment, this consideration should also be taken into account in selection of the data base support.

  Factors which should be considered are that CICS/VS file control supports variable-length records within a fixed-length block for DOS/VS ISAM data sets, standard OS/VS variable-length BISAM data sets, and blocked records for DAM data sets. DOS/VS ISAM does not support these variable-length records for ISAM in a batch partition. They can be accessed in a batch environment by defining them to DOS/VS ISAM as fixed-length unblocked records. However, the batch processing program must itself deblock the variable-length records from the fixed-length block returned to it by DOS/VS ISAM.

  Neither DOS/VS DAM, OS/VS1, or OS/VS2 BDAM supports blocked records. If blocked DAM data sets are to be accessed in a batch environment sequentially, they may be defined as DOS/VS SAM data sets or OS/VS BSAM or QSAM data sets. In this instance, the sequential access method will handle deblocking of records.

  However, if the batch processing programs need to access these blocked records directly instead of sequentially, the responsibility rests with the batch program to define the data set as an unblocked DAM data set and provide its own deblocking of records within that physical block.

  The use online of the CICS/VS file control indirect access and segmented record features requires that special coding to support these features in batch programs be developed by the installation.

  The DL/I products support the same access methods and record formats both online and offline. No additional coding is required to enable batch DL/I programs to access online data bases.

  CICS/VS and concurrently executing DL/I batch programs in other partitions should not attempt to access the same data bases.

- EATCH DATA BASE CREATICN

  CICS/VS file control provides no facility for creation of the online data bases, apart from that provided by standard SAM, VSAM, DAM, and ISAM support. The insertion of indirect access pointers in data sets, and the preparation and organization of segmented records, is the responsibility of the user. Generally, special data base creation programs must be written by the user.

  Similarly, no facilities are provided for maintenance of the online file control data bases in a batch environmert. To provide this, the user's data base creation program should also be designed to allow a maintenance capability.

  DL/I allows creation and maintenance of data bases through the use of various utilities. Furthermore, because the program is independent of the physical data base organization and only refers to its logical organization, considerable flexibility is offered the installation in data base reorganization and maintenance.

- INSTALLATION DATA BASE SUPPORT DIRECTION

  In evaluating each of the selection criteria described above, the system designer must keep in mind the future direction for his installation in the use of particular data base support.

  CICS/VS file control may be utilized if desired, because CICS/VS has been identified by IBM as one of its standard data base/data communications program products. However, the CICS/VS installation may wish to take full advantage of the extensive data base support provided by DL/I, by using the appropriate CICS/VS-DL/I Interface feature.

CHAPTER 6. CICS/VS ADVANCED FEATURES

This chapter describes task control, interval control, and some of the facilities provided to CICS/VS application programs.

A number of facilities are provided to CICS/VS application programs for:

- Immediate creation of tasks
- Task priority change
- Enqueue/dequeue resource allocation
- Terminal read timout
- Isolated task paging
- Automatic task initiation at a future time
- Wait for completion of a time event
- Cancel a future time event

These facilities are provided by the task control program and the interval control program of CICS/VS. They are described in more detail in the CICS/VS Application Programmer's Reference Manual, SH20-9003.

TASK CONTROL

Task control allows CICS/VS application programs to attach new tasks for execution, if required. This may be done either by the automatic task initiation feature of transient data intrapartition queues, by time-ordered initiation (see "Interval Control," below), or by explicit use of the task control ATTACH macro instruction issued by an application program. The ATTACH macro instruction identifies the transaction code corresponding to the program to be executed. This program is initiated in exactly the same way as if a terminal transaction with that same transaction code had been received. The task attached competes for execution with other concurrently executing tasks based upon its task priority, which is determined from the transaction priority.

Priority Change

During the execution of a task, the task priority may be altered by use of the task control CHAP (change priority) macro instruction. The priority may be set to any priority value between 0 and 255. A normally low priority executing task may temporarily change itself to high priority (possibly during a section of logic which may involve updating of data sets) and then later change itself back to its previous low priority, if necessary, through use of the task control CHAP macro instruction. The use of the change priority facility enables data set updating (for example) to be completed in the shortest possible time. Alternatively, a task may execute initially in high priority, and then may be able to complete its execution in low priority.

The use of the task control CHAP macro instruction, without specifying a priority value, enables control to be released by a task to allow any high priority task which is ready for execution to gain control of the CPU. It is equivalent to issuing a task control WAIT macro instruction, and causes a task switch. This should be utilized if a task involves a considerable amount of CPU processing, to ensure that it does not monopolize the CPU and prevent other tasks from gaining control of the CPU.

Wait

The task control WAIT macro instruction functions in a similar fashion to the DOS/VS or OS/VS WAIT macro instruction, enabling a task to wait on completion of a single event or one event in a list of events. However, the WAIT may first result in task switching to another CICS/VS task which is able to process. Only if no CICS/VS task is able to process is control passed to another DOS/VS or OS/VS partition/region.

Terminal Read Timeout

This feature allows the user to specify a timeout limit for a conversational transaction when the transaction is waiting for a terminal input message. This keeps a single transaction from occupying system resources for long periods of time while waiting for a reply from the terminal.

Isolated Task Paging

This option allows the user to participate (with the operating systems page manager) in selecting pages to be made available for pageout. When the TCA storage of a private or long running task is acquired by task control, a number of additional pages may be acquired as specified by the ANTICPG operand in the task's PCT entry. Task control makes these pages available for pageout when the task waits for a response from a terminal, and causes them to be asynchronously paged in when the task is to be given control after the response has been received. This allows pages occupied by data areas belonging to conversational tasks to be paged out faster than the normal paging process.

Enqueue/Dequeue

Task control provides ENQ and DEQ macro instructions for other CICS/VS modules and application programs to enqueue and dequeue on various resources.

An enqueue and dequeue facility is often necessary in a multitasking environment, to ensure that only one task is able to utilize a particular resource at a time. In the case of the potential concurrent updating of records in a file control data set, file control utilizes the task control ENQ and DEQ macro instructions to ensure that the first task that retrieves a particular record for update is given the exclusive use of that physical record. A second or subsequent task which also wishes to update that same physical record while the first task is still in the process of updating it, is prevented from carrying out its update. When the first task has completed its update, and is dequeued from exclusive control of that record, the next task is given exclusive control of the record through the ENQ macro instruction. It carries out its update until it has completed, and then is dequeued from that record.

INTERVAL CONTROL

Future Task Initiation

Tasks can be initiated at a future time, based on either an elapsed period of time or a specific time of day. These future tasks can be initiated either with no data transfer through the use of the interval control INITIATE macro instruction, or with data transfer through the use of the interval control PUT macro instruction.

The INITIATE macro instruction enables a task to be attached at a future time. A unique time request identification can be allocated to that INITIATE macro instruction, to later identify that request if it is necessary to cancel the request before the task is initiated.

The use of the interval control PUT macro instruction also allows a task to be initiated at a future time, based on elapsed time or time of day, and specify data to be transferred to that task. Temporary storage is used to record the data to be passed to the future task. This time request is given a unique identification, and data which is to be transferred to the future task is written by interval control to temporary storage for each interval control PUT macro instruction which is executed.

When the future task is initiated, its data may be retrieved from temporary storage by the applicaticn program issuing an interval control GET macro instruction. Subsequent GET macro instructions will retrieve each reccrd which was initially PUT to temporary storage for the future task, until all expired records have been retrieved.

This facility is useful when designing online applications, to ensure that particular application events which must cccur at a specific time of day, or after a specific elapsed time following scme other application action, can be initiated automatically. However, procedures must be developed during online system design to cancel these future application events, if necessary for the application. Generally, canceling of future events wculd be placed under ccntrol cf the master terminal operator, through user-written programs which are given the same security code as that allocated to the master terminal operator.

Time Event Wait

Application programs may need to wait on completion of a specific period of time or until a specific time of day occurs. This can be achieved by the use of the interval control WAIT macro instruction, specifying the request identification of the original interval control macro instruction which specified the particular time request.

The use of an interval control WAIT macro instruction will utilize CICS/VS resources (such as dynamic storage for residence of the particular program and associated areas), until the WAIT is satisfied. Accordingly, the WAIT macro instruction should not be used unless the time duration is only a matter of seconds. If it is necessary for a longer duration wait to be in effect, this should be achieved by initiating a future task at some shcrt interval of time before the time event to be waited on expires. This future task may issue the interval control WAIT macro instructicn tc wait for completion of the specified time event, and so ensure that CICS/VS dynamic storage is tied up for the shortest pcssible time. To minimize the effect of utilizing these resources, that future task should occupy as little storage as possible.

Application programs may also request the time of day through the use of a CICS/VS interval control GETIME macro instruction.

Time Event Cancel

As indicated previously, it may be necessary for future time events to be canceled. This is best achieved by user-written application programs which issue the interval ccntrol CANCEL macro instruction, specifying the time event request identification fcr that event to be canceled. Ideally, these canceled transactions should be placed under control of the master terminal operator, by allocating a security code to the transacticn code so that it can be utilized only by the master terminal operator.

CHAPTER 7. CICS/VS PERFORMANCE CONSIDERATIONS

This chapter does not provide specific CICS/VS performance information, but instead discusses the various factors which should be considered to ensure adequate online performance for applications executed under control of CICS/VS. It describes various facilities provided by CICS/VS for the evaluation and improvement of online performance and details how these facilities can be used to vary the working set of CICS/VS, and hence its demands for real (main) storage. This chapter, particularly "Nucleus Load Table," is recommended reading in its entirety for all CICS/VS users.

DESIGN CRITERIA

APPLICATION DESIGN

Good application design will ensure that the amount of information to be transmitted between terminal operators and CICS/VS is kept to the minimum required by the application. In addition, good design ensures that only that information required for the application is retrieved from data sets, and possibly updated. The objective of Chapter 11, "Application Design," is to assist the CICS/VS system designer to develop application specifications which will result in satisfactory online performance.

MESSAGE DESIGN

The amount of information to be transmitted by the terminal operator to the CPU and the amount of information transmitted back from the CPU should be kept to a minimum. Most efficient utilization of dynamic storage for conversational tasks may be achieved by limiting, where possible, the use of the CICS/VS terminal control READ or GET macro instructions, or the BMS IN macro instruction, if it takes several seconds for a terminal operator to enter the relevant data. Instead, use should be made of temporary transaction codes (see Chapter 3), or entry of transaction codes by the terminal operator. This will permit the dynamic storage occupied by the program waiting for terminal input to be utilized for other purposes, if the transaction load requires it. When the terminal input is received, the program required to process it is then initiated as for any other CICS/VS task.

Use the versatility offered by the various programmable controllers to perform data editing and message formatting prior to transmission to CICS/VS.

PROGRAM DESIGN

CICS/VS application programs may be modular, and may contain only that code necessary to process a particular transaction. Code not required in every case for the processing of a transaction, such as exception routines or error routines, may be incorporated in separate modules.

However, the amcunt cf prcgram modularity should not be extended so far that it requires too many application programs tc be loaded to process the information in a particular transaction. The need to transfer control (XCTL) or link (LINK) to many application programs in processing each transacticn will add extra processing overhead to the potential response time, because of the need to search larger tables, dynamically allocate more register save areas for LINK macro instructions, or dynamically load application prcgrams frcm the CICS/VS program library if they are nct presently resident. It may also increase virtual storage paging activity (see "Virtual Storage Considerations" later in this chapter).

DATA BASE DESIGN

One of the mcst important factors in the design of online application programs is the data base design. The factors identified in Chapter 5 should be considered when determining the apprcpriate data base support. The particular suppcrt selected is a function of the application requirements, amcunt cf logically related information, degree of data independence and reduction in data redundancy, the number of file accesses necessary tc retrieve and possitly update the required information, and the amount cf main storage in the system to support the CICS/VS applications.

The mcre file accesses that are necessary to retrieve information the longer is the potential respcnse time. The number of file accesses necessary in following a logical chain of information across data sets using the indirect access feature cf CICS/VS file control may be quite extensive in many applications. The utilization of a segmented data base support, such as that provided by the CICS/VS file ccntrol segmented record feature, and by the DL/I products, may be instrumental in reducing the amcunt of file accessing necessary. However, the use of complex logical relationships with DL/I may involve additional file accesses.

DYNAMIC STORAGE

One of the most significant resources in ensuring adequate online performance and respcnse time is the availability of dynamic storage for program execution.

A formula is provided in the CICS/VS General Information Manual (GIM) to enable dynamic stcrage to be estimated. The amcunt of dynamic storage required is a functicn of:

- The maximum number of tasks that are active at any instant in time
- The amount of information to be transmitted between the CPU and the terminals
- The storage required by application programs needed to process the transaction informaticn
- The additional storage required for file input/output areas, file work areas, temporary stcrage requirements, and working storage areas

The amount of dynamic storage allocated should be consistent with the degree of multitasking tc be supported (see "Multitasking" in this chapter). It shculd alsc be large enough to prevent any CICS/VS short on storage (SOS) situations from occurring. Such a situation occurs when the storage cushion has been released to satisfy storage requests, and subsequent storage requests exceed the remaining storage in the cushion. In this instance CICS/VS will temporarily stop inviting terminals to send input messages until sufficient storage has been released by active tasks to reestablish the storage cushion. Terminals will then be invited to send input once more. The effect of the SOS condition then is to temporarily degrade the acceptance of input until the condition disappears. If insufficient dynamic storage is allocated, and SOS occurs, performance may degrade.

It was suggested with previous versions of CICS/VS that SOS be permitted to occur occasionally, by allocating only sufficient dynamic storage to satisfy average transaction loads. At peak transaction loads, SOS could be useful to control the arrival of transactions for processing by the CPU. This technique is not recommended for use with CICS/VS, since an SOS condition can result in virtual storage performance degradation. It can be avoided by allocating a sufficiently large virtual storage partition/region to result in a large dynamic storage area, or by reducing the maximum tasks value used by CICS/VS (see "Multitasking" later in this chapter). The CICS/VS statistics indicate when SOS occurs.

The dynamic storage area should also be large enough to avoid excessive application program loading from the CICS/VS program library, but not so large as to result in the generation of a large number of page faults in a virtual storage environment The amount of dynamic storage allocated, and hence the virtual partition size, is chosen based upon:

- The storage required, using the formula from the CICS/VS GIM

- The amount of multitasking to be used

- The amount of contention for real storage by CICS/VS and other partitions

The optimum CICS/VS virtual partition size is dependent upon the paging characteristics and storage reference patterns of CICS/VS and its application programs, and also the type of concurrently executing batch application programs. While the CICS/VS virtual partition and dynamic storage size can be estimated as described above, the contention between CICS/VS and batch programs for real resources (such as main storage, CPU computing capability, channels, and disk file arms) must be determined. In many cases, benchmarking the particular CICS/VS online application with specific batch programs may be appropriate.

Previous versions of CICS recommended that transaction processing storage be requested in small amounts, be acquired only when needed, and be released as soon as possible. With CICS/VS, several separate requests for storage should be consolidated, so that a larger area is acquired to satisfy the individual requirements from that area. This has the advantage of localizing storage references, and so reduce the possibility of page faults.

Alternatively, storage should be preallocated in the TWA when the TCA for each task is created by CICS/VS. The TWA for each transaction code may be specified by the user when the PCT is generated.

MULTITASKING

The ability of CICS/VS to permit up to 999 tasks to be processed concurrently can enable efficient resource utilization to be achieved. As soon as a transaction reaches the CPU from a terminal, CICS/VS endeavors to create a task to commence processing that transaction as soon as possible. When the task has to wait on the completion of an event, such as an I/O access, another task which is able to continue processing is given control of the system. Although no provision is made by CICS/VS to enable a task to continue processing while an I/O operation is in process for that same task (except for VTAM terminal I/O operations specified for immediate execution and asynchronous journaling activity - see "Journaling" in Chapter 8), the multitasking capability enables other tasks to execute while that task is waiting on the completion of an event.

Multitasking permits many transactions to be processed concurrently and efficiently, with the result that a much smaller range of response times is achieved than if one transaction were processed at a time, that is, in single-thread processing. With single-thread processing, if three transactions reached the CPU at the same time and each required one second for processing, the transactions would be processed one at a time. Thus the first transaction would have a response time of one second, the second transaction would have a response time of two seconds, and the third transaction would have a response time of three seconds. If, however, 50 transactions reached the CPU at the same time, with this example the spread of response times would range between one second and 50 seconds.

Even though the time for processing of each transaction may be slightly increased over that achieved with single-thread processing, the net effect is a reduction in the range of response time. This is very significant to the overall performance of a heavily loaded system.

The degree of multitasking (the "maximum tasks" value) to be carried out by CICS/VS can be specified by the user, either at CICS/VS initialization time, or dynamically during online execution by the master terminal operator. (See the CICS/VS System Administrator's Guide, SH20-9006.) While a large maximum tasks (MAXTASKS) value can reduce the range of response times, an unnecessarily large value can increase the number of virtual storage pages utilized by CICS/VS. This can affect the amount of paging activity carried out by the CPU, and is discussed further in "CICS/VS Working Set" later in this chapter.

PRIORITY PROCESSING

A further factor which affects the processing time and response time of individual transactions is the degree to which the priority processing facilities of CICS/VS are utilized. In an environment in which the transaction load is such that there may be several transactions concurrently in process, certain transactions or tasks may be given a higher priority for execution than other transactions or tasks. Thus, in most instances in which a high priority and a low priority task are able to continue processing, the high priority task is given control of the CPU over the low priority task.

Task priority can be set as described in Chapter 4, by the sum of the allocated terminal priority, operator priority, and transaction priority. The terminal and/or transaction priorities may be dynamically modified during CICS/VS operation by the master terminal operator (as described later under "System Control Changes") to reflect the changing significance during operation of various application transactions or terminals.

The various factors which influence the performance of online applications under CICS/VS have been discussed above. The determination of the actual performance achieved in an operating environment, and the variation of the CICS/VS system to improve performance, is the function of the system administrator.

SYSTEM ADMINSTRATION FUNCTIONS

The system administration, monitoring, and control of the online system are generally functions of the CICS/VS master terminal operator in the installation. An installation may allocate a terminal, ideally a hard-copy printer, to the master terminal operator such that automatic messages directed to his attention by CICS/VS may be immediately printed. While the master terminal operator may sign on at any CICS/VS terminal, it is usually desirable for the installation to provide a specific master terminal. This may be a hard-copy terminal such as a 2740 Communications Terminal or a video terminal such as a 3270 terminal, with either a 3277 or a 3275 display station and a 3284 or 3286 printer for master terminal messages.

A number of monitoring and control functions are available to the master terminal operator. These will now be discussed.

Operating Statistics

The master terminal operator can request particular CICS/VS operating statistics or all CICS/VS operating statistics. (See the CICS/VS System Administrator's Guide, SH20-9006.) These requests can be made at any time. CICS/VS will output the various statistics which have been accumulated since the last time they were requested.

Optionally, the CICS/VS automatic statistics feature can be used to get statistics on a regular interval basis and have them printed in interval and/or summary format.

Some of the statistical information maintained includes:

- Maximum number of tasks for any time period
- Number of times at the maximum tasks value
- Number of tasks initiated
- Maximum number of active tasks in system for any time period
- Total number of ATP transactions
- Total number of ATP batches
- Number of storage acquisitions
- Number of times storage cushion is released
- Number of times storage request is queued
- Number of times storage queues were established
- Maximum number of requests in the storage queue
- Number of times each transaction was used
- Number of times each transaction was stall purged
- Number of times each transaction required storage in addition to the specified anticipatory paging amount
- Time (in seconds) that elapsed while the first search of the PCT was made for each transaction used

- Number of times a program is used
- Time (in seconds) that elapsed while the first search of the PPT was made for each program used
- Total number of storage dumps
- Total number of storage dump write errors
- Number of polls issued per line
- Number of input messages per terminal
- Number of output messages per terminal
- Number of transmission errors per terminal
- Number of transactions per terminal
- Number of transaction errors per terminal
- Number of pipeline messages discarded
- Number of groups of pipeline messages discarded
- Maximum length of a group of discarded pipeline messages
- Maximum number of VTAM RPLs posted in RPL pool
- Total number of times that maximum number of RPLs was reached
- Number of times VTAM was short on storage
- Number of READ requests per data set
- Number of WRITE update requests per data set
- Number of WRITE adds per data set
- Number of READs from overflow area per data set (ISAM only)
- Number of times each individual segment in a segmented record data set is operated upon
- Total number of tasks required to wait for a VSAM string per data set
- Highest number of tasks required to wait for a VSAM string per data set
- Total number of DL/I requests of each type (GU, GN, GNP, GHU, GHN, GHNP, ISRT, DLET, and REPL)
- Number of WRITES or READs (per data set) to extrapartition data sets
- Number of WRITEs or READs (per data set) to intrapartition data sets
- Number of temporary storage records PUT/PUTQ to main storage
- Number of temporary storage PUTs to unique IDs
- Maximum virtual storage used for temporary storage records

- For each journal:

  a. Number of records written

  b. Number of blocks written

  c. Number of times the buffer was full

  d. Number of times the block was shifted up

  e. Average output block size

This statistical information is necessary not only to indicate the actual performance of the CICS/VS system during operation, but also for management's planning for future system growth. The statistics are useful to:

- Help the system programmer determine that efficient data set allocation has been made

- Aid the system programmer in choosing programs to be made permanently resident during system initialization processing, as opposed to those programs CICS/VS is to load dynamically

- Determine the activity of terminals and transactions

- Reorder the sequence of entries in various CICS/VS tables, to ensure that the most active entries are toward the top of tables and thus minimize table scanning

- In general, determine if the resources of the system are being effectively used

Statistics concerning DL/I DOS/VS data base accesses are collected by DL/I rather than by CICS/VS. In addition, DL/I DOS/VS provides utilities which may be used to produce reports of these statistics.

IMS/VS DL/I also collects statistics. In addition, the CICS/OS/VS-DL/I interface maintains statistics in the file control table for the following DL/I CALL functions: GU, GN, GNP, GHU, GHN, GHNP, ISRT, REPL, and DLET.

Performance Monitoring

The statistics described above are particularly useful for the system administrator or master terminal operator, to monitor the overall performance of CICS/VS, and also to vary the working set of CICS/VS (see "CICS/VS Working Set" later in this chapter.)

If the statistics indicate that the maximum number of tasks in the system for any time period approached the maximum tasks value allocated, some improvement may be realized by increasing that maximum tasks value. However, if the maximum number of tasks in the system for any time period is significantly less than the maximum tasks value, that value should be decreased to the maximum number of tasks indicated by the CICS/VS statistics. This may result in improved virtual storage operation. (The maximum tasks value can be varied dynamically by the master terminal operator, as described in "System Control Changes" later in this chapter).

If the storage cushion has not been released and a storage queue has not been established, there is sufficient dynamic storage available to maintain a higher degree of multitasking, if required. On the other hand, if the storage cushion has been released frequently and storage queues have also been established frequently, perhaps approaching the total number of storage acquisitions, insufficient dynamic storage has been allocated to enable effective operation of the online applications.

Statistics on the number of transmission errors per terminal may indicate a possible failing line or terminal which may require maintenance. Similarly, a large number of transaction errors per operator or terminal may indicate inadequate operator training.

CICS/VS statistics may be reset by the master terminal operator as desired. The master terminal operator may also specify that they are to be written out periodically and then reset. For example, it may be desired to output the statistics every 15 minutes. These statistics are directed to the transient data destination CSSO, which may be defined or as an extrapartition destination such as a tape, or disk. A CICS/VS-supplied edit program can be used to produce a formatted report of the statistics to illustrate changes in the system environment by time.

The CICS/VS Performance Analyzer Field Developed Program (Program No. 5798-AZN) may be used to measure the performance of tasks executed in the online system. Refer to "Related Publications" in the Preface for relevant publications.

Class Max Task

This feature allows the user to tune the CICS/VS system by controlling the mix of transactions within CICS/VS. Without this feature, all transactions initiated within CICS/VS are considered equal, and one transaction type can dominate the system. (For example, a broadcast to all terminals would flood CICS/VS with transactions until the broadcast was complete and prevent terminal operators from performing any useful work.)

The class max task feature also allows resource balancing within CICS/VS by limiting the number of transactions allowed to run concurrently. CICS/VS supports ten transaction classes (1 through 10). The function of the classes is at the discretion of the user. An example of the use of the transaction classes is:

Class 1 - Inquiry only transactions.

Class 2 - Update or add transactions.

Class 3 - File browse or weighted retrieval transactions.

Class 4 - Auto-initiated tasks (such as ICP or TDP initiated). Classes 5 through 10 - Could be used to group transactions with similar working set sizes by working set size. This feature, in conjunction with Isolated Task Asynchronous Paging, can be used to control paging load in CICS/VS through use of a data area swapping technique.

Max Active Tasks

This feature allows the user to tune the system by controlling the paging rate within CICS/VS. It provides a tuning tool for CICS/VS conversational systems where max task cannot be less than the number of terminals in the system.

The user can specify a max active task value to CICS/VS in the system initialization table, or as an override at start-up time. The master terminal program can also change the max active task value. (See the System Programmers Reference Manual, Order Number SH20-9004 for additional information.)

System Control Changes

Based upon the CICS/VS operating statistics provided to the master terminal operator, either on request or at system termination, he may monitor the performance of the online system. He may wish to make changes in various significant areas of the CICS/VS online system to attempt to improve its performance. The master terminal operator can also dynamically vary system parameters, as well as change the status of lines and/or terminals and/or data sets. All processing for master terminal operation is controlled by conversational terminal interaction, using the master terminal transaction CSMT. (See the CICS/VS System Administrator's Guide, Order No. SH20-9006.)

Some of the system control functions which the master terminal operator is able to carry out include:

- Altering transaction and/or terminal priority
- Disabling and enabling various table entries (PPT, PCT, FCT, DCT)
- Placing terminals in-service or out-of-service
- Dynamically listing active tasks
- Purging active and suspended tasks
- Dynamically opening or closing selected data sets
- Switching to an alternate dump data set
- Increasing or reducing the maximum number of tasks which will be processed concurrently by CICS/VS
- Changing the storage cushion size
- Acquire and release VTAM terminals to establish CICS/VS-terminal sessions

Of the functions detailed above, the modification of the maximum tasks value and of transaction and terminal priorities may have most effect on overall online system performance.

In addition, the ability to dynamically list active tasks may enabl certain long executing tasks which are tying up system resources and dynamic storage to be identified, and if required, to be purged (abnormally terminated) under control of the master terminal operator. This will enable the resources being utilized by those long-running tasks to be freed for use by other tasks.

CICS/DOS/VS SUBSET OPTION

The subset option of CICS/DOS/VS does not support the following CICS/DOS/VS facilities.

- Program Control - Dynamic program loading
- Interval Control - Except for system stall detection and runaway task control

- Terminal Control - Terminals other than 3270, 2740, 2741, sequential terminals, and CPU console

- File Control - Automatic logging

- Transient Data - Automatic logging

- Temporary Storage - Auxiliary storage residence

- Journal Control

- Sync Point Management

- Supervisory Terminals

- Asynchronous Transaction Processing

- BMS Terminal Paging

- BMS Message Routing/Switching

- System Recovery - Partition abnormal termination interception

- Emergency Restart

- System Log/Journal Utilities

- 2260 Compatibility

- Built-in Functions

Removal of this support reduces the working set of CICS/DOS/VS substantially. Refer to the Subset User's Guide (DOS), Order No. SH12-5404 for further details.

VIRTUAL STORAGE CONSIDERATIONS

CICS/VS executes in a virtual storage environment under control of DOS/VS, OS/VS1, or OS/VS2. A number of factors must be considered in the virtual storage environment to ensure that optimum online performance is achieved, consistent with satisfactory batch program performance.

Page Pool

With the availability of virtual storage on System/370 computers, the function of main storage (that is, real storage) in a computer has been changed from an essential resource, necessary to enable various programs to execute, to a performance resource.

The majority of application programs can execute in a virtual storage environment regardless of the size of the program or the amount of real storage available. However, the more real storage available, the less the amount of paging which must be carried out by the virtual storage supervisor to enable that program to execute. Because each page fault represents I/O and CPU execution overheads, the real storage available for execution of virtual storage programs should be sufficiently large as to minimize the number of page faults necessary.

The page pool is the total real storage available to all partitions or regions, after operating system requirements are met. It is difficult to change the size of the page pool without adding more real storage. In many cases, the adequacy of the page pool cannot be effectively determined without benchmarking the batch programs and CICS/VS together and evaluating performance (as measured by response time for CICS/VS, and as measured by throughput for the batch programs) as the demands on the page pool vary.

The size of the page pool necessary for satisfactory performance of the CICS/VS applications together with concurrently executing batch programs is very dependent on the "working sets" of all the programs operating concurrently. The working set of a program is the number of page frames (the amount of real storage) required by that program at any given point in time. Performance degrades or improves in direct relation to the amount of contention for the page pool caused by the working sets of all the concurrently operating programs.

To determine the factors affecting total performance, CICS/VS and its working set on the one hand should be balanced with the rest of the system and its working sets on the other. As the working set of CICS/VS is increased, the available real storage for the rest of the system is decreased, possibly degrading the performance of other operating programs. The converse is also true. It is the user's responsibility to determine the number of partitions or regions to be used and the best mix of batch programs to operate concurrently with CICS/VS.

Those factors which affect the working set and performance of CICS/VS will now be examined.

- VSAM Shared Resources (CICS/OS/VS Only)

VSAM shared resources enable the real storage requirements for VSAM control block and buffer resources to be reduced. This results in a reduction of the working set of CICS/VS. Shared resources permit efficient utilization of storage in an environment in which many VSAM data sets are open and it is difficult to predict the amount of activity against a given data set, or in a situation where each transaction may access several VSAM data sets. (See Chapter 5 for additional information concerning VSAM shared resources.)

CICS/VS provides statistics to evaluate the efficiency of resource sharing. If too few VSAM buffers or strings are available, tasks may wait an excessive amount of time. If too many are available, real storage may be being used inefficiently. The statistics are calculated by CICS/VS to provide the following information:

- Maximum key length
- Number of strings
- Size of buffers
- Number of buffers of each size

The effectiveness of these resources can be evaluated using the following statistics:

- The highest number and total number of requests that had to wait for the availability of a buffer according to control interval sizes. This information may be used to change the percentage of the maximum amount of resources required, or to specify the sizes of buffers and number of buffers for each size to be allocated.
- The highest number and total number of requests that had to wait for a string. This information may be used to change the percentage

of the maximum amount of resources required, or to specify the number of strings to be allocated.

- The maximum number of strings active concurrently. This may be used to change the percentage of the maximum amount of resources required, or to specify the number of strings to be allocated.

- The number of successful read requests that were satisfied by data that was already in the buffer.

- The number of buffer reads.

- The number of buffer writes.

The last three statistics may be used to change the percentage of the maximum amount of resources required, or to specify the sizes and number of buffers for each size to be allocated.

CICS/VS Working Set

- Activity

The message rate, while not directly under the control of the system programmer, has the effect of increasing or decreasing the CICS/VS demands on the system. The batch processing load in other partitions may need to be varied when the message rate increases at times of peak online activity.

Through the use of the MAXTASKS and/or AMXT parameters, the master terminal operator can directly affect the amount of activity that CICS/VS will attempt concurrently. Each initiated task increases the CICS/VS working set by the demands of the task for dynamic storage and application programs. The MAXTASKS and AMXT parameters are tuning aids for the system.

As discussed previously in this chapter, an arbitrarily large value which significantly exceeds the maximum number of tasks active in the system for a given period of time (see "Performance Monitoring") can result in unnecessarily high demands on real storage. This may result in performance degradation, if an inadequate page pool is available to satisfy these real storage demands. The user should vary the MAXTASKS and AMXT values until he determines the best value for his installation's performance requirements and real storage availability.

At periods of low online activity, a method of controlling the frequency of use of CICS/VS is by the use of the system partition/region exit time interval (ICV). A low ICV value causes control to return to CICS/VS quickly with a strong possibility of the more frequently utilized CICS/VS management routines remaining resident in real storage, and thus also a strong possibility of improved response time. The ICV value can be varied by the master terminal operator to meet high and low CICS/VS demands at times of differing online activity.

- Short-Term and Long-Term Transactions

CICS/VS enables transactions in the program control table (PCT) to be defined as being either short term or long term in execution.

Short-term execution transactions are allocated dynamic storage from common pages. A page is utilized to satisfy the storage requests of concurrently executing short-term transactions. Pages are allocated for concurrent usage by short-term transactions as required. When all the storage allocated in a short-term transaction page has been freed, it is returned to a common CICS/VS pool of pages for use in satisfying other dynamic storage requests. The storage required by a short-term transaction is intermixed in pages with the storage required by other short-term transactions.

On the other hand, all the storage required for execution of each long-term transaction is included in one page (or more, if necessary), and each long-term transaction storage is kept separate from other long-term transaction storage in different pages. If there are long periods of inactivity for these long-term transactions, the pages in which storage has been allocated for them may be paged out, if necessary, to enable those page frames to be utilized for other purposes.

Identification of short-term and long-term transactions is made by the user in the program control table. If not specified for a transaction, CICS/VS will default to a long-term specification for that transaction. A large number of long-term transactions may result in inefficient utilization of virtual storage pages, if those transactions are in fact short-term. This will increase the CICS/VS working set, and result in higher demands for real storage than would otherwise be necessary, with possible performance degradation. The CICS/VS General Information Manual, GH20-1280, and CICS/VS System Programmer's Reference Manual, SH20-9004, describe the allocation of dynamic storage by CICS/VS to various subpools, based upon the expected use of storage by the application programs.

If insufficient dynamic storage is allocated, together with a large maximum tasks (MAXTASKS) and maximum active task (AMXT) values, CICS/VS will, in allocating storage for short-term transactions, encounter short-on-storage (SOS) conditions. Accordingly, MAXTASKS and AMXT values should be utilized consistent with the allocated dynamic storage area, as discussed previously in "Dynamic Storage" in this chapter. Too large a value may result in a large number of SOS situations.

- Nucleus Load Table

The nucleus load table (NLT) is used by CICS/VS to control the load order of the CICS/VS nucleus during system initialization. This enables the user to vary the nucleus load sequence to reflect his installation's use of various CICS/VS management modules, and to reduce the amount of real storage required by CICS/VS. The purpose of the NLT is to allow tailoring of the nucleus loading to provide the user with the smallest possible working set for his particular environment.

The user should first identify those CICS/VS management modules most frequently used by his application programs, and those less frequently used, or not used at all. For example, the TCT, CSA, KCP, ICP, TCP, and SCP are used frequently in polling terminals and servicing terminal input. The PCT, PPT, and PCP are used when initiating task processing. During execution, tasks may initiate activity in the FCP, FCT, JCP, and JCT. Other management modules such as TSP, TDP, DCT, and BMS may be referenced during execution. However, modules such as DCP, SRP, SCR, and the SRT are only referenced to handle abnormal conditions, CICS/VS dumps, or to intercept abnormal conditions and attempt recovery. (See the CICS/VS System Programmer's Reference Manual, SH20-9004, for NLT examples.) Use of the NLT enables these modules to be sequenced with all frequently referenced programs packaged together to reduce the number of pages required, and may be page fixed in real storage if desired.

- Application Load Table

The application load table (ALT) is used to control the load order of application programs. The application load table is an optional feature in CICS/VS. If it is not used, application programs will be loaded based on parameters in the processing program table. If the application load table is used, the application programs specified in the application load table will be loaded first. Any programs in the PPT not specified in the ALT will be loaded with the options specified in the program processing table. Options such as page fix, page align, page out, and so on, are available in the ALT.

- Page Fixing

The NLT enables CICS/VS modules to be page fixed. By fixing these CICS/VS management modules in real storage, their availability can be assured when needed to ensure their responsiveness to requests for service from application programs.

In addition to fixing some CICS/VS nucleus modules, various application programs which will be used frequently, or for which optimum response is required, may be defined as being resident and may also be fixed in real storage. This is specified in the program processing table. These fixed application programs are never paged out of real storage.

Page fixing of parts of CICS/VS and of various application programs may degrade batch processing program throughput and possible CICS/VS reponse times. These fixed page frames are available only to the CICS/VS partition and, in effect, increase its working set. The net effect is the reduction of the number of remaining page frames for other partitions. When considerable contention exists for real storage, this reduction of available page frames for use by other partitions may significantly degrade the performance of programs operating in those partitions and result in deactivation of low priority partitions (see "Batch and CICS/VS Page Contention" later in this chapter).

- Anticipatory Paging and T/P Balancing (CICS/DOS/VS only)

In a low volume online environment, batch program contention may force most of CICS/VS to be paged out. To permit these pages to be rapidly paged in again when needed, CICS/VS supports anticipatory paging of CICS/VS nucleus modules.

The CICS/VS user may specify (using the NLT) various frequently used modules which are to be paged in when the CICS/VS partition is given control by DOS/VS. These pages may have been paged out by DOS/VS to satisfy the paging requirements of concurrently executing batch programs in a low volume online environment. CICS/VS constructs a page-in list and, as soon as it is given control, requests DOS/VS to page-in the user-specified modules if they are not already in real storage. This is called "anticipatory block paging," and is supported by CICS/DOS/VS V1.0.1 or higher. It is not supported by CICS/OS/VS.

All pages referenced in the page-in list are migrated by the T/P Balancing feature of the DOS/VS supervisor to "most recently used" status in its paging algorithm. Batch pages may have been paged out to enable the referenced CICS/VS pages to be paged in. Batch partition execution is permitted to continue if required. At this time, CICS/VS may specify, using the NLT, certain infrequently referenced modules which are to be paged-out.

CICS/VS modules can be paged out at times of no activity until a terminal transaction is received for processing. DOS/VS then pages in blocks of pages identified by CICS/VS with minimum I/O activity. Batch processing may be deactivated to ensure that CICS/VS has exclusive use of all of the resources it needs. The result is consistent online response time; the trade-off is possible reduced batch throughput.

Anticipatory block paging and T/P balancing of CICS/VS nucleus modules should not be specified when CICS/VS is continually active, as in a high volume online environment, because of the additional processing overhead of anticipatory paging. In this case, the NLT can be used to minimize the CICS/VS working set. However, it should not specify page-in or page-out for any modules. Normal CICS/VS and batch program execution contend for the available page pool as in a normal virtual storage environment.

If CICS/VS is executing in a dedicated environment without concurrent batch execution, no benefit is gained by anticipatory block paging or T/P balancing; it should not be specified in the NLT. Refer to the CICS/VS System Programmer's Reference Manual for additional information on anticipatory paging and T/P balancing using the nucleus load table.

- Nucleus Load Table Design

A default NLT is supplied by CICS/VS. This is documented in the CICS/VS System Programmer's Reference Manual, SH20-9004, together with several examples of typical nucleus load tables. If the system programmer wishes to design an NLT for his installation's environment, the following factors should be considered:

- The use of BTAM and/or VTAM modules, and reference characteristics reflected by terminal communication activity
- Application program characteristics as reflected by use of CICS/VS nucleus modules
- Individual nucleus module sizes
- Module grouping to minimize page faults

Use may be made of CICS PLOT, a Field Developed Program (Program No. 5798-CCG), to determine the virtual and real storage usage of CICS/VS and identify possible nucleus load table structures.

- Resident CICS/VS Application Programs

Application programs may be specified in the PPT as resident in the nucleus or dynamically loaded (as required) in the dynamic storage area. Resident programs are loaded with the CICS/VS nucleus at initialization time, and are packed to occupy the minimum number of pages. The PPT also provides facilities to define page alignment and force pageout requirements for resident programs. These options can be used to improve the utilization of system address space.

When a program is selected for force pageout, CICS/VS issues a force pageout command to the operating system when the program has terminated processing and is no longer being used by the application program. The force pageout command will include the complete page or pages occupied (or partially occupied) by the program. The program will be paged out by the operating system upon first occurrence of a page fault.

Dynamically loaded programs occupy unique pages in the dynamic storage area; a program occupies contiguous pages. Each program is loaded on a page boundary and space at the end of a page is not used. Consequently, dynamically loaded programs may increase the working set of CICS/VS.

CICS/DOS/VS and its subset option permit execution using a DOS/VS supervisor which specifies no asynchronous processing support (AP=NO). See CICS/DOS/VS Operations Guide, SH20-9012. This reduces the size of the DOS/VS supervisor by approximately 5K. The 5K becomes available as extra storage for the page pool. However, if AP=NO is specified, CICS/VS journal control facilities and the CICS/DOS/VS-DL/I DOS/VS interface cannot be used.

- CICS/VS Generation Options

As more CICS/VS generation options are selected, the size and number of the management modules increase. As mentioned earlier, some of these modules may be page fixed by the user, while others are always pageable. The CICS/VS working set will generally increase as more generation options are selected. The user should therefore give careful thought to planning his CICS/VS system generation.

The subset option (CICS/DOS/VS) defines an environment which specifies only selected generation options for ease of use and installation and for a reduced working set. Refer to "CICS/DOS/VS Subset Option" in this chapter for the CICS/VS facilities not supported.

- Network Control System (NCS)

The user may wish to examine the relevance of NCS to his environment, and its effect on the real storage requirements for his online applications. Refer to "Related Publications" in the Preface for relevant NCS publications.

- General Considerations

The size of the CICS/VS dynamic storage area is an important factor in considerations involving the CICS/VS working set, as identified previously. The dynamic storage area increases as the CICS/VS virtual partition size allocated by the user is increased, and the converse is also true. Too small a dynamic storage area can cause frequent CICS/VS program loading and use of the storage cushion, which may slow terminal response due to a short-on-storage (SOS) condition. A large maximum tasks value with a small dynamic storage area will also result in an SOS condition, with subsequent performance degradation.

On the other hand, in the same online environment, too large a dynamic storage area where there is considerable contention with other partitions for the available page pool may increase the amount of virtual storage paging necessary. The most suitable dynamic storage area for the user's environment can best be determined by varying the CICS/VS virtual partition size and maximum tasks value in separate runs, and evaluating their effect on both CICS/VS performance and the performance of programs executing in other partitions.

Program residency in a virtual storage environment has a different meaning than in a nonvirtual storage environment. Page fixing in CICS/VS is equivalent to program residency in previous versions of CICS. A resident CICS/VS program is loaded when the CICS/VS system is initialized, but can be paged in and out as needed. The user should evaluate, in his own environment, the effect of specifying various application programs as resident, and also as page fixed, on CICS/VS performance as well as on the performance of programs executing in other partitions.

The number and size of I/O areas used by application programs for file and terminal I/O also have a direct effect on the CICS/VS working set. File I/O areas will be short-term page fixed for the duration of the I/O operation. Terminal and line I/O areas, on the other hand, are long-term page fixed while terminals are being invited to send input. In addition, all of BTAM and the TCT are page fixed at CICS/VS initialization. The number and variety of different terminal types supported by BTAM and VTAM in the user's environment increase their real storage requirement, and hence the working set of CICS/VS.

CICS Dynamic Map (Program No. 5798-AXR), and CICS PLOT (Program No. 5798-CCG) Field Developed Programs may be used to analyze virtual and real storage usage of the CICS/VS partition during execution. Refer to "Related Publications" in the Preface for relevant publications.

Batch and CICS/VS Page Contention

As discussed previously, the most significant impact on online and batch program performance, executing in a virtual storage environment, occurs when there is considerable contention between the various concurrently executing programs for available page frames in the page pool. The extent to which the total pages required by all concurrently operating programs exceeds the number of page frames in the page pool is called the degree of "overcommitment." The degree of overcommitment will determine the amount of performance degradation for both online and batch programs. If the overcommitment of the page pool is too great, such that the amount of paging necessary to enable various partitions to execute exceeds certain limits (depending upon the virtual storage operating system used), the particular operating system will attempt to deactivate the lowest priority partition. This situation is called "thrashing." The page frames occupied by this deactivated partition are made available for programs executing in higher priority partitions. When the paging rate reaches an acceptable level again, the lowest priority partition is reactivated.

Thrashing is a situation in which the demands made by various partitions for real storage result in a high proportion of necessary paging activity to enable the system to complete useful work. It is temporarily alleviated by the operating system deactivating a low priority partition to reduce the demand for real storage. Depending upon the degree of overcommitment, reactivation of the low priority partition by the operating system causes thrashing and subsequent deactivation.

In this environment, the only permanent solutions to avoid thrashing are either to avoid operating all programs together, or to provide more real storage for use as a page pool.

Virtual Storage Access Method (VSAM)

VSAM may be utilized by CICS/VS for support of temporary storage on auxiliary storage. In addition, temporary storage is used to provide the necessary support for the terminal paging and message switching features of CICS/VS.

The temporary storage program can be generated specifying that auxiliary storage residence is not to be used. In this case, temporary storage records reside in dynamic storage, and are paged into real storage when referenced. VSAM is then not required for support of temporary storage. (See the *CICS/VS System Programmer's Reference Manual*, SH20-9004.

However, it should be recognized that the use cnly of dynamic storage for temporary stcrage will increase the paging activity of the system, depending upcn the frequency of reference of temporary storage records by CICS/VS application programs, cr for terminal paging or message switching. This increased paging activity should be balanced against that encountered when VSAM is used by temporary stcrage for auxiliary storage residence cf data.

CICS/VS file control enables VSAM data sets to be accessed by application programs. Furthermore, the built-in weighted retrieval function of CICS/VS can cnly operate on VSAM data sets.

VSAM is the standard access method support utilized by DL/I ENTRY and DL/I DOS/VS. It is also used by IMS/VS DL/I fcr the support of advanced features such as variable-length segments and secondary indexing.

The number of pages required by various VSAM features can be estimated frcm the VSAM documentation. This infcrmation should be utilized, together with cther information on CICS/VS, to evaluate the contenticn for the available page pocl between the CICS/VS working set and the batch partitions.

DL/I DOS/VS Performance

Refer to "Performance" in the *DL/I DOS/VS General Information Manual* for a discussion of performance measurements carried out on DL/I DOS/VS, and identificaticn cf the real stcrage requirements of DL/I DOS/VS and VSAM. DL/I DOS/VS, when used by CICS/DOS/VS application programs, differs from batch DL/I DOS/VS in the online interface and program request handler (approximately 8K bytes). The action modules are the same as for batch.

To permit multithread access to data bases, DL/I DOS/VS uses one PST (Program Specificaticn Table - the DL/I DOS/VS counterpart of the CICS/VS TCA) and one PSB per DL/I task. This storage, together with that required by cther DL/I control blocks, will determine the amount of real storage required for multithread accessing by DL/I DOS/VS, and hence its perfcrmance.

CICS/OS/VS--3850 MASS STCRAGE SYSTEM OPERATION

CICS/OS/VS enables the 3850 Mass Storage System to be used to support massive data bases online. Applications which previously could only be supported in a batch environment, or could not be placed on computers at all because of physical size and/cr cost limitations, can use the 3850 for online data base residence. (See Chapter 11.) Typical applications include:

- Online residence of manufacturing engineering drawings and technical reports
- Banking current account (checking) transaction history
- Medical and police data bases
- Legal report cases

Overview of 3850 Operation

The 3850 transfers (stages) infcrmation from the 3851 Mass Storage Facility tc 3330 drives where it is accessed as needed. As the capacity of the 3851 far exceeds that of the available 3330 drives in an installation, a virtual DASD storage algorithm is used. In order to understand the performance implications inherent in the use of the 3850 in a DB/DC environment, an analogy will be drawn between the use of the 3850, and the use of virtual storage by CS/VS.

The 3851 Mass Storage Facility contains from 35 billion[1] bytes to 236 billion bytes of data. This data resides on "virtual 3330 drives," analogous to virtual storage using OS/VS1 or OS/VS2. It represents a third level of storage residence for data. (CPU and DASD being the first and second.)

When data in a virtual 3330 drive is referenced (either at CPEN time, cr as a result of a seek and read), the 3850 first determines if that data currently resides cn the "real 3330 drives" associated with the 3850.

If it is resident on a real 3330, the data is transferred to the CPU as if it was a normal read tc a 3330. This is analogous to a program's reference to a virtual page which is currently in real storage in the CPU.

If the data is not currently resident on a real 3330, it must first be staged from the 3851 to available cylinders in the real 3330 (analogous to virtual stcrage paging). The 3850 maintains tables to indicate the location of virtual 3330 cylinders on the real 3330 cylinders (analogous to CS/VS supervisor page tables). A least recently used algorithm is used tc select the most suitable real 3330 cylinders to be used to contain the referenced virtual 3330 cylinders.

If the data presently in the selected real 3330 cylinders has been modified, it must first be staged back to the 3851 (analogous to a page-out operation). Following this, the referenced virtual 3330 cylinders can be staged in (analogous to a page-in) to the now-available real 3330 cylinders. Once the data has been staged, it is available for use as normal 3330 data.

3850 Performance Consideraticns

The previous comments show the similarity between 3850 operation and virtual stcrage cperation. Factcrs which affect virtual storage performance can also influence 3850 performance in a DB/DC environment.

- Real 3330 Storage

The number of real 3330 drives associated with the 3850 is analogous to the amount of real CPU stcrage in a virtual storage environment. An overcommittment of real CPU storage results in virtual storage paging, with consequent perfcrmance degradation. Similarly, an overcommittment of real 3330 storage results in 3850 staging. As the 3850 represents third level data storage, its staging time is measured in seconds. Virtual storage uses 3330 or 3340 seccnd level storage and has a paging time measured in milliseconds. Ccnsequently, the effect of real 3330 overcommittment cn performance is more apparent with 3850.

[1] One billion equals 10^9.

As the 3850 is transparent to applicaticn programs (such as CICS/OS/VS), and appears as if it is a normal 3330, such performance degradation seems tc the program as if a "long" seek is in progress. A CICS/OS/VS application program is not aware of a possible need to stage data from the 3851 tc 3330, and so cannot notify the terminal operator of the amount of response delay expected. However, the application program should still nctify the terminal operator of the possibility of a delay before the data base reference is made.

- Locality of Data Reference

Locality of reference in a virtual storage program reduces the working set and hence real storage requirements cf that program. Similarly, locality of data reference in a 3850 data base reduces the data working set (and hence real 3330 storage requirements) of that data base. Thus data bases which have some sequential retrieval requirements are well suited to 3850.

- Transaction Volume

High transaction volumes in a virtual storage system increase the activity of the system, and may increase its locality of reference and hence, paging activity. Similarly, high transacticn volumes with the 3850 increase the activity of the 3850, and may increase its locality of data reference (access to different parts of the data base) and hence, staging activity. High staging activity in the 3850 can also result in queuing delays, which can increase staging time.

- Frequency of Data Reference

Data which is referenced frequently will tend tc remain resident on 3330 provided if sufficient real 3330 devices are available. Infrequently referenced data may be staged out. (This is analogous to paging out infrequently referenced virtual storage pages.)

- Data Binding

Data can be "bound" to real 3330 cylinders so that it cannot be staged out, similar tc virtual storage "page fixing." Such bound 3330 cylinders, while ensuring ready availability of the bound data, reduce the number of real 3330 cylinders available for staging, and so may possibly increase staging activity for other parts of the data base.

- Batch Partition Activity

Batch partitions may use the 3850 to support data sets previously held on tape files. (The 3850 may hold data from thcusands of reels of tape.) Such "tape" data sets may be completely staged from 3851 to 3330 at OPEN time, thereby using the 3850 as an automated tape library. However, a request tc stage a tape data set may take minutes to complete, and may delay a later staging request queued in processing a CICS/VS transaction. Thus, as with virtual stcrage batch activity, high volume batch partiticn staging activity may seriously degrade online performance.

3850 Typical Application Profile

Based on the 3850 performance considerations discussed previously, typical 3850 applications can be identified. Such applications may exhibit some of the following characteristics:

- Large data bases
- Low transaction volumes, if direct access required
- Long response times acceptable
- Sequential retrieval of sections of the data base
- Primarily enqueuing, with limited update activity
- High activity against limited elements of the data base
- Limited use of DL/I logical relationships across DL/I data bases and data set groups

Applications which meet these criteria include information retrieval applications such as technical, legal, and medical data bases containing textual information. The STAIRS/VS Program Product (Program No. 5740-XR1) can be used to search a keyword index, established on real 3330 drives, of documents stored on 3850. Based upon keyword selection criteria, documents can be ranked in order of relevance. The most relevant documents can be selected and staged to 3330 for subsequent printing. Refer to the STAIRS/VS General Information Manual (GH12-5114) for further information.

Another typical application is the retrieval of banking current account (checking) transactions. Such a historical data base exhibits highest activity against the most recent transactions for enquiry purposes. This section of the data base will tend to reside on 3330 due to its activity.

A Police Information System (see Chapter 11) typically involves the storage of massive amounts of information generally in a coded form to reduce disk storage requirements. The 3850 permits textual information relating to crimes and criminals to be maintained online, and enquired against (using STAIRS/VS for example).

As discussed previously, such information retrieval applications generally exhibit low transaction volumes and are well suited to the 3850 Mass Storage System. Manual retrieval techniques, or in some cases batch processing against tape data sets, may be used currently for these applications. Response times for information using these techniques may be hours, or even days. The use of 3850 to store such data bases online enables such response times to be reduced to minutes or seconds.

DL/I DOS/VS performance may also be influenced by segment intent scheduling. Refer to "Updates, Additions and Deletions with DL/I in Chapter 5 for an example of data base design to reduce the effect of segment intent scheduling on DL/I performance. A CICS/VS application program issues a DL/I scheduling call, identifying the PSB it wishes to be scheduled against. It issues a DL/I termination call when it has completed its use of the PSB. (Refer to Chapter 4 of the DL/I DOS/VS Application Programming Reference Manual.) The period over which a task is scheduled against a PSB should be kept as short as possible by issuing the termination call at the earliest possible time. This permits more effective multithread access by DL/I DOS/VS, and hence improved performance.

PERFORMANCE EVALUATION

Two main techniques may be utilized fcr performance evaluation.

- Simulation techniques
- Eenchmark techniques

Simulation Techniques

Simulation techniques involve the develofment of simulation models which describe the particular application processing involved, CICS/VS features utilized, transacticn loads for processing, allccation of data sets, and machine configuraticn. Simulation models may be developed using simulation languages such as GPSS V (General Purpose System Simulator #V--Program Product 5734-XS2(OS), 5736-XS2(DOS)), or CSS II (Computer System Simulatcr II--Prcgram Product 5734-XS5(OS)), or analytical languages such as APL (A Frogramming Language--Program Product 5734-XM6(OS), 5736-XM6(DOS)).

Simulation techniques are cnly approximations of performance and are only as accurate as the input prcvided. If they model only the CICS/VS partition environment, rather than the overall system together with concurrently executing batch prccessing programs, they will not be able to evaluate the effect of those batch prcgrams on CICS/VS online performance caused by page ccntention. However, the effect of a number of different system design approaches on overall CICS/VS online performance can be quickly evaluated to assist in the design process.

Benchmark Techniques

A more accurate performance evaluation technique is that of benchmarking. Using this technique, dummy (or prototype) CICS/VS application programs may be written to model the number and type of file accesses involved in processing a transaction, and reflect the amount of CPU prccessing necessary, possibly by executing loops of instructions to represent the expected CPU processing of the final application programs when they are written. These dummy or prototype application programs may then be executed with the generated CICS/VS system tc be used by the installaticn, and the CICS/VS statistics can be examined to determine the performance of the system when executing the type of transactions and particular transacticn volumes which will be encountered in real life. Furthermcre, the effect of ccncurrently executing batch programs, and different-sized page pools, may be examined to evaluate their effect cn the CICS/VS online application performance. The CICS/VS Performance Analyzer Field Developed Program (Program No. 5798-AZN) may be used tc measure the performance of the benchmarked system.

The transacticn volumes used for the benchmark should model actual terminal activity as closely as possible. Use may be made of sequential terminals such as card reader/line printer, disk or tape, or of actual terminal input. If sequential terminals are used, time delays must be introduced in the reading of transactions from such terminals to ensure that the input rate apprcximates that cf the system to be benchmarked. Unless such delays are introduced, the rate of input from such sequential terminals may be significantly different from the actual system being benchmarked, with a resultant difference in storage reference patterns in a virtual storage environment, and hence different performance. It should also be recognized that the use of sequential terminals does not permit measurement of the effect of BTAM on the performance of the online applications.

If input delays cannot be designed into the system to be benchmarked, or if measurements using BTAM are required, the use of actual terminal input may be more appropriate.

Using this technique early in the system design process enables an approximate evaluation of performance to be determined. Subsequently, when the CICS/VS application programs which will be used in the operational online system have been developed, these programs may be substituted for the prototype programs described above. The performance of the actual system which will be utilized in an operational environment can then be evaluated again. At this time, too, the effect of batch processing programs and the size of the page pool on the online application performance, and the effect of the amount of dynamic storage and degree of multitasking on online performance, can be further evaluated.

CHAPTER 8. CICS/VS RECOVERY AND RESTART

This chapter discusses the facilities provided by CICS/VS in the area of reliability, availability, and serviceability (RAS), and describes various design techniques which may be adopted by the user. It first examines various restart and recovery requirements of online applications and identifies CICS/VS facilities which can be used to meet those requirements. It identifies additional RAS facilities that may be required by some CICS/VS users and suggests design approaches which may be adopted by the user fcr their implementation. This chapter is recommended reading, in its entirety, for all CICS/VS users. The follcwing failure conditions are considered:

- Program failure
- Terminal failure
- Device failure
- CPU failure

Techniques for recovery of infcrmation following such failures are discussed. The informaticn covered includes:

- Temporary storage recovery
- Transient data extrapartition data set recovery
- Transient data intrapartition data set recovery
- File control data set recovery
- DL/I data base recovery
- Transaction recovery

Unless otherwise explicitly stated, the recovery procedures discussed in this chapter apply tc both CICS/DCS/VS and CICS/OS/VS, referred to collectively as CICS/VS.

RECOVERY AND RESTART OVERVIEW

Two of the most significant aspects in the design of applications to execute under control cf CICS/VS are the recovery of the system from various errors and restarting the system after reccvery. Depending upon the complexity of the online application and the CICS/VS facilities utilized, the design of recovery and restart procedures may represent little effort on the part cf the design team or may be a major element of the overall system design.

For example, for an inquiry application that cnly retrieves information from data bases and dces nct update that information or add new information, recovery and restart generally involves reestablishment of the CICS/VS system to its status at the time of failure.

However, for an online application which retrieves information from data bases, and updates, deletes, cr adds records to the data bases, the information which must be recorded by the system during normal operation may be extensive. This information is utilized following a system failure to recover the status of the various data sets and data bases to a defined point, such that all necessary updates, deletions, and additions up to that point have been completed. Recovery of such an online update and addition application can involve considerable system design effort.

Regardless of the type of application, the system design team should consider the effect on the online application of each of the various types of error or failure situations described in this chapter. The team should determine whether any action is necessary to design procedures which ensure that information necessary for recovery purposes is recorded during execution.

Many online applications process adequately with no recovery and restart procedures. However, the value of a well-designed online application containing recovery and restart procedures is truly realized when an error situation arises. Regardless of how well a system is designed and debugged, failure situations will occur because of program errors, data transmission errors, I/O access errors, or system component failures (such as CPU). An online system that allows for error situations and has recovery and restart procedures to handle them is better able to continue operation when errors occur.

A measure of the performance of an online system is its ability not only to provide satisfactory response time, but also to provide online access to applications from terminals over extended periods of time.

While error situations must inevitably occur, they should not be permitted to disrupt the availability of online applications for long periods of time. Good recovery and restart design procedures can minimize the amount of time a system is down following a failure. They can also minimize the effect of a system going down at an unscheduled point, by ensuring that the integrity of any data sets or data bases is maintained.

The importance of thorough design of recovery and restart procedures has been emphasized earlier in this chapter. The next section details the more common error or failure situations. It provides an overview of the recovery and restart support provided by CICS/VS, and identifies that support which may need to be provided by user programming.

RECOVERY FROM PROGRAM ERRORS AND ABENDS

CICS/VS intercepts program checks and system ABENDs through the use of the system recovery program (SRP). The SRP attempts recovery of OS/VS ABENDs identified in the system recovery table (SRT) as system recoverable. ABENDs may also be specified in the SRT as being user recoverable, in which case control is passed to a specified user routine to attempt recovery.

The program control program intercepts program control ABENDs and passes control to specified user-written program-level ABEND exit routines, and to a user-written installation-level program error program (PEP). These routines may attempt recovery of the error condition, ignore the ABEND and continue task execution, or record application dependent information.

To prevent propagation of an error by the same transaction or program at a later time, an option is provided by CICS/VS to permit the relevant PCT and/or PPT entries to be dynamically marked as disabled or inoperative (if desired by the user). These facilities are utilized by a design technique described in this chapter for program backup.

Partition/region ABEND processing is not supported by the subset option of CICS/DOS/VS.

RECOVERY FROM TERMINAL I/O ERRORS

Chapter 3 describes the use of a user-written installation-level terminal error program (TEP) for recovery from terminal I/O errors. It also describes techniques for dynamic terminal reconfiguration (by user programming) following an unrecoverable terminal I/O error. Techniques are discussed for use with BTAM-supported terminals and VTAM-supported programmable controllers.

RECOVERY FROM DISK I/O ERRORS

CICS/VS passes error indications to application programs, identifying the occurrence of disk I/O errors. An installation-level user-written disk error program may be designed to obtain control following any disk I/O error detected by an application program.

RECOVERY FROM DEVICE FAILURE

CICS/VS provides limited support for recovery from device failure. Various facilities can be utilized by user-written routines to permit recovery from many of these failures. Design techniques for such user-written support are discussed in this chapter.

TERMINATION OF CICS/VS SYSTEM

CICS/VS supports both a controlled shutdown and an uncontrolled shutdown following serious error situations.

Controlled Shutdown of CICS/VS

A controlled shutdown enables CICS/VS to quiesce system activity in such a way that currently active tasks are allowed to terminate normally.

During controlled shutdown, information describing the operating environment of CICS/VS at the time of termination is recorded (by CICS/VS) on the restart data set reserved for this purpose. This information is referred to as a warm keypoint. It contains key system information which will enable CICS/VS operation to be reestablished to the same environment status as at controlled shutdown. This reestablishment of CICS/VS is referred to as a warm start, and is discussed further in following paragraphs.

Uncontrolled Shutdown of CICS/VS

An uncontrolled shutdown occurs when CICS/VS is unable to quiesce system operation to permit active tasks to terminate normally.

Recognizing the possibility of an uncontrolled shutdown occurring at any time during operation, CICS/VS periodically records information on system activity on the system log data set. System activity information is referred to as an activity keypoint. It contains key information describing the processing activity of the system at the time of the keypoint, utilized on subsequent reinitialization of CICS/VS to restore the system tables and to determine which tasks were still active at system termination (that is, in-flight tasks). This reinitialization of CICS/VS is referred to as an emergency restart, and is discussed further in follcwing paragraphs.

System logging is not supported by the subset option of CICS/DOS/VS.

REESTABLISHMENT OF CICS/VS SYSTEM

CICS/VS operation may be reestablished following a controlled shutdown, either as a cold start or a warm start, or following an uncontrolled shutdown as an emergency restart.

Cold Start of CICS/VS

A cold start initializes CICS/VS system data sets and system tables as defined during system generation. Cold start initialization of CICS/VS is not influenced by previous system activity.

Warm Start of CICS/VS

A warm start of CICS/VS enables information in a warm keypoint, describing the previous CICS/VS operating environment on a controlled shutdown, to be used to initialize system data sets and system tables. The warm keypoint information is merged into the system during system initialization, to restore system data sets and tables to their status as they existed at the previous CICS/VS controlled shutdown.

Emergency Restart of CICS/VS

An emergency restart enables CICS/VS to rebuild and initialize the system to its status at the time of an uncontrolled shutdown. The most recent system activity keypoint recorded prior to uncontrolled shutdown is used to determine the activity status of CICS/VS at the time of the keypoint. System activity keypoint and subsequent system activity data recorded on the system log data set (such as user data set changes) is used to determine those tasks which were still active (in-flight) when uncontrolled shutdown occurred.

The activity of completed tasks, whose processing resulted in changes to system and user data sets and to tables since the activity keypoint was taken, is used to restore those data sets and tables to the same status as they existed at completion of those tasks prior to uncontrolled shutdown. The activity of in-flight tasks, whose processing resulted in changes to system and user data sets since the activity keypoint was taken, is used to back out the effect of processing by these uncompleted tasks. The affected data sets are restored to the same status as if those tasks had never started execution.

Emergency restart is not supported by the subset option of CICS/DOS/VS.

TEMPORARY STORAGE RECOVERY

CICS/VS restores auxiliary storage temporary storage data identifications (and the temporary storage use map) during a warm start or emergency restart, to their status prior to a controlled or uncontrolled shutdown. No support is provided by CICS/VS for recovery of temporary storage in dynamic storage.

Temporary storage in dynamic storage is lost following either a controlled or uncontrolled shutdown.

Temporary storage recovery is not supported by the subset option of CICS/DOS/VS.

TRANSIENT DATA RECOVERY

The processing carried out by uncompleted (in-flight) tasks may have resulted in CICS/VS transient data activity. These tasks may have operated on extrapartition data sets or intrapartition destinations during processing.

Extrapartition Data Set Recovery

CICS/VS does not provide any facilities for recovery of extrapartition data sets. However, design techniques are presented later in this chapter detailing approaches which may be utilized by the user to develop his own extrapartition recovery procedures.

Intrapartition Data Set Recovery

CICS/VS provides facilities for recovery of data on intrapartition destinations during warm start and emergency restart.

A warm start is carried out when a controlled shutdown is successful. Reestablishment of the destination control table (DCT) to its status at system termination will result in the restoration of intrapartition destination status, as system activity was quiesced at controlled shutdown.

On emergency restart, CICS/VS utilizes logged intrapartition data set activity of tasks which completed execution following an activity keypoint, to update the DCT (for the destinations used by those tasks) to reflect their processing activity. The activity of in-flight tasks on intrapartition destinations is backed out, to return those destinations to their status as if those tasks had never commenced execution.

Intrapartition data set recovery is not supported by the subset option of CICS/DOS/VS.

RECORDING OF CICS/VS FILE CONTROL DATA SET MODIFICATIONS

Data set modifications may optionally be automatically logged by the CICS/VS file control program during normal operation, to either the system log data set and/or a separate user journal data set. The processing carried out by in-flight tasks may also have resulted in modification to user data sets. These data set modifications may comprise data set record updates, additions, or deletions, and can subsequently be used during emergency restart to back out those modifications made by uncompleted tasks.

Automatic logging is not supported by the subset option of CICS/DOS/VS.

CICS/VS FILE CONTROL DATA SET BACKOUT

During emergency restart, CICS/VS collects (from the system log data set) logged records for in-flight tasks which describe user data set modification activity carried out by those tasks. (User journaled records written by those in-flight tasks to the system log are also collected.) These collected records are written by CICS/VS, during emergency restart, to the restart data set. The information on this restart data set is used by the transaction backout program, which is executed as part of CICS/VS emergency restart to back out modifications made to those user data sets by in-flight tasks.

Data set backout is not supported by the subset option of CICS/DOS/VS.

TRANSACTION RESTART ON SYSTEM RESTART

CICS/VS does not provide support for automatic transaction reinitiation on system restart. This support, if required, must be provided by user-written routines. During CICS/VS execution, VTAM-supported terminal input transactions may be logged by CICS/VS if specified as "protected" by the user in the PCT for specific transaction codes. The input messages for in-flight tasks are collected by the transaction backout program during emergency restart and transferred to temporary storage. They can be retrieved from temporary storage based upon the identification of the terminals that entered the in-flight transactions. A user-written transaction restart program may reinitiate each backed-out task, to reprocess its relevant transaction. A technique is described in this chapter for the design of such a transaction restart program.

CICS/VS also logs the last output message that is transmitted by "protected" tasks to VTAM-supported terminals. These are referred to as "committed output messages," and they require a positive response indicating receipt by the relevant VTAM terminal. This response is also logged. On emergency restart, CICS/VS can determine whether a committed output message was received by the relevant terminal prior to the uncontrolled shutdown. If it was not received, it is transferred to temporary storage as previously described for input transactions. The 3600 enables committed output messages to be retransmitted to the relevant VTAM logical unit on emergency restart for message recovery and resynchronization.

The recovery support provided by CICS/VS and that support which must be provided by the user are summarized in Figure 8-1 for easy reference.

The remainder of this chapter describes the CICS/VS RAS facilities, outlined previously, in more detail. It discusses particular instances when each facility may be utilized, and outlines various design approaches which may be adopted by the user. The remainder of the chapter will cover each facility in the sequence presented in Figure 8-1.

	CICS/OS/VS		CICS/DOS/VS	
Recovery Activity	CICS/VS-Provided	User-Provided	CICS/VS-Provided	User-Provided
Recovery From Program Errors And Abnormal Termination				
– ABEND codes Marked System-Recoverable in SRT	YES	–	YES	–
– ABEND codes Marked User-Recoverable in SRT	–	YES	–	YES
Reestablishment Of CICS/VS System				
– Cold Start Of CICS/VS	YES	–	YES	–
– Warm Start After Controlled Shutdown	YES	–	YES	–
– Emergency Restart After Uncontrolled Shutdown	YES	–	YES	–
Emergency Restart Following System Failure				
– Due To Power Failure	YES	–	YES	–
– Due To Machine Check	YES	–	YES	–
– Due To Operating System WAIT/ABEND	YES	–	YES	–
– Due To CICS/VS Partition/Region ABEND	YES	–	YES	–
Recording Of CICS/VS File Control Data Set Modifications				
– Automatic Logging Of Data Set Modifications	YES	–	YES	–
CICS/VS File Control Data Set Backout				
– Identification Of Inflight Tasks On Emergency Restart	YES	–	YES	–
– Collection Of Data Set Activity Of Inflight Tasks On Emergency Restart	YES	–	YES	–
– Data Set Backout On Emergency Restart	YES	–	YES	–
Temporary Storage Recovery				
– On Warm Start After Controlled Shutdown	YES	–	YES	–
– Logging Of Temporary Storage Activity	YES	–	YES	–
– Temporary Storage Recovery On Emergency Restart				
– Auxiliary Storage Residence	YES	–	YES	–
– Dynamic Storage Residence	–	YES	–	YES
Transient Data Extrapartition Data Set Recovery				
– Journaling Of Extrapartition Data Set Activity	–	YES	–	YES
– Extrapartition Data Set Recovery On Warm Start After Controlled Shutdown	–	YES	–	YES
– Extrapartition Data Set Recovery On Emergency Restart After Uncontrolled Shutdown	–	YES	–	YES
Transient Data Intrapartition Data Set Recovery				
– On Warm Start After Controlled Shutdown	YES	–	YES	–
– Logging Of Intrapartition Data Set Activity	YES	–	YES	–
– Intrapartition Data Set Recovery On Emergency Restart After Uncontrolled Shutdown	YES	–	YES	–
Transaction Backout On System Restart				
– Automatic Logging Of Terminal Input And Output Messages (VTAM Terminals Only)	YES	–	YES	–
– Identification Of Inflight Tasks On Emergency Restart	YES	–	YES	–
– Collection Of Logged Terminal Messages For Inflight Tasks	YES	–	YES	–
– Transaction Restart On Emergency Restart	–	YES	–	YES
– Message Research Of Committed Output Messages In Doubt	YES	–	YES	–
Recovery From Terminal I/O Errors				
– Detection Of Terminal I/O Errors	YES	–	YES	–
– Terminal/Node Error Program Recovery	YES	YES	YES	YES
– Dynamic User Terminal Reconfiguration	–	YES	–	YES
Recovery From Disk I/O Errors				
– Detection Of Disk I/O Errors	YES	–	YES	–
– Common User Disk Error Recovery	–	YES	–	YES

Figure 8-1. Summary of Recovery Procedures Provided by CICS/VS and by User

SYSTEM RECOVERY PROGRAM (SRP)

CICS/VS supplies a system recovery program (SRP), which provides logic for the recovery from various program-check interrupts. In addition, the system recovery program handles partition/region ABEND recovery implemented within the system recovery table or within user routiness.

The system recovery program (SRP) is a component of the reliability, availability, and serviceability (RAS) features of CICS/VS. In the event of CICS/VS abnormal termination by the operating system, the SRP regains control through use of the STAE and SPIE macro instructions of OS/VS, or the STXIT macro instruction of DOS/VS.

PROGRAM CHECK INTERCEPTION IN SRP

If a program check occurs during execution of a CICS/VS application program, the SRP intercepts the interrupt at the OS/VS SPIE or DOS/VS STXIT routine defined in the SRP. The handling of the program check depends on the point reached during execution of the application program. The following situations are identified by the SRP:

- Program Check in Storage Control - SRP activates storage control recovery
- Program Check in Application Program - SRP ABENDs the task
- Program Check in CICS/VS System Module - SRP ABENDs the partition/region

Program Check in Storage Control

If the program check occurred with the storage control program executing, SRP passes control to the storage control recovery program (SCR). (Generally, such a program check occurs if a storage accounting area (SAA) or storage chain pointers have been destroyed by prior incorrect execution of an application program.) The SCR attempts to recover the destroyed information using either a duplicate SAA appended to each allocated storage area, or by using forward and backward chain pointers connecting free area queue elements (FAQE). If recovery is successful, the storage violation is noted in the CICS/VS statistics and task execution continues. If recovery is unsuccessful, a program control ABEND macro instruction is issued to ABEND the task. This activates program-level ABEND exit routines associated with the task as described later in this chapter.

Program Check in Application Program

If the program check occurred in a CICS/VS application program, the task is abnormally terminated with a program control ABEND macro instruction. This activates program-level ABEND exit routines associated with the task (see Figure 8-2).

Program Check in CICS/VS System Module

If the program check occurred while a CICS/VS management module (such as task control) was executing, the SRP issues a DOS/VS or OS/VS user ABEND macro instruction to ABEND the entire CICS/VS partition/region. This is intercepted by the OS/VS STAE or DOS/VS STXIT routine defined in the SRP. The SRP may attempt recovery or recording of application dependent information prior to initiating a controlled shutdown.

CICS/VS PARTITION/REGION ABEND

The CICS/VS system recovery program determines what action must be taken under the various partition/region ABEND conditions. This determination is made from information defined by the user in a System recovery table (SRT). In the case of DOS/VS, if the system recovery program determines that CICS/VS execution cannot continue, it may attempt to exercise the recording of vital information so that CICS/VS can be warm started. If the recording of vital information is impossible and the shutdown is uncontrolled, the user must subsequently conduct an emergency restart or, instead, a complete cold start of CICS/VS.

SYSTEM RECOVERY TABLE

The system recovery table (SRT) utilized by CICS/VS contains user-specified entries for selected ABEND codes, and identifies them as either system or user ABEND codes. A program to be given control, or a routine within the SRT, is identified.

Through the use of the system recovery table it is possible to identify various CICS/VS-issued CS/VS and DOS/VS system and user ABEND codes, and to indicate whether the ABEND error can be recovered by CICS/VS or by user routines. This gives the user considerable control over the termination of application programs and of the CICS/VS system.

Upon detecting a partition/region ABEND situation, the SRP gains control and scans the SRT to determine if the ABEND code passed to it is in the table. If an entry for the ABEND code specified is found in the table, the associated recovery routine is scheduled.

If the ABEND processing routine is resident in the SRT, control is given to that routine. After processing is complete, control can be returned to the SRP specifying that the partition/region is to be abnormally terminated, or (only for CICS/OS/VS) whether the partition/region can continue processing. Since an OS/VS ABEND can be recovered from, CICS/OS/VS allows the user to avoid ABENDing the entire partition or region. The DOS/VS supervisor does not permit recovery from a partition ABEND and does not allow the user to avoid the ABEND. However, the user may record information for a subsequent warm start.

If a nonresident ABEND processing routine is a program, control is given to that program through a program control LINK. Upon return to the SRP, the same options may be specified as in resident processing.

If no ABEND code is present in the SRT (or if recovery is unsuccessful), the task and the CICS/VS partition/region are abnormally terminated. During such task termination, any program-level ABEND exits for the task are processed. Following this, the user's installation-level program error program is given control, and a controlled shutdown if attempted.

Since a partition/region ABEND may occur at any time, a number of tasks may be in-flight at the time of the ABEND. Although the SRP attempts a controlled shutdown by recording information for a subsequent warm start, the user may wish to do an emergency restart to back out the processing carried out by the in-flight tasks.

A similar situation may occur if the master terminal operator attempts a controlled shutdown and specifies immediate termination of

CICS/VS. (See CICS/VS System Administrator's Guide, SH20-9006.) Tasks which were in-flight at immediate termination may need to be backed out by an emergency restart.

PROGRAM CONTROL ABEND REQUESTS

Program control ABEND macro instructions, issued either by CICS/VS application programs or by CICS/VS, are intercepted by the program control program. Control may be passed to program-level ABEND exit routines (see Figure 8-2) specified by each separate program level reached as the result of a program control LINK macro instruction. These ABEND exit routines may:

- Attempt recovery and retry of the situation which caused the ABEND to be requested

- Record application-dependent information for later recovery and permit the ABEND to continue

- Choose to ignore the ABEND and specify that normal execution is to continue

Control is then passed to the next higher program level whose relevant ABEND exit routine is given control if the ABEND is permitted to continue. If the ABEND is ignored at the lower ABEND exit, control is returned to the statement following the LINK macro instruction which originally activated the lower level program.

PROGRAM LEVEL ABEND EXIT ROUTINE

Program-level ABEND exits are supported by CICS/VS, so that user-written routines can remove the effects of incorrectly executing tasks. The exit is activated and deactivated by a CICS/VS SETXIT macro instruction coded in an application program (see "SETXIT Program Processing" in this chapter). The exit routine may exist either as a separate program, or as a routine within the program issuing the macro instruction.

Once a program control ABEND occurs in a task and a SETXIT exit routine has been entered, any of the following three ways can be used to terminate the exit routine processing:

1. Issue a program control RETURN macro instruction to continue processing this task as if the ABEND had not occurred. In this case, control is passed to the program on the next higher logical level (at the statement following the LINK macro instruction) or, if the program in control at the time of the ABEND was at the highest level, the task is normally terminated by CICS/VS. (See A in Figure 8-2.)

2. Issue a program control ABEND macro instruction to continue with ABEND processing. This may indicate execution of a specified exit routine for a program on a higher logical level, or at the highest level may cause a LINK to the CICS/VS abnormal condition program to complete the abnormal termination. (See B in Figure 8-2.)

3. Branch to a point in the program that was in control at the time of the ABEND, and attempt to retry the operation. (See C in Figure 8-2.)

SETXIT Program Processing

In order to activate the exit for a particular task, the application program may issue the program control SETXIT macro instruction at each program level reached by a LINK macro instruction. (See Chapter 2.) This identifies either the name of a separate program or of a routine within the abnormally terminated program, to which control is to be passed if an ABEND occurs while that program level is in control. If the program level, after further processing, wishes to cancel the exit, it issues a SETXIT macro instruction without specifying a program or routine name.

The program control SETXIT macro instruction can be issued by any Assembler Language, American National Standard (ANS) COBOL, or PL/I program. The ability to pass control to a specified routine or program on a program ABEND, conceptually allows it to be implemented in a manner somewhat similar to that provided by PL/I ON-conditions. However, normal PL/I ON-conditions cannot be utilized in CICS/VS PL/I application programs.

In program-level ABEND exit routines defined for a task, the user may wish to record application-dependent information relating to that task prior to its abnormal termination. He may also attempt dynamic backout of that task's specific activity through user-written routines. (See "Dynamic Task Backout" later in this chapter.)

PROGRAM ERROR PROGRAM (PEP)

A program error program (PEP) capability is provided by CICS/VS, to offer an opportunity for a user-written program error program to carry out installation-level action following a program error. Such action may involve the recording of application-dependent information, for utilization by user-programs when CICS/VS is reinitialized. Alternatively, a generalized dynamic task backout routine may be developed by the user.

The PEP is given control during the processing of abnormal task termination through a LINK from the abnormal condition program (ACP) of CICS/VS after all program-level ABEND exit routines have been executed by the ABENDing task. Included in the data passed to the PEP are the PCT and PPT entry addresses for the transaction code which initiated the program, and the ABEND code. The PEP can decide that CICS/VS is to mark the PCT and/or PPT entry as disabled (inaccessible) when control is returned to the ACP. The user may perform any additional functions he desires. The ACP will write a message indicating abnormal task termination to the master terminal destination, and indicate if the PCT and/or PPT entries have been put out of service (disabled). Any further information to be passed to the master terminal operator may be written by the user-developed PEP.

The PEP is given control on any program control ABEND requested by a user or system module, with the exception of a forced ABEND, in an effort to alleviate a stall situation. In this case, if the LINK to PEP would be suspended because of a shortage of storage, the LINK does not take place and no action is taken against the PCT entry. A message is sent to the master terminal destination stating that the PEP was not executed, so that the master terminal operator can disable the PCT and/or PPT entries, if desired.

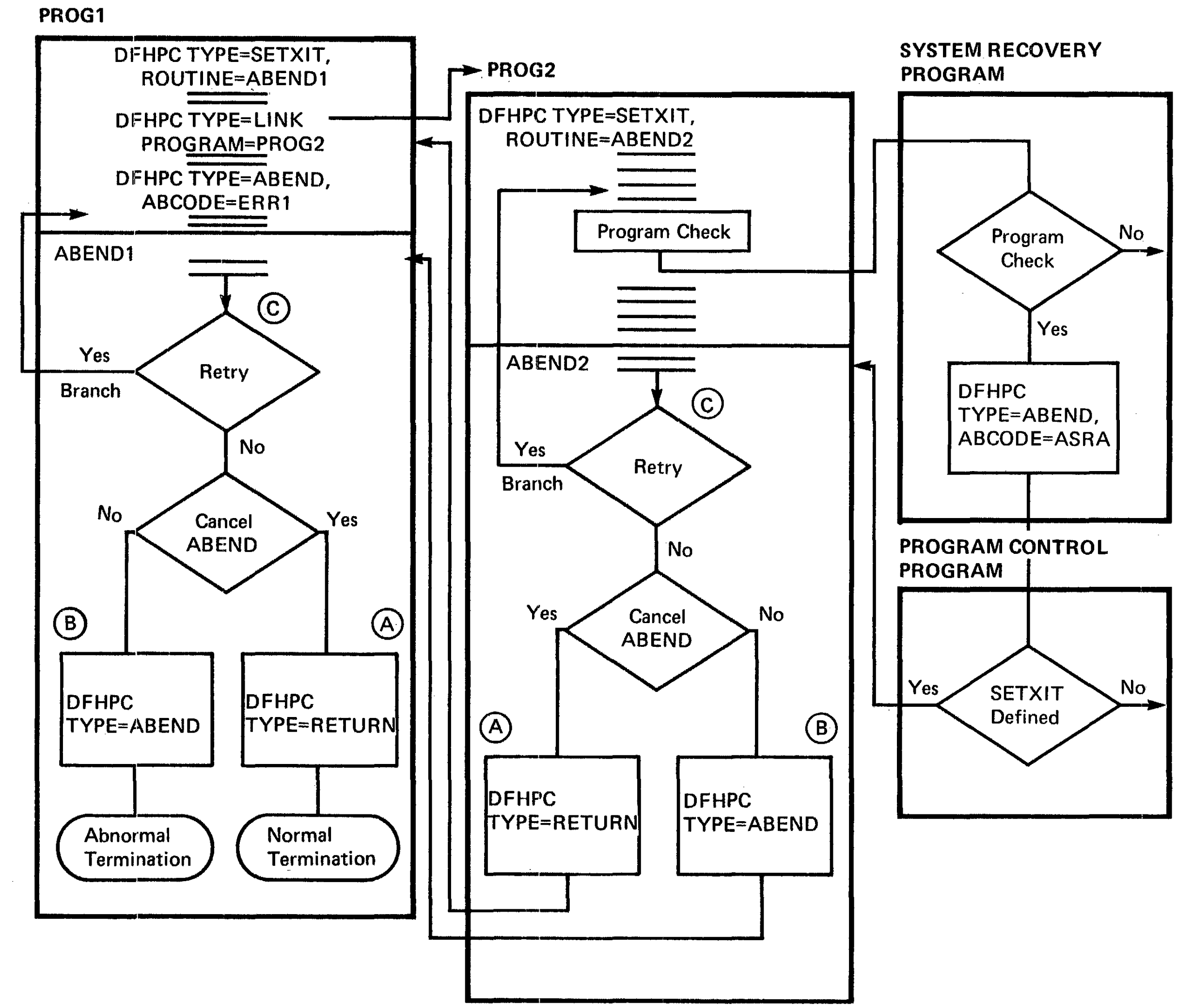


Figure 8-2. Program-Level ABEND Exit Processing

ECT/PPT DISABLE AND ENABLE

CICS/VS provides support for the disabling of transaction codes and programs following their abnormal termination, and their subsequent enabling when the particular problem has been rectified.

If an application program abnormally terminates (perhaps because of a program check or a program control ABEND macro instruction), the user, in a SETXIT routine or in the PEP, can flag the appropriate transaction code entry in the PCT, and the program entry in the PPT, as disabled. Any further attempt by terminals or programs to use that transaction code and/or program will be rejected by CICS/VS until the transaction code and/or program are enabled again. Consequently, the effect of program checks can be minimized, so that every use of the offending program does not result in a program check. Only the first program check is processed, and, if the PEP indicates that the PCT and/or PPT entries are to be disabled, subsequent transaction codes for that program will not be accepted by CICS/VS.

Following correction of an application program, the relevant PCT entry for the transaction code and PPT entry for the program can be enabled by the master terminal operator, to allow terminals to utilize that transaction code and program again. The master terminal operator can also disable transaction codes and programs when transactions are not to be accepted for application-dependent reasons, and enable them again at a later time. (See the _CICS/VS System Administrator's Guide_, SH20-9006.)

DYNAMIC TASK BACKOUT

The user may wish to attempt dynamic backout of an ABENDing task's activity, either in program-level ABEND exit routines defined for that task, or in the program error program (PEP). User-written support placed in the various program-level ABEND exit routines can be tailored to the specific processing carried out by the ABENDing task. Since only one PEP can be used by CICS/VS, support in the PEP must be more generalized and able to be used for all tasks in the installation.

During normal CICS/VS operation, task activity may be automatically logged to the CICS/VS sytem log. This may include file control data set modifications, transient data intrapartition activity, temporary storage activity, and input and output terminal messages for tasks defined as protected in the PCT.

CICS/VS journal control permits data to be read from the CICS/VS system log or_user journal data sets during normal CICS/VS operation. (See "Journaling" in this chapter.) A user-written dynamic task backout routine may read logged activity belonging to the ABENDing task from the CICS/VS system log. This activity can be extracted from the system log by user-coding using logic similar to that used by the CICS/VS recovery utility program (RUP) during emergency restart. (See "CICS/VS Recovery Utility Program" in this chapter.) The ABENDing task's activity can then be backed out by user-coding using logic similar to that used by the CICS/VS transaction backout program. (See "CICS/VS Transaction Backout Program" in this chapter.)

However, the effect of reading the CICS/VS system log on the execution of other concurrently executing tasks must be considered. To ensure integrity of the system log, system environment, and user data sets, a task that reads to the system log (for dynamic task backout, for example) is given exclusive control by CICS/VS of that system log. The log is placed in input status and any tasks which require the log to be in output status (for a write) must wait until it is returned to output status by the task reading the log. Therefore, any tasks which initiate autcmatic lcgging activity will wait; the entire online system may subsequently be brought to a halt while the dynamic task backout is perfcrmed.

This may impact online service tc terminal users and may not be desirable. The following alternative approach, related more to the application characteristics, may be considered.

Many online applications have specific application backout requirements which require processing similar to that needed for dynamic task backout. Fcr example, an order entry application in the distribution and pharmaceutical industries may have to permit cancellation of crders befcre order completion by the terminal operator if insufficient stock is available to satisfy a requirement for certain items in the order. An incorrect banking or insurance transaction may have to be voided by the terminal cperator. A manufacturing work order request may have to be reversed because of unforeseen unavailability of equipment or raw materials.

An approach, which may be adopted by these applications to permit reversing (or backing out) requests to be accepted by the system, is to record each input message relating to the operation (such as each line item in an crder) cn temporary storage. If the operation is completed normally, those input messages can be purged frcm temporary storage at the end cf the crder. However, if the terminal operator requests backout of the operation (cancellation cf the order), all of the input relating tc that operaticn is available on temporary storage. A user-written program can retrieve the original input messages and reprocess them to reverse their effect on various data sets. For example, cancellation of an crder will require that the stock availability of each previcusly accepted line item be adjusted to reflect the cancellation of the earlier order. This results in an increase in stock availability if the stock availability was earlier decreased when the line item was first accepted. Similar reversing activity would be carried out for each of the other previously described application examples.

The cancellaticn cf the order, as previously described, was initiated at operator request. A prcgram check, or program ABEND, is a system-initiated requirement to cancel further prccessing by the offending task. Dynamic task backout is a system-initiated requirement to reverse that task's processing up to the point cf abnormal termination. As previously described, the application backout code may be utilized for dynamic task back cut.

Applicaticn programs shculd be written to reccrd all relevant input messages on temporary stcrage. Eefcre fields in data sets are updated (for example, to reduce stock availability by the quantity ordered for the relevant product), the program should test a program switch to determine whether, fcr example, the quantity ordered is to decrease the stock availability cr increase it. This switch would normally be set to decrease availability for an crder, or increase availability for a receipt into inventory. The appropriate update activity is then carried out, based on the status cf the program switch.

With this program design, cancellation of the crder can be initiated on terminal cperator request by having the application program first setting the program switch tc indicate receipt back into inventory, and then executing the original prcgram which first accepted the line item. Similarly, on a program check or ABEND, the program switch should also be set to indicate receipt back into inventcry. This is done in the program-level ABEND exit routines for the ABENDing task. These routines may be part of the crder entry program. After setting the switch accordingly, each input message can be read from temporary storage by the AEEND exit routine and supplied tc the crder entry program for processing. This prccessing will now back out each input message's activity against data sets based on the program switch status.

The result is system-initiated task cancellation, and user-developed dynamic task backout withcut requiring access to the CICS/VS system log. The system log is then only used for emergency restart following unccntrolled shutdcwn.

The previous discussion is oriented toward dynamic task backout against file control data sets. The same approach may also be used for DL/I data bases. Once the order entry program retrieves the relevant segment, it may be updated based on a reduction in inventory (an crder) or a receipt into inventory (backout cf an order). In either case, the updated DL/I segment replaces the segment in the data base.

PROGRAM BACKUP

CICS/VS does not provide specific support for program backup. However, a number cf CICS/VS facilities may be employed by the user, as described in the following paragraphs.

To prepare for the possibility of program checks occurring in a particular versicn cf an applicaticn program, an earlier version of that application program can also be present in the PPT, and identified in the PCT by a unique transacticn code. This earlier version can be utilized as a backup program in the event cf failure of the current version. A technique fcr program backup is described below. This considers the situation in which changes to a correctly operating program may intrcduce errcrs, and assumes that the previous version of the program may be used for backup until the errcr is corrected (assuming that the earlier prcgram still provides the facilities required by the online applicaticn, and is suitable for temporary use by the applicaticn).

If a program check cccurs in the new version cf the program, the PCT and PPT entries may be disabled by CICS/VS cr by the user-written PEP. Through the use of the CICS/VS message switching transaction (CMSG - see Chapter 3), terminal operators may be notified by the master terminal operator to use another transaction code, which will initiate an earlier version of the prcgram in error (see Figure 8-3).

This technique requires terminal operators to change their operating procedures. It may intrcduce terminal operating procedure difficulties, as it requires the terminal cperator to utilize that alternative transaction code until subsequently notified by the master terminal operator that the original transacticn code can be used once more.

This notification would be made when the program in error has been corrected, recataloged tc the CICS/VS program library, and the PPT updated by the CICS/VS-provided master terminal cperator transaction (CSMT) to point to the ccrrected program. (See "Online Program Maintenance" in this chapter.) Fcllcwing this, the CICS/VS master terminal operator transaction may be utilized to enable the relevant PPT and PCT entries, tc activate the program and transaction code again. A CMSG transaction may then notify terminal operatcrs of this fact.

The correction cf the errcr prcgram, and subsequent activation of the original transaction ccde, may take a considerable amount of time. During this time the terminal operatcr must remember to use the alternative transaction code (and transaction format if the backup program demands a different format). If the terminal operator does forget, and instead uses the original transaction code for the program in error, CICS/VS will nctify him that the transaction code and/or program is disabled.

If a new terminal operator later signs onto CICS/VS, he may not receive the master terminal operator message notifying him to use the alternative transaction code. Similarly, a terminal operator may forget to use the alternative code. To allow for these situations, the terminal operating procedures should indicate that if a CICS/VS message is received indicating that a transaction code and/or program is disabled, the terminal operator should refer to a user-provided table in the user's terminal operating procedures documentation, which identifies the alternative transaction code to be utilized.

The above program backup technique does not require any modification of CICS/VS, but does require different terminal operating procedures to be adopted until the error program is corrected. The user must weigh the difficulties which these different operating procedures introduce in his application environment, with the problems involved if the functions carried out by the error program are unavailable to the terminal users.

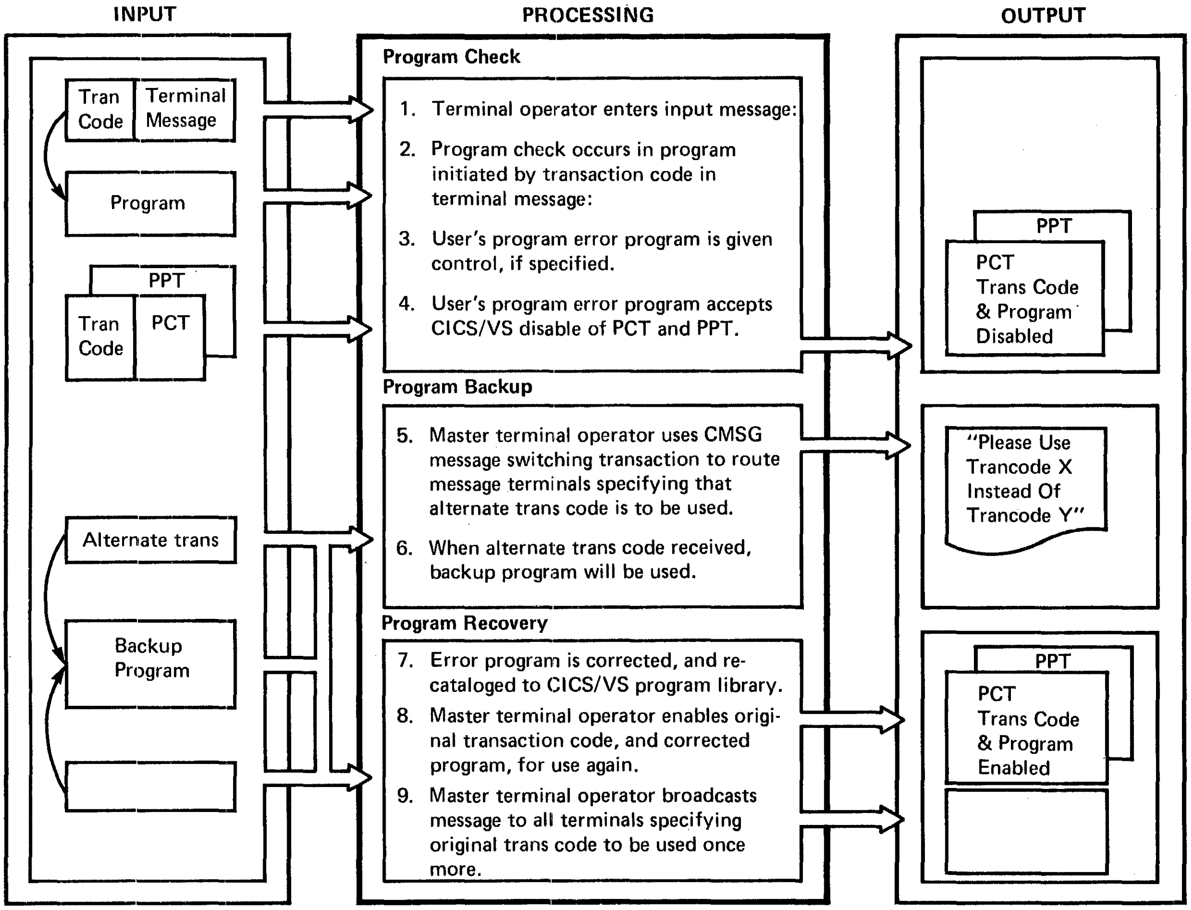

INPUT
PROCESSING
OUTPUT
Tran Code
Terminal Message
Program
PPT
Tran Code
PCT
Alternate trans
Backup Program
Program Check
1. Terminal operator enters input message:
2. Program check occurs in program initiated by transaction code in terminal message:
3. User's program error program is given control, if specified.
4. User's program error program accepts CICS/VS disable of PCT and PPT.
Program Backup
5. Master terminal operator uses CMSG message switching transaction to route message terminals specifying that alternate trans code is to be used.
6. When alternate trans code received, backup program will be used.
Program Recovery
7. Error program is corrected, and re-cataloged to CICS/VS program library.
8. Master terminal operator enables original transaction code, and corrected program, for use again.
9. Master terminal operator broadcasts message to all terminals specifying original trans code to be used once more.
PPT
PCT
Trans Code
& Program
Disabled
"Please Use Trancode X Instead Of Trancode Y"
PPT
PCT
Trans Code
& Program
Enabled

Figure 8-3. Program Backup Technique

DUMP DATA SET

If a program check or program control ABEND occurs, a program dump of all the storage associated with the task in error is automatically produced by CICS/VS. This includes any application programs which have been linked to, or any terminal I/O areas, file I/O areas, file work areas, and other working storage utilized by that task.

These program dumps are directed to one of two dump data sets. Relevant CICS/VS areas such as the CSA (common system area) and TCA (task control area), together with the associated terminal control table entry for that task, are also dumped.

Two dump data sets are utilized by CICS/VS, but only one is in active use at a time. Any program dump, resulting from abnormal termination or use of the CICS/VS dump control macro instructions issued by application programs, is directed to the active dump data set. The master terminal operator can indicate that the second dump data set is to be made the active dump data set, and the first dump data set is to be left inactive. This switching of dump data sets is accomplished to enable dumps previously written to a dump data set to be printed from that data set while CICS/VS execution is in progress. The printing of dumps is achieved through the use of the CICS/VS dump utility program, which may be executed in a batch partition concurrently with CICS/VS. (See the relevant _CICS/VS Operations Guide_ for DOS or OS.)

ONLINE PROGRAM MAINTENANCE

CICS/VS provides support for online program maintenance. The master terminal operator is notified by CICS/VS when an abnormal program termination occurs. If a hard-copy terminal is used for such automatic master terminal operator output, that indication will be immediate, provided the terminal is in service and not presently being used for other terminal I/O. Depending upon the severity of the problem, the master terminal operator may decide to switch dump data sets and run the CICS/VS dump utility program in another partition to print the particular offending dump as soon as possible. That dump can then be passed to maintenance programmers for debugging and correction, while CICS/VS execution continues for other application programs.

When the error is corrected, and the corrected version of the program is compiled and cataloged to the CICS/VS program library, the master terminal operator can alter the processing program table entry for that program to point to its newly link-edited version. This is achieved by the master terminal operator transaction (CSMT) provided by CICS/VS (see _CICS/VS System Administrator's Guide_). Program maintenance and correction can therefore proceed together with CICS/VS execution. Following correction of the program in error, the relevant PPT and PCT entries may then be enabled by the master terminal operator, to allow the corrected program and transaction code to be put into active use again. (See "Program Backup.")

KEYPOINTING OF CICS/VS

The function of "keypointing" is to collect selected system information to be used in a subsequent restart. CICS/VS provides two types of keypoint:

- Warm keypoint

- Activity keypoint

These functions are described in the following paragraphs, together with two other recovery facilities:

- Logical task synchronization

- Protected resources

SYSTEM WARM KEYPOINTS

A system warm keypoint is written by the CICS/VS-provided keypoint program as part of controlled shutdown, when all system activity has been quiesced. It is used during a warm start of CICS to restore the operating environment following a controlled shutdown. The following information is recorded as part of a warm keypoint:

- FCT - File status (disabled, enabled, closed, read-only)
- TCT - Terminal and line status and negative poll delay (for nonswitched terminals)
- DCT - Intrapartition destination status
- TST - Temporary storage control blocks and data set bit map
- ATP - Asynchronous transaction processing control blocks
- ICP - Interval control outstanding requests
- PPT/PCT - Disabled entries
- CSA - Information in the common system area; for example, storage cushion size, ICV, ICVS, ICVR, and MAXTASK

The above information is written to the CICS/VS restart data set just prior to a controlled shutdown. Since the system may be inoperable, this function is performed without the use of CICS/VS macro instructions. The addresses of the data are also checked for validity.

SYSTEM ACTIVITY KEYPOINTS

The purpose of system activity keypoints is to indicate to CICS/VS, during an emergency restart, the user tasks active at the time of the keypoint, the status of transient data intrapartition destinations, temporary storage data identifications, and terminal control table status for VTAM-supported terminals. This is used to minimize the amount of emergency restart activity necessary.

A system activity keypoint is written periodically by CICS/VS during normal operation, to record on the system log data set the processing activity status of CICS/VS and user tasks at the time of the keypoint. The frequency of recording system activity keypoints is a function of the number of output operations to the system log. This frequency may be specified by the user either at CICS/VS system generation or at system initialization, and may also be dynamically changed during online operation through the use of the master terminal operator transaction (CSMT). The frequency of the activity keypoint, and the amount of journaling performed by active tasks, determines the amount of data on the system log to be processed when CICS/VS is restarted. The amount of this data will influence the duration of the CICS/VS emergency restart.

The information recorded by the system activity keypoint comprises the following CICS/VS tables and control blocks:

- TCAs - status of tasks that have at least logged one record or have completed a LUW by issuing a sync point
- DCT - status of transient data intrapartition destinations
- TST - status of temporary storage unit table entries

- TCT - identification of terminals waiting for response to committed output messages (see later)

The activity keypoint function is initiated by the transaction CSKP which is attached periodically by the journal control program (JCP). This transaction transmits system data to the system log and then issues a conditional link to a user activity keypoint program, DFHUAKP. The purpose of this facility is to allow the user to insert application-dependent information in the system activity keypoint. In order to operate properly, the program must be resident and should use only storage control and journal control functions. Journal operations should be asynchronous without start I/O. Synchronization is provided by an end-of-keypoint synchronous record, written by the activity keypoint program at the end of keypoint processing. The user should ensure that the keypoint transaction has higher priority over other tasks that are eligible to log data.

During an emergency restart, the recovery utility program (RUP) copies to the restart data set only that data which has been output by in-flight tasks (tasks that were still active at system termination time). In order to force RUP to copy user activity keypoint data to the restart data set, the user must provide a journal record identifier with the high-order bit ON in the record ID field of the user keypoint data record. Refer to the CICS/VS System Programmer's Reference Manual for more detail. The user keypoint should be used to keypoint only limited amounts of data, for example, selected user data or tables.

LOGICAL TASK SYNCHRONIZATION

Logical task synchronization defines the completion of a logical unit of work (and the intent to begin another). It enables the user to split a long duration task into logical units of work of short duration which better fit the task's recovery requirements. A logical unit of work (LUW) is a user unit of work which performs a complete processing function. One task may perform a complete LUW, or several LUWs may be performed, as identified by the user's application program. The completion of a logical unit of work is referred to as a "sync point" (synchronization point), and may be explicitly defined by the user task (see CICS/VS Application Programmer's Reference Manual). A sync point is generally implicitly defined to be at the completion of processing of an entire task. The completion of a logical unit of work indicates to CICS/VS that:

- All updates or modifications performed by the task up to that point in time are logically complete, and should not be backed out in the event of a subsequent system failure.

- Functions requested prior to the synchronization point, but deferred until the end of the logical unit of work, should be initiated. An example of this is a transient data track release function which is performed at end of LUW, in the case of recoverable intrapartition destinations.

- All resources which were protected by the task up to this point will be released. Such a resource may be a transient data intrapartition destination which is logically associated with a task (see "Protected Resources," following).

The location of a sync point for a task on the system log, relative to other journaled activity for that task, determines the extent to which CICS/VS may need to provide transaction backout.

Transient data intrapartition data set activity of tasks which had not reached logical task synchronization (a sync point), at the time of an uncontrolled shutdown may be backed out by CICS/VS to the previous sync point for that task (or to the start of the task) on emergency restart.

Sync points are also used by CICS/VS to delimit the extent to which user data set modifications may need to be backed out for tasks which had not completed a logical unit of work at the time of an uncontrolled shutdown. CICS/VS collects all user data set modifications during emergency restart, only for tasks which had not completed a logical unit of work at the time of an uncontrolled shutdown. These data set modifications are copied by CICS/VS to the restart data set during emergency restart. They are read by the CICS/VS transaction backout program to back out the data set modifications made by tasks which failed to complete a logical unit of work. This is discussed in more detail in "User Data Set Backout" later in this chapter.

PROTECTED RESOURCES

Consider the following example of two tasks which update the same record in a data set. Task A gains exclusive control of the record, updates it, and releases exclusive control. Then task B gains exclusive control, updates the record, and completes normally. However, task A could not complete normally because of system termination. Consequently, the effect of the partially completed task A must be backed out on restart. When task A's update is backed out, the update of task B will also be backed out, erroneously.

To avoid this, CICS/VS provides a protection facility to enqueue a task on specific records which are being updated, deleted, or added using "protected" data sets (defined in the FCT entry LOG=YES). The enqueue applies until the end of a logical unit of work, and protects records from subsequent modification activity by other tasks, until it is determined that the logical unit of work is completed.

When task A issues a file control GET for update, CICS/VS enqueues explicitly on that record. It then reads the record from the data set and logs the input record to the system log. When the task issues the subsequent PUT, to update the record, CICS/VS does not dequeue its use of that record until task A indicates completion of a logical unit of work. CICS/VS then dequeues the task from the data set record.

In the meantime, if task B wishes to update the same record, when CICS/VS enqueues on that record, the task is placed on the suspended task chain by CICS/VS until task A completes and dequeues from the record. Task B then gains control of the record and carries out its update. If the system terminates before the completion of task A, the task A update can be backed out without affecting other processing of the record because task A had exclusive control of the record of termination.

Enqueuing on the record for the duration of the task therefore protects the record from possible loss of an update made by a completed task, because of backout of the update of a partially completed task.

The same procedures also apply to additions and deletions.

Similar protection is carried out by CICS/VS for transient data intrapartition destinations. Only one task at a time is permitted to access an intrapartition destination for input and one is permitted to access it for output. The destination is regarded by CICS/VS as two separate resources--one input resource, and one output resource.

By serializing updates as described above, the integrity of the data set following backout on restart is protected. However, this may have some effect on performance if several tasks are operating on the same resource at some time during their combined total period of execution. This effect on performance must be evaluated in terms of improved integrity.

The user should recognize the possibility of a lockout occurring if several tasks attempt to update two or more records concurrently in a protected data set (FCT LOG=YES), and the records are not accessed in the same sequence by each task. This is illustrated in the following example.

Consider an order entry application, which accepts orders for several products in the same CICS/VS transaction. This transaction updates the stock availability for each product in a product data set, that specifies automatic logging in its FCT entry (LOG=YES). If one terminal enters a transaction for orders against product numbers 638, 815, and 1068, the application program will update those product records. CICS/VS file control will enqueue against each record until the task passes through a user-synchronization point, or terminates. If another terminal also enters a transaction at the same time for orders against product numbers 501, 1068, and 815, an enqueue interlock will occur on product numbers 815 and 1068, and neither task will terminate. It will appear to the operators of these terminals that the system has gone down, while in fact it is still processing other terminal transactions successfully.

To avoid this automatic logging enqueue interlock, three actions are possible:

1. The terminal operator should always enter product numbers in the same sequence (such as ascending sequence).

2. The application program first sorts the input transaction contents so that product numbers are ascending, before processing begins.

3. The application program issues a user sync point macro instruction after processing each product order in the transaction.

The first solution requires special terminal operator action which may not be practical within the constraints of the application. (For example, orders may be taken by telephone in random product number sequence.)

The second solution requires additional application programming, but imposes no external constraints on the terminal operator or application.

The third solution requires less additional programming than the second solution. However, by issuing a user sync point, it implies that previously processed product orders in the transaction are not to be backed out on emergency restart if a system failure occurs before the processing of the entire transaction is completed. This may not be valid for the application, and raises the question on emergency restart of which products in the transaction were processed (orders accepted) and which were backed out by CICS/VS. If the entire transaction must be backed out, either a user sync point should not be issued, or only one product order should be entered in each CICS/VS transaction.

Of the three solutions, the second solution (sorting product numbers into ascending sequence by programming) is most widely accepted.

The possibility of an automatic logging enqueue interlock occurring exists for any application which processes several application-oriented logical units of work (product orders), within one CICS/VS logical unit of work.

PROTECTED MESSAGES

VTAM-supported terminals, such as the 3600, enable message recovery and resynchronization to be attempted during emergency restart of CICS/VS. The user can specify in the PCT that various transaction codes are to be "protected."

Message Recovery and Resynchronization

If message-protected transaction codes are used with VTAM-supported terminals, the first input message during a logical unit of work associated with a task is logged to the system log. If uncontrolled shutdown occurs, the input message for each in-flight LUW is identified during emergency restart and transferred to temporary storage. It can be retrieved from temporary storage by user programs based on the identification of the terminal which entered the message. The in-flight activity associated with each task is backed out during emergency restart. The user may then initiate reprocessing of that input message. (See "Transaction Recovery and Restart" later in this chapter.)

The last output message transmitted by message-protected tasks without a WAIT, are regarded as "committed output" messages. The output message is logged, and the message is transmitted by VTAM together with a request for a positive response from the VTAM terminal on receipt of the message. When the positive response is received by VTAM, it notifies CICS/VS, which logs the receipt of the positive response to the system log.

If an uncontrolled shutdown occurs before the positive response is logged by CICS/VS, it can be detected by the CICS/VS recovery utility program during emergency restart. The output message is transferred to temporary storage, from which it can be retrieved by user programs based on the identification of the terminal designated to receive the output. The task which generated the output message may have terminated normally prior to uncontrolled shutdown, but the terminal may not have received that committed output message. On emergency restart 3600 logical units, CICS/VS retransmits this output message with a request for positive response, to ensure its receipt by the terminal.

CICS/VS uses the VTAM sequence numbers, which are allocated to each input and output message associated with a logical unit, to establish message resynchronization with programmable controllers during emergency restart. These are obtained from the system log by the CICS/VS recovery utility program.

Inquiry transactions which do not modify data sets should not be specified in the PCT as protected. Logging of input or output messages will not occur. Such transactions can be reprocessed on emergency restart, if necessary.

Deferred Output Integrity

An uncontrolled shutdown may occur during the processing of an LUW belonging to a protected task. Such an in-flight LUW will be backed out on emergency restart by CICS/VS. If a committed output message is transmitted to a terminal when first requested by the application program, it may reach the terminal even if an uncontrolled shutdown cccurs before the prctected task terminates normally. The contents of the output message may indicate tc the terminal that processing of the task was ccmpleted befcre unccntrolled shutdown. However, if the task was in-flight, it will be backed cut on emergency restart. The task must then be reprocessed fcllowing emergency restart. The terminal operator is not aware that the task was in-flight and was backed out, and would not normally reenter it cn emergency restart. The result is the loss of that task's prccessing.

To avoid this possibility, CICS/VS defers transmission of the last output message until completion cf the task's current logical unit of work. Thus, the output will be transmitted cnly if the LUW completes normally. In the event of uncontrolled shutdown before completion of the LUW, the in-flight LUW is backed out on emergency restart, and the remote programmable ccntroller (cr the terminal cperator) can retrieve the original input message frcm temporary storage and resubmit that input message for reprocessing. If the LUW completed, and output was initiated, but uncontrolled shutdcwn occurred before a pcsitive response was received from the terminal, the completed LUW is not backed out on emergency restart. The ccmmitted 3600 output message is retransmitted by CICS/VS on emergency restart as previously described.

The result of this deferred output is improved message integrity. The trade-off is a delay befcre transmission of the output with a possible increase in response time at the terminal. If response time is not to be increased, the user should either request a user sync point, or terminate the task as soon as possible after requesting output.

Terminal control and BMS both permit the user to request immediate initiation of terminal I/O fcr VTAM-supported terminals. (See "Terminal Control Using VTAM" in Chapter 2.) This avoids the deferred output delay, but the possibility is increased that the terminal operator may receive output indicating completicn of tasks which are subsequently backed out.

An alternative approach is to break the output into two or more sections. The program can request output of the first section, specifying either immediate cutput or the default cf delayed output. When output of the next section is requested, the first section is transmitted if it indicated delayed output. This continues until either a sync pcint is reached, or the task terminates when the last section of output is considered a committed cutput message and is handled as previously described. The partial sections cf output may satisfy the requirement for rapid respcnse time, with each section indicating that further cutput is following. Only receipt of the last section of output is an indication tc the terminal operator of completed task execution. The disadvantage of this approach, however, may be increased CICS/VS and VTAM overheads.

The use of BMS terminal paging intrcduces another consideration. If the terminal which initiated a protected task is in transaction and request page status, terminal pages are written to temporary storage for subsequent display, but are not directed to the terminal. The application program shculd send a ccmmitted output message to the terminal indicating completicn of the task, and availability of the terminal pages through the terminal paging commands. (See _CICS/VS Terminal Operator's Guide_, SH20-9005.) If the terminal is in TRANSCEIVE status, the first page will be transmitted tc it automatically when the protected task completes.

CICS/VS TERMINATION

The shutdown of CICS/VS may be either a controlled shutdown or an uncontrolled shutdown (such as following a machine check or power failure).

CONTROLLED SHUTDOWN

A controlled shutdown preserves certain vital information about the CICS/VS environment during normal or abnormal termination of CICS/VS. This information, recorded on the restart data set, may be subsequently used to warm start CICS/VS and reinitialize it to its status at termination. The user may, optionally, elect to warm start only certain parts of CICS/VS, and allow a cold start (complete reinitialization) of other parts of CICS/VS.

Two CICS/VS tables are used to facilitate the functions of controlled shutdown and warm start:

- Transaction list table (XLT)
- Program list table (PLT)

Transaction List Table (XLT)

On CICS/VS controlled shutdown, a transaction list table (XLT) may be loaded. This table identifies a list of transaction codes accepted by CICS/VS during termination. All other transaction codes will be rejected by CICS/VS.

Program List Table (PLT)

The program list table (PLT) is a list of programs to be executed either during controlled shutdown or during system initialization. It is generated by the user, who generally specifies two tables. One PLT identifies various user-written programs which are to be executed during either the first or second stage of CICS/VS controlled shutdown (see below). These user programs may record application-dependent information, which will permit user recovery of that information on subsequent system initialization.

Another PLT can be used to identify various user-written programs which are to be executed during the post-initialization phase of CICS/VS system initialization. These user programs may locate the application-dependent information written by PLT-identified programs during CICS/VS controlled shutdown, and use that information to reestablish the online applications as required by the user. Several other uses of the PLT are described later in this chapter.

A normal controlled shutdown causes the status of various areas to be written to the restart data set to permit a warm start to take place. It uses the program list table (PLT), as described above, and carries out termination in two stages: the "first quiesce stage" and the "second quiesce stage." During the first quiesce stage, terminals are still active, but they are only permitted to enter transactions defined in the transaction list table (XLT). Programs defined in the first section of the PLT are also executed. During the second quiesce stage, terminals are deactivated and programs defined in the second section of the PLT are executed. The following paragraphs further describe the purpose of these two stages of termination.

- FIRST QUIESCE STAGE OF TERMINATION

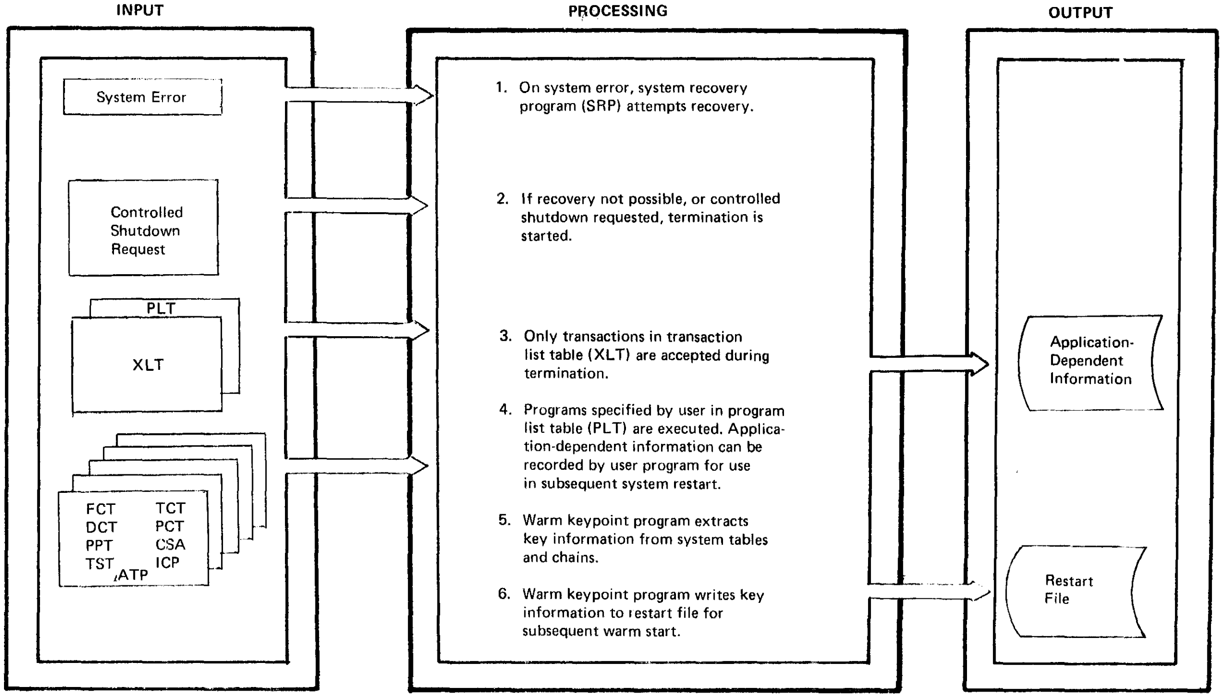
  During the first quiesce stage of termination, a group of user-written programs may be sequentially executed. These programs perform special operations that are unique to the installation. All CICS/VS facilities are available to the programs during this stage. The programs to be linked to are defined in the program list table that is loaded during system termination.

  In addition, only those transactions defined in the transaction list table are accepted from terminals. Existing tasks, tasks to be automatically initiated, or ATP batches in process are allowed to continue unhampered to their normal conclusion (see Figure 8-4).

- SECOND QUIESCE STAGE OF TERMINATION

  At a user-defined point, termination activity waits until all system activity stops. Termination then continues in the second quiesce stage without accepting any further terminal transactions.

  When all program list table programs defined to execute in the second quiesce stage have been executed, the warm keypoint is taken and CICS/VS terminates further execution (see Figure 8-4).



1. On system error, system recovery program (SRP) attempts recovery.
2. If recovery not possible, or controlled shutdown requested, termination is started.
3. Only transactions in transaction list table (XLT) are accepted during termination.
4. Programs specified by user in program list table (PLT) are executed. Application-dependent information can be recorded by user program for use in subsequent system restart.
5. Warm keypoint program extracts key information from system tables and chains.
6. Warm keypoint program writes key information to restart file for subsequent warm start.

Figure 8-4. CICS/VS Controlled Shutdown

ABNORMAL TERMINATION

An abnormal termination of CICS/VS may occur if an application program destroys part of the CICS/VS nucleus or key system information maintained in dynamic storage. This is generally the result of an application program error, and may cause an ABEND of the CICS/VS partition/region. For this reason, the CICS/VS system initialization program issues an OS/VS STAE (or DOS/VS STXIT) macro instruction to enable the system recovery program to regain control in the event of a CICS/VS ABEND. If an abnormal termination subsequently occurs, the system recovery program (SRP) attempts to either correct or circumvent the problem (in the case of CICS/OS/VS) to continue operation (see "System Recovery Program" in this chapter). If recovery is not possible, or advisable, a controlled shutdown will be initiated by the SRP to quiesce other CICS/VS system and task activity, and record the environment status of the CICS/VS system on the restart data set for subsequent warm start (see Figure 8-4).

Even though a controlled shutdown may be taken before the CICS/VS partition/region ABEND completes, executing tasks may not be able to terminate normally. These in-flight tasks may need to be backed out by an emergency restart of CICS/VS, such as required to restart after an uncontrolled shutdown.

If the SRP is unable to correct or circumvent the problem resulting in abnormal termination, and is further unable to initiate a controlled shutdown (such as in the event of destruction of part of the controlled shutdown routines), an uncontrolled shutdown will occur.

UNCONTROLLED SHUTDOWN

An uncontrolled shutdown of CICS/VS can basically result from four different causes:

- Power failure

- Machine check

- Operating system WAIT/ABEND

- Partition/region ABEND

In each case, termination of system operation is either immediate, or so shortly after appearance of the cause that insufficient time, CPU resources, or system facilties are available to permit CICS/VS to complete a controlled shutdown.

System or user tasks may still be active at the time of termination, as CICS/VS is unable to quiesce system activity. Consequently, a subsequent warm start of CICS/VS is impossible. Instead, an emergency restart must be carried out.

Recognizing the possibility of any of the above failure situations occurring at any time, CICS/VS periodically records system activity keypoints (described earlier in this chapter). Synchronization point records are also written during the execution of tasks, at the completion of each logical unit of work.

This information records, on the system log data set, the dynamic activity of CICS/VS, and of active tasks, and permits an emergency restart to be performed at some later time using the system log. The use of this information to restore CICS/VS to its status at the time of the uncontrolled shutdown and to back out the processing of tasks which had not completed a logical unit of work, is discussed in "Emergency Restart" later in this chapter.

CICS/VS INITIALIZATION

CICS/VS can be initialized either:

- With a complete cold start
- With a complete warm start
- With a partial warm start
- With an emergency restart

COMPLETE COLD START

A complete cold start results in complete reinitialization of CICS/VS and system data sets to their status as specified at system generation, withcut regard for any previcus system activity. The system initialization table (SIT) is used tc specify the particular versions of the different CICS/VS system prcgrams and tables that are tc be utilized for CICS/VS initialization.

COMPLETE WARM START

A complete warm start reintializes CICS/VS to the status that existed at the previous ccntrolled shutdown -- all system activity having been quiesced normally prior to shutdcwn. All CICS/VS system programs and tables are first ccld started using the SIT as described previously. If the SIT indicates that a complete warm start is to be performed, all CICS/VS system tables are then reestablished to their status as at controlled shutdown, using the warm keypoint infcrmation written to the restart data set at that time.

PARTIAL WARM START

A partial warm start is similar tc a complete warm start, except that only selected CICS/VS system tables are warm started, as specified in the SIT. Information is ctained from the warm keypoint written at ccntrolled shutdcwn, cnly for thcse tables specified to be warm started. The remaining tables are cold started. An example requiring a partial warm start is an applicaticn which requires a warm start of the DCT so that data queued to intrapartiticn destinations prior to a controlled shutdown may be retrieved cn restart. The FCT and TCT may alsc need to be warm started tc reestablish file and terminal status as at controlled shutdown. The application may, hcwever, require a cold start of temporary storage. Figure 8-5 illustrates a CICS/VS warm start.

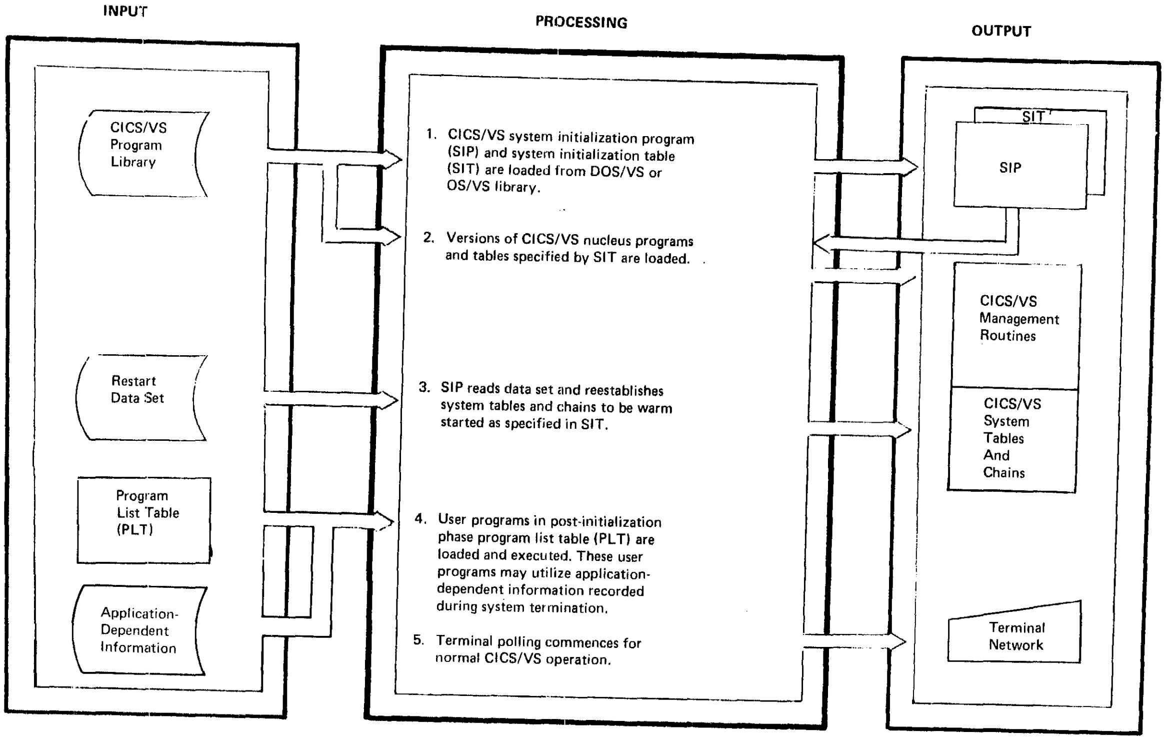

INPUT
PROCESSING
OUTPUT
CICS/VS Program Library
Restart Data Set
Program List Table (PLT)
Application-Dependent Information
1. CICS/VS system initialization program (SIP) and system initialization table (SIT) are loaded from DOS/VS or OS/VS library.
2. Versions of CICS/VS nucleus programs and tables specified by SIT are loaded.
3. SIP reads data set and reestablishes system tables and chains to be warm started as specified in SIT.
4. User programs in post-initialization phase program list table (PLT) are loaded and executed. These user programs may utilize application-dependent information recorded during system termination.
5. Terminal polling commences for normal CICS/VS operation.
SIT
SIP
CICS/VS Management Routines
CICS/VS System Tables And Chains
Terminal Network

Figure 8-5. CICS/VS Warm Start Procedure

After all items have been initialized and control is about to be given to CICS/VS, the group of user-written programs specified in the program list table is sequentially executed. This is referred to as the "post-initialization" phase. These programs perform application-dependent functions, for the recovery of application-dependent information recorded by the user on termination, prior to complete restart of CICS/VS. All CICS/VS facilities are available except for direct terminal communication. Following post-initialization execution of programs in the program list table, the terminal control program is activated to enable terminal transactions to be received and processed.

EMERGENCY RESTART

An emergency restart restores certain CICS/VS facilities to a predefined point which existed prior to an uncontrolled shutdown. Information describing all changes, modifications, and updates made to various system tables and to user data sets during previous CICS/VS execution is recorded on the system log data set. Another data set, the restart data set, is created during emergency restart. (Emergency restart is not supported by the subset option of CICS/DOS/VS.)

The system log contains all changes made to recoverable file control data sets, recoverable transient data intrapartition destinations, and temporary storage protected destinations. It also contains input and output messages for message-protected tasks executed by VTAM terminals.

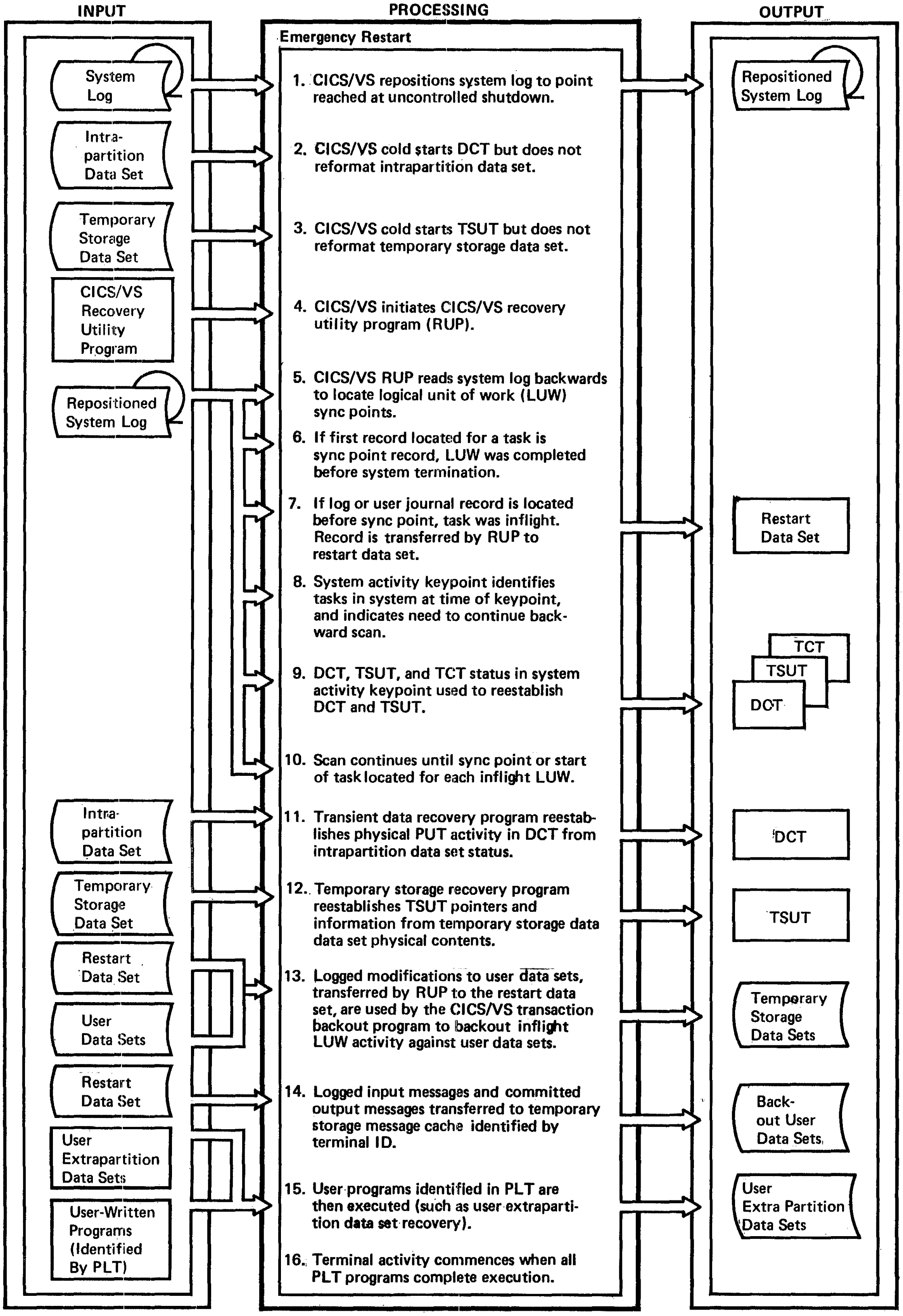
The restart data set is created during emergency restart, and contains system log activity and user journal records for those tasks whose processing activity had not reached a logical completion point when the uncontrolled shutdown occurred. (Such tasks are referred to as in-flight tasks in the following discussion of emergency restart.) This infcrmation can be utilized by the CICS/VS transaction backout program, for example, to remcve the effect of data set modification by in-flight tasks.

On an emergency restart, the CICS/VS system initialization program (SIP) carries out a number of steps to restore CICS/VS operation to a predefined point prior tc unccntrolled shutdown.

These steps are carried out by CICS/VS-provided routines. The user may wish to extend the functions carried out in emergency restart by the addition of user-written programs. One example of such an extension to emergency restart is the execution cf a user-written extrapartition data set backout program to repositicn extrapartition data sets to the point reached on unccntrclled shutdown. To permit such user-written programs to be utilized, CICS/VS identifies in-flight tasks which may require user backout, and copies from the system log, in backward sequence, all system-logged records, and user journal records, for those in-flight tasks to the restart data set.

The design of such a user-written extrapartition recovery program is described in "Extrapartition Data Set Reccvery." Tc utilize the suggested design technique effectively, it is important that the system designer have an appreciation of the steps carried out by CICS/VS during emergency restart.

The remaining topics in this section provide an overview of the CICS/VS emergency restart prccedure. This procedure is illustrated in Figure 8-6. Additional tcpics in this chapter discuss the information in the fcllowing overview in more detail.



1. CICS/VS repositions system log to point reached at uncontrolled shutdown.
2. CICS/VS cold starts DCT but does not reformat intrapartition data set.
3. CICS/VS cold starts TSUT but does not reformat temporary storage data set.
4. CICS/VS initiates CICS/VS recovery utility program (RUP).
5. CICS/VS RUP reads system log backwards to locate logical unit of work (LUW) sync points.
6. If first record located for a task is sync point record, LUW was completed before system termination.
7. If log or user journal record is located before sync point, task was inflight. Record is transferred by RUP to restart data set.
8. System activity keypoint identifies tasks in system at time of keypoint, and indicates need to continue backward scan.
9. DCT, TSUT, and TCT status in system activity keypoint used to reestablish DCT and TSUT.
10. Scan continues until sync point or start of task located for each inflight LUW.
11. Transient data recovery program reestablishes physical PUT activity in DCT from intrapartition data set status.
12. Temporary storage recovery program reestablishes TSUT pointers and information from temporary storage data data set physical contents.
13. Logged modifications to user data sets, transferred by RUP to the restart data set, are used by the CICS/VS transaction backout program to backout inflight LUW activity against user data sets.
14. Logged input messages and committed output messages transferred to temporary storage message cache identified by terminal ID.
15. User programs identified in PLT are then executed (such as user extrapartition data set recovery).
16. Terminal activity commences when all PLT programs complete execution.

Figure 8-6. CICS/VS Emergency Restart Procedure

1. Reposition System Log Data Set

The system initialization table (SIT) is used by the system initialization program to identify whether the system log is on tape or on disk. If it is on disk, the system log will be repositioned to the point reached at uncontrolled shutdown when it is opened for backward processing (see the "CICS/VS Post-Initialization Processing" step later in this section).

If the system log is on tape, a separate system subtask is initiated to reposition the tape to the point reached at uncontrolled shutdown. This subtask utilizes a CICS/VS-provided tape end-of-file utility program which reads the tape system log forward and locates the last journal record written prior to uncontrolled shutdown, by comparing time stamps in each journal record. The tape end-of-file utility program is executed as a system subtask to permit this tape positioning to be overlapped with the emergency restart processing described in the following steps. Once the tape is repositioned, it will subsequently be read backward to determine system activity at the time of shutdown (see the "CICS/VS Post-Initialization Processing" step later in this section).

2. Transient Data Initialization

The system initialization program performs a normal cold start function for the destination control table (DCT) at this stage, but does not reformat the intrapartition data set. The status of extrapartition data sets is lost following an uncontrolled shutdown. The status of intrapartition destinations will subsequently be recovered for those destinations identified in the DCT as being recoverable. This recovery is carried out by the CICS/VS-provided transient data recovery program (TDRP) which is discussed in the "CICS/VS Recovery Utility Program" step later in this section.

3. Temporary Storage Initialization

The system initialization program performs a normal cold start function for temporary storage but does not reformat the temporary storage data set. Temporary storage data in dynamic storage is lost following an uncontrolled shutdown. A design technique is described later in this chapter for user recovery of temporary storage in dynamic storage.

4. CICS/VS Post-Initialization Processing

The remainder of emergency restart processing is accomplished by the CICS/VS-provided recovery utility program. This is executed as a normal CICS/VS application program, under control of the terminal control program's TCA. (This TCA is used, because normal CICS/VS operation and terminal activity have not been initiated at this time.) Upon completion of the system initialization steps outlined above, and after the system log has been repositioned (if tape) to the point reached on uncontrolled shutdown, control is then passed to the recovery utility program.

5. CICS/VS Recovery Utility Program (RUP)

The recovery utility program (RUP) reads the system log (either on tape or disk) backward, to determine system activity prior to the uncontrolled shutdown. As the log is read backward, task synchronization records (sync points) are located. (See "Logical Task

Synchronization," earlier in this chapter.) These define the normal completicn of a logical unit of work for a task. If any user data set modifications (logged either autcmatically by CICS/VS, or ty the user) or any user journal records are located for a task before a sync point (or its initial log reccrd) is read, this indicates that the task had not completed a logical unit of work at CICS/VS uncontrolled shutdown (that is, was "in-flight"). The data set modifications carried out by an in-flight task may need tc be tacked out to the previcus sync point for that task, cr to the start of the task.

CICS/VS carries out this backcut later using the transaction backout program. The recovery utility program identifies in-flight tasks. It collects data set or user journal records fcr in-flight tasks which were written after the start of the task or after a sync point, and copies them to the restart data set. As the system log is read backward, the restart data set is written forward. The restart data set therefore will reflect in-flight task activity prior to the unccntrolled shutdown. The restart data set can subsequently be read by user-written tackout prcgrams, which are executed as post-initialization programs specified in the prcgram list table (PLT). The user-written backout programs can be automatically initiated by CICS/VS following emergency restart, and before terminal activity starts.

The first record located for a task on the backward scan of the system log may be a sync pcint, indicating normal completion of a logical unit of work. Transacticn backout is then not necessary, and journaled or logged records for that task are not copied to the restart data set. (Note: Records output by ccmpleted tasks are collected to the restart data set only if the user specifies a special journal-type code with the high-crder bit ON.)

A log record located for a task on the backward scan of the system log may have been automatically lcgged by the CICS/VS transient data prcgram for intrapartition destinaticns identified in the DCT as recoveratle. These records are used by the recovery utility programs to reestablish the DCT status for the relevant destination. Refer to "Intrapartition Data Set Reccvery after Uncontrolled Shutdcwn" later in this chapter for further discussicn.

The backward scan continues until the following two conditions occur: (1) a system activity keypoint is reached, and (2) all jcurnal and log records output by in-flight IUWs have been retrieved. The system activity keypoint contains information defining the status of intrapartition and temporary storage destinations, TCAs for in-flight tasks in the CICS/VS system at the time of the keypoint, and TCT entries for VTAM terminals with ccmmitted cutput messages outstanding. The status of intrapartition destinations is used to update the DCT, to back out intrapartition activity carried out by in-flight tasks, and to reflect the activity carried cut ty completed tasks.

The TCAs in the system activity keypoint indicate in-flight tasks at the time of the keypoint. A lcng-running task may have been present when the activity keypcint was taken, but may not have logged any processing activity, or written a sync point between the system activity keypoint and the point when unccntrolled shutdown occurred. Such a task had not completed a logical unit of work at the time of unccntrolled shutdown. The backward scan must therefore be continued until a sync point, or first record logged, for that long-running task is encountered. If the first record located for the task is a sync point, the task completed a logical unit of work and no backout is necessary. If a data set modificaticn log record, or a user journal record is encountered before a sync point, those records are copied to the restart data set.

The TCT identification in the system activity keypoint of terminals which have committed output messages outstanding identifies a need to continue the backward scan until these logged output messages are located. These committed output messages are transferred to the temporary storage message cache for each relevant terminal. (See "Temporary Storage Recovery" later in this chapter.) The TCT information in the system activity keypoint is used to prime the TCT with the VTAM sequence numbers from the last completed LUW from each VTAM terminal for subsequent message resynchronization by CICS/VS with the programmable controller.

Activity keypoints identify the necessity of continuing the backward scan until all in-flight tasks have been accounted for. Without activity keypoints, it would not be possible to identify all in-flight tasks without scanning the entire system log backward to its start.

The recovery utility program (RUP) identifies in-flight task activity, and transfers automatically logged file control activity, and user journaled activity from the system log to the restart data set. It reestablishes the DCT status of recoverable intrapartition destinations using information from the latest system activity keypoint, sync point, or end-of-task records of completed LUWs which had activity against recoverable destinations.

The transient data recovery program (TDRP) then scans physically recoverable destination queues on the intrapartition data set to locate their latest PUT activity prior to uncontrolled shutdown. This is used to establish the PUT pointer in the DCT for each physically recoverable destination.

The temporary storage recovery program (TSRP) uses status information collected by RUP from the latest activity keypoint, sync point, or end of task records of completed LUWs which had exclusive control activity against temporary storage destinations (DATAIDs). This status information reflects the logical status of each destination at uncontrolled shutdown.

Following the above RUP, TDRP, and TSRP functions, a new system log and user journal data sets are then opened by the system initialization program for output.

6. CICS/VS Transaction Backout Program (TBP)

RUP processing identifies in-flight LUWs and collects their automatically logged file control activity and user-journaled (to the log) activity on the restart data set. RUP also identifies user data sets which had in-flight activity transferred to the restart data set. Originating input messages and unresponded output messages for in-flight message-protected tasks are also written to the restart data set.

The CICS/VS transaction backout program (TBP) is executed after completion of RUP processing, and after a new system log and user journal data sets have been opened for output. The purpose of TBP is to back out all in-flight activity against user data sets based on information read from the restart data set.

For messages, TBP places originating input messages and committed output messages in a temporary storage message "cache", and primes TCTTEs with the VTAM sequence numbers to be used for reestablishing message traffic.

TBP provides a number of exits to permit the user to participate in data set backout. A TBP initialization exit enables the user to ignore backout against specific data sets. For example, uncontrolled shutdown may have occurred because of an unrecoverable I/O error against a data set. This data set must be recovered by user programs from a backup copy prior to emergency restart. Consequently, these data sets should not have backout activity during emergency restart.

An input exit is also provided. This is given control each time a record has been read from the restart data set and the user may choose to ignore specific records.

An error exit is also given control if an error condition is returned by file control when TBP attempts to back out data set activity. The user may specify alternative activity to be undertaken. (One instance when this exit is given control is when TBP cannot back out an add due to the file organization.) If so, the user may back out adds against VSAM entry-sequenced data sets or ISAM or DAM data sets by flagging the added record as "logically deleted." TBP then writes this logically deleted record to the relevant data set on return from the input exit. User application programs must subsequently check this flag to identify logically deleted records in the data set. (See "Data Set Backout" later in this chapter.)

The result of TBP is to back out all in-flight task activity against VSAM key-sequenced and entry-sequenced data sets or ISAM and DAM data sets. The functions carried out by TBP are described in more detail in "Data Set Backout" later in this chapter.

7. Message Resynchronization for VTAM Terminals

During TBP processing, input messages for in-flight LUWs or committed output messages are transferred to a temporary storage message cache identified by the relevant terminal associated with the message. Committed output messages, for which positive response had not been received from the terminal, are optionally retransmitted during emergency restart. Input messages for in-flight LUWs can be retrieved from temporary storage by user-written programs for reprocessing (see "Transaction Restart" later in this chapter.)

CICS/VS must establish resynchronization with VTAM terminals on emergency restart. This is done using the VTAM sequence numbers placed by TBP in the TCT. These were established by RUP from information in the latest system activity keypoint, sync point, or end-of-task record for the last, completed, LUW against each VTAM terminal. CICS/VS issues VTAM STSN (set and test sequence number) commands to each VTAM logical unit, notifying each programmable controller of the sequence numbers known by CICS/VS. The programmable controller can compare these sequence numbers with those logged on its own disk to determine whether any messages were lost because of the uncontrolled shutdown. These may either be input messages for protected tasks, which should be retransmitted to CICS/VS by the programmable controller, or may be committed 3600 output messages.

Both CICS/VS and the programmable controllers participate in message resynchronization to ensure that no protected messages are lost because of the uncontrolled shutdown.

8. User Post-Initialization Programs

Following execution of the recovery utility program and the opening of new journals, user-written programs identified in the program list table (PLT) are executed. These programs may carry out application-dependent recovery functions, such as repositioning extrapartition data sets or other user recovery functions.

9. <u>Terminal Control Activation</u>

At this point, emergency restart and user backout are complete. The system initialization program (SIP) then initiates a system activity keypoint. Following this, terminal activity is initiated, and normal CICS/VS operation commences.

10. Controlled Shutdown Follcwing Emergency Restart

If a ccntrolled shutdcwn is then requested by the master terminal operator immediately follcwing emergency restart, the warm keypoint necessary for a ccntrolled shutdcwn is taken, and CICS/VS terminates operation. The system may then be initialized at a later time by a normal warm start.

SYSTEM FAILURE DURING EMERGENCY RESTART

System failure during emergency restart represents cne of the most difficult types cf failures to diagncse and correct. The user must be fully aware of the functions performed during emergency restart, the sequence in which these functions are performed, and the effect that abnormal termination during this operation has on data sets and tables.

Prior to initiating emergency restart, an analysis of the failure which caused the system to abnormally terminate shculd be performed. It is possible that the ccndition which caused the system to ABEND will also cause emergency restart to fail. One example of this could be a physically damaged data set which caused an uncontrolled shutdown, causing the identical failure during emergency restart if the CICS/VS transaction backcut program attempts to back out modifications made to that data set.

If a file ccntrol data set has become physically damaged, a user-provided data set reccvery prcgram(s) must recover the data set prior to the user backout program attempting to back out modifications to this data set. Data set recovery involves restoring the contents of that data set from some previcus copy, and then applying all modifications made tc it since the ccpy was taken. CICS/VS automatic journaling (see "Data Set Journaling" later in this chapter), can be used to keep track of data set mcdifications performed during online execution.

If the transient data intrapartition data set cr temporary storage data set is physically damaged, it will nct be possible for CICS/VS to emergency restart these facilities. CICS/VS reccvery of these facilities is dependent upcn the physical contents of the relevant data set as it existed prior to system failure. Therefcre, if the data content cf the data set has to be restcred because of physical damage, CICS/VS may not be able to successfully reconstruct the DCT or TSUT to reflect the status of the restored data set.

User journaling may be utilized, if required, tc produce an audit log of all system data set activity. This audit lcg can be created on a user journal data set, and utilized by user prcgrams for subsequent reconstruction of all system data sets (such as intrapartiticn or temporary storage) which may have been physically damaged.

CICS/VS emergency restart is nct complete until the CICS/VS transaction backout program has successfully completed, and an activity keypoint has been taken. (Optionally, a controlled shutdcwn could be taken at the completion of user back out, if system execution is to be terminated.) If any failure is encountered prior to this time during emergency restart, this procedure must be followed:

- Determine the cause cf the failure: The cause of the failure of emergency restart must be determined and corrected. If the transient data intrapartition data set is damaged, that intrapartition data set and the DCT must be cold started by CICS/VS. Its contents may subsequently be restored by the user, if required, by post-initialization (PLT) program processing. (This is also true for the temporary storage data set.) If a data set is damaged it must be physically recovered by user data set recovery programs.

- Restart Emergency Restart: The emergency restart procedure is executed again using the original system log as input. The original system log is the tape or disk volume which was being used for output when the original system failure occurred. As this data set is not used for output during emergency restart, its contents are available to reinitiate the emergency restart procedure, and recover CICS/VS to its status prior to abnormal termination.

At the completion of emergency restart, the recovered status of CICS/VS has been recorded on the new system log if system execution is to proceed, or on the system restart data set as a warm keypoint if the system is to be terminated. This status represents the predefined point to which the system is recovered; system table, system data set, and user data set status are all logically synchronized. If restart becomes necessary from this point on, the new system log must be used for restart.

If the system was terminated upon completion of emergency restart, without an intervening system failure, the system restart data set contains the fully recovered CICS/VS status in the form of a warm keypoint. A CICS/VS warm start may be performed using this data to initiate CICS/VS execution with the recovered system status.

DATA BASE RECOVERY

The most significant element in system restart is the recovery of various data bases utilized by CICS/VS application programs. This data base recovery is considered in two sections:

- CICS/VS file control recovery

- DL/I data base recovery

CICS/VS file control recovery and online DL/I data base recovery will now be discussed. Batch DL/I data base recovery is discussed later in this chapter.

CICS/VS FILE CONTROL RECOVERY

File control automatically logs modification activity against protected data sets during normal operation. (See "Journaling" in this chapter.) CICS/VS provides support using the transaction backout program (TBP) during emergency restart to back out the activity of in-flight tasks against those protected data sets.

Since abnormal termination of a task caused by an uncontrolled shutdown may have prevented it from completing its use, and possible modification, of protected data sets, all data set activity by that task must be completely removed, or "backed out" to restore the data set to its status as if the task had never been initiated. Once this backout is complete, the abnormally terminated task may be reprocessed, if necessary, using the transaction restart technique described later in this chapter in "Transaction Recovery."

The recovery procedures for CICS/VS file control data bases increase in complexity depending upon whether data sets are read-only, or whether updates, deletions, and additions are made online.

READ-ONLY DATA SETS

For read-only data sets, no data base recovery is required. Retrieval of infcrmaticn from these data sets dces not result in subsequent updates, deleticns, or additions. The data sets are not changed online, and so dc not need to be restored cn restart.

However, an exception to this is the case when an uncontrolled shutdown may have resulted from an unrecoverable I/O error for a particular read-only data set. In this case, the user may wish to follow the procedures for use of a duplicate backup data set described in "Device Recovery" in this chapter.

UPDATE, DELETION, AND ADDITION TO DATA SETS

CICS/VS provides facilities fcr automatic logging of data set records which are to be updated, deleted from, or added to various file control data sets. This automatic lcgging is an cptional facility specified in the file contrcl table by the user for each required data set and indicates that the data set is to be protected. With automatic logging active for a particular data set, whenever a CICS/VS application program specifies a read-for-update cperation for that data set, the record retrieved cn the read, fcr subsequent update, will be automatically logged (written) to the system log. Similarly, any new records which are to be added, or existing reccrds tc be deleted, will be automatically logged to the specified journal data set. In this way, information is maintained of each record's contents prior to an update, a deleticn, or an additicn of a new record.

This logged information is utilized by the transaction backout program on emergency restart following uncontrolled shutdown, to back out the effects cf tasks which had nct completed at the time of termination. This is discussed more completely in "Data Set Backout" later in this chapter.

JOURNALING

Journaling provides a generalized facility for reporting and reviewing modifications to data bases and other important data sets.

The journal control program is table-driven from information defined by the user in the journal control table (JCT). Application programs may issue journal control macro instructions to request specific journaling activity. (See _CICS/VS Application Programmer's Reference Manual_, SH20-9003.)

Data may be journaled synchronously or asynchronously, thus providing for application processing overlap with journaling operations. In the synchronous mode, the requesting task is put in a WAIT state until the journal write has completed successfully, to guarantee that a journal copy of data exists on auxiliary storage before user processing continues. This mode of operation is similar to the way in which most CICS/VS management modules function: the user task receives control only when the requested operation has been performed.

The asynchronous mode allows the requesting task to retain control. The journal write is not synchronized unless and until the task requests synchronization either directly by use of the appropriate journal control WAIT macro instruction, or indirectly by issuing a synchronous journal request. There is no guarantee that the journal copy of the data is written to auxiliary storage until the user task performs synchronization.

Journal records consist of a system prefix, an optional user prefix, and user data. Information placed in the system prefix includes:

- Task identification (that is, transaction code)
- Task sequence number allocated by CICS/VS to uniquely identify this task
- Terminal identification associated with the task
- Time of day
- Journal data set identification

In addition to that detailed above, the following information is provided by the automatic journaling option of file control.

- Data set identification (internally generated by CICS/VS)
- Record identification supplied by the application program

The user prefix is generally application dependent and is defined by the user. For automatic logging, the user data may comprise the record read from the data set for updating, the record to be deleted, or the record to be added to the data set. It may instead be data supplied by the user task, such as terminal messages.

Specification of Journaling

As indicated previously, automatic logging may be specified by the user in the file control table for each required protected data set. Automatically logged information is used by the TBP to back out data set modification activity initiated by in-flight tasks. The user may also specify additional journaling activity in the FCT. This is called automatic journaling, and it enables the user to request that activity (beyond that logged by file control for backout purposes) also be journaled by file control. This activity may include journaling of records after an update is completed; for example, for user programs to recover a data set following an unrecoverable I/O error based on a previous backup copy of the data set. Automatic journaling activity can be directed to the system log, or to a user journal data set. (Refer to the *CICS/VS System Programmer's Reference Manual*, SH20-9004, for further information on specification of automatic logging and automatic journaling.)

Automatic logging will also be initiated by transient data to record activity against intrapartition destinations defined in the DCT as recoverable. Temporary storage update, add, and delete activity against exclusive control destinations (DATAIDs) will also be automatically logged by temporary storage. Refer to 'Transient Data Recovery" and to "Temporary Storage Recovery" later in this chapter for further information.

Journal requests may be made directly by user tasks, through the use of journal control macro instructions. User journal records issued in this way may comprise data for audit purposes, for example, or may contain data to assist in subsequent application-dependent recovery, such as terminal messages.

Use of Journals at System Initialization

On a warm start of CICS/VS, no data set backout is necessary. All transaction activity was completed and the system was quiesced when the previous controlled shutdown took place.

On an emergency restart, the CICS/VS-provided recovery utility program (RUP) identifies all in-flight tasks at uncontrolled shutdown. It transfers all automatically journaled data set modification journal records (and other user journal records), written to the system log by the in-flight tasks to the restart data set. The CICS/VS transaction backout program backs out protected data set activity initiated by in-flight tasks. The post-initialization phase is then entered.

During this phase of system initialization, user programs identified in the program list table for use during post-initialization are executed. These user programs may issue journal control macro instructions to access user journal data sets, or may read data set journal records from the restart data set.

Journal Requests

The following types of journal requests can be requested during normal execution of CICS/VS by application programs. They are explained in more detail in the *CICS/VS Application Programmer's Reference Manual*, SH20-9003.

WRITE - Create a logical record for the relevant journal data set.
WAIT - Synchronize request.
PUT - Implies WRITE, WAIT.
NOTE - Note current logical record positioning of journal data set.
CHECK - Test return code for any exceptional conditions.
POINT - Position journal to a specified logical journal record.
GETF - Get (forward) next logical journal record.
GETB - Get (backward) next logical journal record. For further discussion of the WRITE, WAIT, PUT, and CHECK journal requests, refer to the *CICS/VS System*

Programmer's Reference Manual for further discussion of the *NOTE*, *POINT*, *GETF* and *GETB* journal requests.

Transaction Journals

The journaling facility of CICS/VS can be utilized by user programs to journal any application-dependent information, which can subsequently be retrieved by user-written post-initialization programs. This journaled information may include terminal transactions. A transaction journal produced in this way provides a record of every terminal transaction received by the system, and may also include every output message sent to terminals. A terminal input message may be written by the user to a unique user journal data set and/or the system log, immediately after a task is given control to start processing the input message. The journaling of the input message should be the first activity carried out by the initial program which operates on that transaction.

The transaction journal may be used for audit and for transaction recovery purposes (see "Transaction Restart" later in this chapter). Transaction journals developed in this way may avoid the need for BTAM terminal operators to retransmit transactions on system restart, if those transactions had been entered completely before system termination.

The first input message received from VTAM terminals for each logical unit of work carried out by message-protected tasks (as specified in the PCT) is automatically logged by the VTAM terminal control program. Similarly, committed output messages are logged. CICS/VS uses these logged messages to establish message recovery and resynchronization with VTAM programmable controllers on emergency restart. Input messages belonging to in-flight LUWs or committed output messages for which a positive response had not been received, are transferred during emergency restart to temporary storage. In-flight input messages in temporary storage can be utilized to resubmit transactions for reprocessing, after their activity has been backed out. The final decision as to whether transactions must be resubmitted on restart depends upon the application requirements.

PREPARATION OF USER JOURNALS

Disk extents that receive journal data set output must be allocated and preformatted prior to their use in CICS/VS execution. A journal format utility program is supplied by CICS/VS to complete this formatting. (See the *CICS/VS Operations Guide*.) Once formatted, extents can be (and are) reused for successive CICS/VS executions. The system operator is notified when an extent is full, to enable the scheduling (concurrently with CICS/VS) of a user-written batch program to copy the journal extent to an archive data set on disk or tape before allowing the extent to be reused, if required by the user. A PAUSE option is provided for the user to prevent reuse of the extent by CICS/VS until the archive copy is completed. More than one journal extent may be specified by the user to permit CICS/VS to make an extent switch when one extent is full, and to continue CICS/VS operation while the full extent is being copied to the archive data set.

Tape journals may utilize either one or two tape drives. If one tape drive is used, CICS/VS directs the system operator when another tape must be mounted, and provides its own tape label management. If two tape drives are used, it automatically switches to the other tape drive and directs the operator to dismount the previously used tape.

The CICS/VS-provided tape end-of-file utility program may be used offline to reposition and close correctly a journal tape after an uncontrolled shutdown. For the tape system log, this function is performed by CICS/VS during an emergency restart. In order to enable the repositioning program to operate properly, the tapes must be formatted (with the CICS/VS-provided tape format utility program) the first time they are used for journaling.

Logged information is utilized by various CICS/VS programs during emergency restart to back out the processing of in-flight tasks.

Journaled information may be utilized during the post-initialization phase on restart by user-written application programs (identified in the program list table). This information may be read either from the particular user journal data set, or from the restart data set following an emergency restart, if it was originally written to the system log before uncontrolled shutdown.

The information included in the system prefix detailed above, and in the journal control data record, can be used to identify:

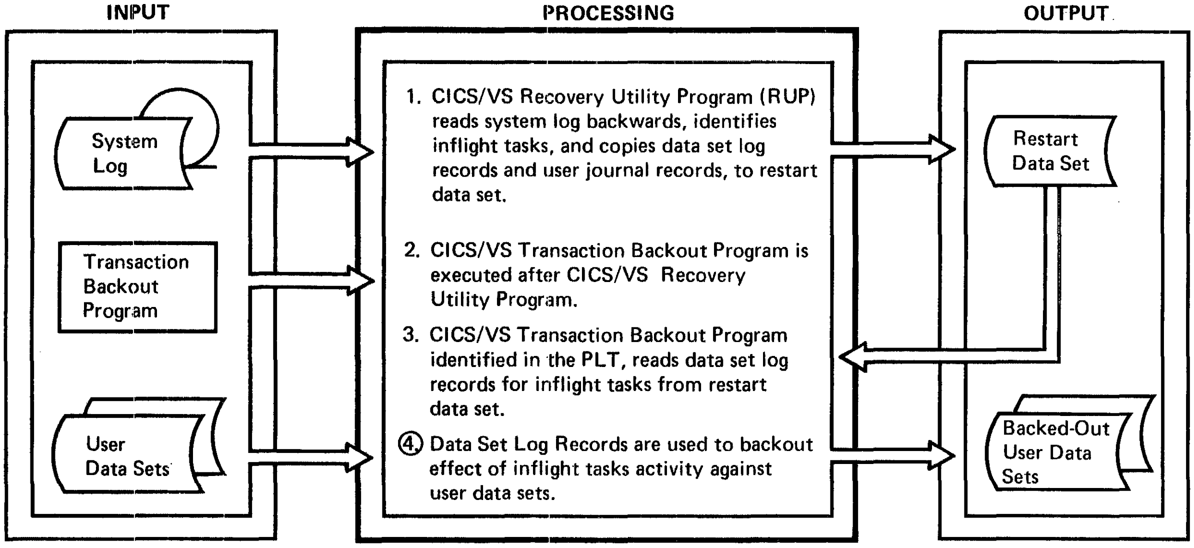
- The transaction identification of the task that logged or journaled the data
- The sequence number of that task allocated internally by CICS/VS
- The identification of the terminal involved
- The time of day that the data set record was written to the journal data set

For automatically logged data sets, the information in the system prefix can be used to identify:

- The original data set to which that record belongs
- The identification of the particular logical record within the data set

DATA SET BACKOUT

Data set backout during emergency restart is achieved by the CICS/VS transaction backout program, which reads data set log records for in-flight tasks from the restart data set, where they were written by the CICS/VS-provided recovery utility program as described previously. The transaction backout program is executed during the post-initialization phase of emergency restart, and backs out the effect of these in-flight tasks. See Figure 8-7 for a description of the transaction backout program.



1. CICS/VS Recovery Utility Program (RUP) reads system log backwards, identifies inflight tasks, and copies data set log records and user journal records, to restart data set.
2. CICS/VS Transaction Backout Program is executed after CICS/VS Recovery Utility Program.
3. CICS/VS Transaction Backout Program identified in the PLT, reads data set log records for inflight tasks from restart data set.

④ Data Set Log Records are used to backout effect of inflight tasks activity against user data sets.

Extended Description

④ The "Before" Log Record of an update replaces the original record in the data set. A "Before Deletion" record for a key sequenced VSAM data set is added back to the data set. An addition to a key sequenced VSAM data set is physically deleted. An addition to other access method data sets is logically deleted by the user by turning on a "Delete Flag" in the record. (It is the user's responsibility to subsequently recognize this record as being logically deleted, via this flag.)

Figure 8-7. Transaction Backout Program (User Data Set Backout)

The log records read from the restart data set by TBP are either the record read from the data set prior to update or deletion, or the identification of a new record added to that data set.

For an update, the relevant logical record in the data set is retrieved, and the contents of that updated record are replaced by the original data record contents from the journal record. The effect of the in-flight task prior to system termination is therefore reversed, and the logical record on the data set is returned to its state before the update.

For a deletion (which can only be made to key-sequenced VSAM data sets), the deleted record from the journal must be added to the data set again, by the backout program, to remove the effect of the inflight task.

File control automatically logs the contents of a record to be deleted, before the physical deletion takes place. It appears on the system log (and hence on the restart data set) as if it were a record before an update operation. TBP, therefore, first attempts to replace the data set record by the logged record as if to back out an update. If a "no record found" error indication is returned, TBP recognizes that the record is assciated with a prior delete operation and adds the logged record to back cut the delete operaticn. This apprcach avoids pcssible duplication cf records, which could otherwise cccur under the following conditions:

a. Uncontrolled shutdown befcre delete completed. CICS/VS may have automatically lcgged a record tc be deleted, but an uncontrolled shutdown may have occurred before the physical delete operation cculd be completed.

b. Restart of emergency restart. In the event of a system failure during emergency restart, emergency restart can be run again as previously described. If a deletion is always backed out by addition, records which were added prior tc emergency restart failure are added again when restart is performed again.

The TBP therefore, first checks that the deleted record is present on the data set. If it is, it does not add it again. This approach permits emergency restart to be rerun as often as necessary without possible duplication of records.

If the log record indicates an addition, and the data set accessed is a VSAM key-sequenced data set, a CICS/VS file ccntrol DELETE macro instruction is issued to remove that added record from the data set. However, if the added record was made to an entry-sequenced VSAM data set, a DAM data set, or an ISAM data set, that added record should be logically flagged by the user to indicate that it has been deleted. This can be achieved by user programming developed for use in the error exit of TBP. Such a "logically deleted" reccrd shculd be ignored if it is subsequently retrieved by ncrmal CICS/VS application programs. Each application program which accesses that data set during normal CICS/VS operation must check the lcgical deletion flag set in the error exit of the transaction backout program, and consider that record deleted if it is ON. In this way, the addition of records by in-flight tasks is removed.

CICS/VS applicaticn programs can also achieve the lcgical deletion of a reccrd in a VSAM entry-sequenced data set, or in an ISAM or DAM data set by turning ON the logical delete flag as previously described. The logically deleted record then "updates" the original record on the data set. If the logical deleticn must be backed cut on emergency restart, it must be "logically added" to the data set. This is acheived by turning OFF the delete flag.

However, such a lcgically deleted record appears to CICS/VS as if it is a normal updated reccrd. CICS/VS, during normal operaticn, logs the "before" image of the record prior to its update being written. The before image has the delete flag OFF. During emergency restart, TBP backs out the "update" (that is, logical deletion) using the before-image logged record, which therefore automatically "adds" back the logically deleted reccrd. Thus, no special user backout action is necessary.

Backout processing can take place correctly only if the data set content has not been damaged during the abnormal termination. This could occur, for instance, in the case of a power failure. If the backout cannot take place because cf a damaged data set, the user must recover the data set from a backup ccpy before attempting another restart.

ONLINE DL/I DATA BASE BACKOUT

The facilities described previcusly for CICS/VS file ccntrol backcut are also used, in part, by DL/I when used online with CICS/VS.

DL/I LOGGING USING CICS/VS SYSTEM ICG

DL/I DOS/VS and IMS/VS DL/I (when used online with CICS/DOS/VS and CICS/OS/VS respectively) cptionally direct DL/I lcg activity to the CICS/VS system lcg by issuing journal control requests. Thus all CICS/VS and DL/I log activity appears on the same system log.

A CICS/VS application program schedules itself against a DL/I PSB by issuing a PCB scheduling call. (See the relevant *CICS/VS and DL/I Application Programmer's Reference Manual*.) The program indicates it has finished its use of the PSB by either issuing a DL/I TERM (termination) call, cr by terminating itself.

DL/I logs the scheduling and termination call requests to the CICS/VS system log, identifying the task and the related PSB name. Between the scheduling and terminaticn calls, cther DL/I calls which result in modification of DL/I data bases will initiate log activity. Similarly, CICS/VS activity against prctected resources (data sets or intrapartition or temporary storage destinations) will also be logged. Such log activity will appear on the CICS/VS system log between the DL/I scheduling and termination log records.

DL/I TERMINATION ACTIVITY

A DL/I termination call fcrces all DL/I data base records, which had been modified in DL/I buffers in storage by the relevant task, to be written to their apprcpriate data bases. The termination call indicates the logical completion of DL/I activity by the task, which should nct be backed out in the event of a subsequent uncontrolled shutdown. The termination call indicates the completion of a DL/I logical unit of work. It also is recognized by CICS/VS as the completicn of a logical unit of work for any CICS/VS log activity which preceded the call. Thus it is regarded by CICS/VS as similar to a user synchronization pcint.

Similarly, a user synchronizaticn pcint request issued by the task is regarded as a DL/I termination call request. Thus, a DL/I termination record, and a user sync point record on the CICS/VS system log indicate completion of the current LUW.

DL/I DATA BASE BACKOUT DURING CICS/VS EMERGENCY RESTART

During emergency restart, the recovery utility prcgram scans the CICS/VS system lcg backward. End-cf-task records, sync point records, and DL/I termination records enccuntered on the backward scan identify completed LUWs. CICS/VS and DL/I log activity initiated by in-flight LUWs is collected by RUP as previously discussed. CICS/VS backs out file control data set activity fcr in-flight LUWs using the transaction backout program as described earlier. CICS/VS alsc issues the necessary DL/I calls to back cut in-flight DL/I activity directly against the relevant DL/I data bases during emergency restart. It is not necessary to run the batch DL/I backout utility.

If the uncontrolled shutdcwn was caused by an unrecoverable I/O error on a DL/I data base, that data base must first be recovered prior to emergency restart. This is achieved by running the batch DL/I recovery utilities.

ONLINE DL/I ENTRY DATA BASE BACKOUT TECHNIQUE

DL/I ENTRY permits online replace, delete, and insert operations against HISAM data bases. However, it does not provide a backout utility as does DL/I DOS/VS. The following technique may be considered by users who wish to implement their own backout support for DL/I ENTRY.

DL/I ENTRY HISAM data bases are defined to CICS/VS in the FCT as VSAM key-sequenced data sets. (A VSAM entry-sequenced data set is not also used as it is for DL/I DOS/VS or IMS/VS DL/I HISAM data bases.) These VSAM data sets support update, addition, and physical deletion of logical records.

DL/I ENTRY uses CICS/VS file control macro instructions to carry out I/O against the VSAM key-sequenced data set used to contain a HISAM data base. A DL/I HISAM data base record may comprise one or more VSAM logical records. DL/I ENTRY will translate application program calls to replace, insert, or delete DL/I segments into file control requests to update, add, or delete VSAM key-sequenced logical records.

If HISAM data bases are specified in the FCT as protected VSAM key-sequenced data sets (that is, LOG=YES), file control will automatically log update, add, and delete requests issued by DL/I ENTRY. The CICS/VS application program must issue an explicit user sync point request, at the point where a DL/I termination call is issued, as DL/I ENTRY will not log a termination record. However, the sync point will indicate completion of the logical unit of work. It will also dequeue VSAM logical records which were modified during the LUW. In emergency restart following an uncontrolled shutdown, the CICS/VS recovery utility program will identify in-flight LUWs and will transfer logged data set activity for such in-flight tasks to the restart data set. The CICS/VS transaction backout program will then back out the logged activity against the relevant VSAM key-sequenced data sets. The user can participate in this backout through use of the TBP input exit. (See "Transaction Backout Program" earlier in this chapter.) This backout will restore the HISAM data base to its status prior to the initiation of DL/I activity by in-flight tasks.

The implementation of this technique is a user responsibility. The technique does not represent a commitment by IBM to support backout of online DL/I ENTRY data bases. It may require further investigation by the user to ensure that data base integrity is not compromised. This technique is practical for DL/I ENTRY because it does not support HDAM or HIDAM, and because it uses file control to request I/O.

DL/I ENTRY SEGMENT SCHEDULING

DL/I ENTRY does not use segment intent scheduling, as implemented for DL/I DOS/VS or IMS/VS DL/I when executed online with CICS/VS. Instead, the specification in the FCT of automatic logging for VSAM key-sequenced data sets used by HISAM data bases enables CICS/VS file control protection to be used. (See "Protected Resources" earlier in this chapter.) When a task requests a replace, insert, or delete operation through a DL/I call, the relevant file control I/O request causes automatic logging for backout, and also enqueues the task on the relevant file control logical record. If a concurrently executing task also attempts modification of the same logical record, it will wait until the first task dequeues from the record before it enqueues on the record. This is done automatically when the application program issues the CICS/VS user synchronization point request at the termination of DL/I activity by the task, or when the task terminates. The second task then enqueues on the record and proceeds with its activity.

The result is data integrity at the VSAM logical record level, without the potential performance disadvantage of segment intent scheduling.

TEMPORARY STORAGE RECOVERY

TEMPORARY STORAGE RECOVERY AFTER CONTROLLED SHUTDOWN

CICS/VS supports the warm start of temporary storage records on disk following a controlled shutdown, but does not attempt warm start of temporary storage records in dynamic storage. The following paragraphs discuss data that can be stored in main storage and that which must be stored on auxiliary storage for subsequent warm start following a controlled shutdown.

Main Storage Residence

Temporary storage allocated to main storage is not recoverable in the event of either abnormal or normal termination. Although application procedures can be developed to attempt recovery of this information, main storage generally should be used as a temporary storage medium for data that does not need recovery. Typically, it should be used only for short-term storage of information, such as for transferring information between programs executing under control of the same task. If information is to be passed to programs executed by other tasks (implying longer-term storage of information), auxiliary storage should be used.

Auxiliary Storage Residence

If, during system initialization, temporary storage is to be warm started, the temporary storage warm keypoint (containing, for example, the temporary storage use map and auxiliary storage data identifications) is utilized by CICS/VS to restore the status of temporary storage.

If, prior to controlled system termination, certain data identifications were released or purged, the warm keypoint taken at termination will reflect this action. On warm start, this deleted information will not be used in reestablishing the various temporary storage data identifications.

Terminal Paging Availability

Terminal pages presented by application programs to BMS are written to temporary storage on disk for later retrieval either automatically, or on request by terminal operators. Temporary storage on disk is retained on a warm start, and terminal pages are still available following system restart.

Message Routing Availability

Similarly, messages to be transmitted to terminals using the CICS/VS message routing facility are written to temporary storage on disk and are still available for transmission on warm start of temporary storage.

TEMPORARY STORAGE RECOVERY AFTER UNCONTROLLED SHUTDOWN

Temporary storage is not recovered by CICS/VS following an uncontrolled shutdown. Temporary storage in dynamic storage is lost. However, a technique is presented later in this section for user implementation of temporary storage recovery for dynamic storage.

TEMPORARY STORAGE RECOVERY AFTER UNCONTROLLED SHUTDOWN

Temporary storage (on auxiliary storage) is recovered by CICS/VS during emergency restart following an uncontrolled shutdown. Temporary storage in dynamic storage is lost. However, a technique is presented later in this section for user implementation of temporary storage recovery for dynamic storage. (See "User Recovery of Temporary Storage Using Dynamic Storage.")

Temporary Storage Automatic Logging

Temporary storage automatically logs activity which results in a change to the temporary storage data set. This includes the following operations:

- Output (temporary storage PUT and PUTQ macro instructions)
- Update (PUT or PUTQ with TYPEOPER=REPLACE macro instructions)
- Release (temporary storage RELEASE and PURGE macro instructions)

(Refer to the CICS/VS Application Programmer's Reference Manual, SH20-9003, for a description of these macro instructions.)

When a task accesses a temporary storage destination (DATAID) for any of the above operations, temporary storage enqueues the task on that destination until the end of its current logical task unit of work if the destination has been defined as protected in the temporary storage table (TST). Any other task which attempts to operate on the same destination for other than a nondestructive read request will result in an attempt to enqueue on the same destination, and will wait until the first task completes its LUW and is dequeued from the destination. The second task is then enqueued to carry out its required processing.

Any of these operations represent an implied enqueue issued by temporary storage. Alternatively, the application program may issue an explicit enqueue request by issuing the temporary storage GET or GETQ macro instruction specifying TYPEOPER=EXCL. This may be desirable when the destination has not been separately defined as protected. The program must ensure that no other task has concurrent access to the same destination for the duration of its logical unit of work.

If a record in temporary storage is updated, the old copy of the record is logged to the system log. This is used to back out the effect of that update, if a subsequent uncontrolled shutdown occurs with the current LUW still in-flight.

Automatic logging for output and release operations (PUT, PUTQ, RELEASE, and PURGE) is deferred until the end of the logical unit of work. At this time, temporary storage forces output of the temporary storage VSAM write buffer if it contains changed data for the particular destination (DATAID). The new status of the destination is logged to the system log identifying the total record count for that DATAID. If the DATAID is to be retained, the logical current record count and total record count are updated in the temporary storage tables in dynamic storage to reflect processing activity carried out during the LUW. If the DATAID is to be released, the RELEASE or PURGE operation is carried out at this time. The task is then dequeued from the destination.

If an uncontrolled shutdown occurs prior to completion of the LUW, all activity by the in-flight LUW against the temporary storage destination is backed out on emergency restart.

Temporary Storage Recovery

On emergency restart, the recovery utility program identifies in-flight tasks and collects information defining the logical status of each protected temporary storage destination at the time of the uncontrolled shutdown. RUP links to the temporary storage recovery program (TSRP), which reads the temporary storage data set to collect all data pointers required to reconstruct the temporary storage tables in dynamic storage for protected destinations. Each record on disk is self-describing and contains the DATAID and the record sequence number within a message set (queue). As the data set is read, and the temporary storage tables are reconstructed, the status of each VSAM control interval is determined and used to reconstruct the temporary storage use map.

At the completion of TSRP processing, all recoverable destinations have been restored to their most recently recorded logical status with in-flight LUW activity backed out.

Recovery of temporary storage in this way also restores terminal paging and message routing.

In addition, interval control requests and automatic task initiation requests from transient data (which are recorded on temporary storage by CICS/VS during normal operations, for subsequent recovery) are reestablished on emergency restart.

USER RECOVERY OF TEMPORARY STORAGE USING DYNAMIC STORAGE

The user can specify, at CICS/VS initialization time, that auxiliary storage residence of temporary storage is not required. (See the *CICS/VS System Programmer's Reference Manual*.) Temporary storage requests are then satisfied using dynamic storage only. This may be desirable, for example, in an installation with limited real storage availability that (except for the use of VSAM for temporary storage residence on disk) does not otherwise have a need for VSAM support. In this environment, the user may wish to specify that auxiliary storage residence is not required for temporary storage support, if the potential performance degradation trade-off is acceptable. (See "Virtual Storage Access Method (VSAM)" in Chapter 7.)

The user should also recognize that temporary storage recovery is not supported for dynamic storage residence. Temporary storage recovery in this environment is a user responsibility. The following technique may be used to implement recovery of temporary storage resident in dynamic storage. This technique can be used either for a warm start following a controlled shutdown of CICS/VS, or for an emergency restart following an uncontrolled shutdown. It does not attempt to back out the processing carried out by in-flight logical units of work prior to uncontrolled shutdown. It does recover all temporary storage activity up to the time of shutdown. Use is made of the temporary storage program user exit "before request analysis" (XTYPREQ) to reduce the amount of coding necessary for recovery in the user's CICS/VS application programs. (See *CICS/VS System Programmer's Reference*

Manual, SH20-9004.) Figure 8-8 illustrates this temporary storage recovery technique.

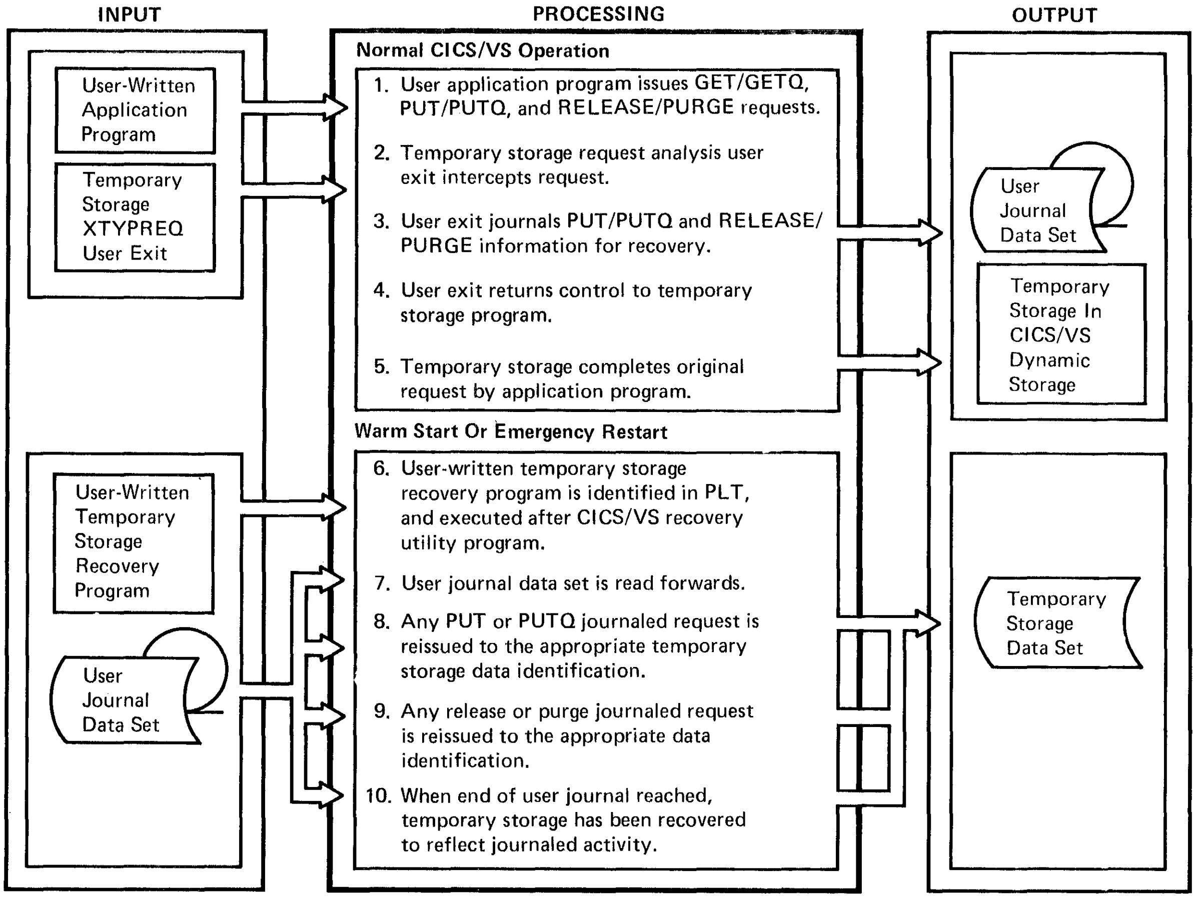

INPUT
PROCESSING
OUTPUT
User-Written Application Program
Temporary Storage XTYPREQ User Exit
Normal CICS/VS Operation
1. User application program issues GET/GETQ, PUT/PUTQ, and RELEASE/PURGE requests.
2. Temporary storage request analysis user exit intercepts request.
3. User exit journals PUT/PUTQ and RELEASE/PURGE information for recovery.
4. User exit returns control to temporary storage program.
5. Temporary storage completes original request by application program.
User Journal Data Set
Temporary Storage In CICS/VS Dynamic Storage
User-Written Temporary Storage Recovery Program
User Journal Data Set
Warm Start Or Emergency Restart
6. User-written temporary storage recovery program is identified in PLT, and executed after CICS/VS recovery utility program.
7. User journal data set is read forwards.
8. Any PUT or PUTQ journaled request is reissued to the appropriate temporary storage data identification.
9. Any release or purge journaled request is reissued to the appropriate data identification.
10. When end of user journal reached, temporary storage has been recovered to reflect journaled activity.
Temporary Storage Data Set

Figure 8-8. Temporary Storage User Recovery for Dynamic Storage Residence

The temporary storage "before request analysis" user exit intercepts requests from programs which direct data to temporary storage and journals that data to a user journal data set. This is done prior to completing the temporary storage PUT, PUTQ, RELEASE, or PURGE request. The user journal record must contain the record to be written by a temporary storage PUT or PUTQ macro instruction, together with the data ID. A request to update an existing temporary storage record must result in the old copy of the record being journaled. A user journal record containing only the data ID must be constructed for a temporary storage RELEASE or PURGE macro instruction.

These user journal records are written to the user journal data set either synchronously using a journal control PUT macro instruction (see "Journal Requests" in this chapter), or asynchronously using a journal control WRITE macro instruction. Synchronous journaling will ensure that a copy of the user journal record exists on the user journal data set before the temporary storage operation is carried out. Asynchronous journaling enables the journal I/O to be overlapped with subsequent application program processing. While asynchronous journaling results in improved performance, it introduces added complexities during subsequent recovery.

This journaling, carried out by user-written code incorporated in the CICS/VS temporary storage program by means of a user exit, has the advantage of not requiring any special coding by the application programmer. Furthermore, it permits records directed to temporary storage (such as terminal pages cr routed messages) by BMS to be journaled for subsequent recovery also. This additicnal user exit code may be easily removed frcm the temporary storage program in the future, if desired.

If a controlled or unccntrolled shutdown subsequently occurs, a warm start or an emergency restart will be carried out to restore CICS/VS to its status at termination. This will cold start temporary storage in dynamic storage, and subsequently execute user-written programs identified in the prcgram list table during the post-initialization program phase.

The user-written temporary stcrage recovery prcgram can be specified in the PLT and executed. This reccvery prcgram reads the user journal data set forward. As each journal record is read, the data ID and temporary storage macro instructicn, issued during previous CICS/VS operation, is issued again tc the same data ID. Any data record in the journal reccrd is written tc temporary storage using a temporary storage PUT or PUTQ macrc instruction as originally issued. Any temporary storage RELEASE or PURGE macro instructicn identified frcm the user journal is alsc issued to the relevant data identification.

The result of this program's execution is that all temporary storage records written, and RELEASEd or PURGEd, during previous system operation are reestablished in the same chronological sequence as prior to CICS/VS shutdown.

TRANSIENT DATA RECOVERY

CICS/VS supports the recovery cf intrapartition destinations, but does nct provide support fcr the reccvery of extrapartition destinations. Such recovery is the responsibility of the user. The procedures for recovery cf the twc types of transient data are discussed in the following.

INTRAPARTITION DATA SET RECOVERY AFTER CONTROLLED SHUTDOWN

On warm start after a ccntrolled shutdown, CICS/VS reestablishes all intrapartition destinaticns and reconstructs the destination control table (DCT) entries to reflect their status at termination. The transient data use bit map (which is used for the allocation of tracks to various intrapartition destinations) is also reestablished. In this way, the status cf intrapartition destinations is automatically recovered on a transient data warm start.

Automatic Task Initiation

The queue count and trigger level of each intrapartition destination are reestablished cn warm start cf transient data. Consequently, automatic task initiation can be utilized on warm start as if termination had not cccurred.

Terminal Output

Because of the reestablishment cf transient data intrapartition destinations on a warm start, and the ability to use automatic task initiation, terminal output directed to transient data intrapartition destinations is autcmatically recovered, and will be transmitted to the relevant terminal as scon as that terminal is in service and able to receive output.

Low (or High) Priority Processing

As discussed in Chapter 3, information received from terminals may be edited, validated, and, if correct, written to transient data intrapartition destinations to enable further processing. This processing may permit the updating of various data sets, for example, to be carried out while the terminal is able to continue entering other transactions.

Information written to intrapartition destinations will be recovered on a warm start. This enables transactions entered from terminals to be quickly validated, and then recorded on disk to allow the processing necessary for those transactions to be carried out on warm start of CICS/VS (if necessary) after controlled shutdown. The terminal operator need not be required to reenter those transactions.

INTRAPARTITION DATA SET RECOVERY AFTER UNCONTROLLED SHUTDOWN

CICS/VS provides recovery of the intrapartition data set during emergency restart following an uncontrolled shutdown. The recovery attributes of each intrapartition destination may be specified by the user in the DCT. The definition of recovery attributes for each intrapartition destination causes CICS/VS to automatically journal to the system log the information necessary to recover each destination queue.

The recovery attributes of an intrapartition destination may specify either physical or logical recovery.

Physical Recovery

When physical recovery is specified for a destination, the following functions are performed during CICS/VS operation:

- The destination entry is keypointed as part of a periodic system activity keypoint.
- The destination entry is automatically logged by the transient data program to the system log on the first transient data PUT macro instruction issued by any task in the system against each physically recoverable destination, and when the destination is PURGEd.
- The destination entry is automatically logged by the transient data program before execution of each transient data GET macro instruction.
- The destination entry is automatically logged, by the transient data program for reusable queues, as a track is released - only if it has been completely read and a new transient data GET macro instruction is issued to the queue (that is, the record will be read from the next track).

During emergency restart, a physically recoverable queue is restored by the CICS/VS transient data recovery program as follows:

- The start of the queue, identified in the DCT, is set to the first track in existence for the queue.
- The location of the next record to be read from the queue (the GET pointer), is set in the DCT to the last record previously read if the task (or LUW) that issued the GET did not complete successfully. It is set to the following record if the task or LUW had completed.

- The location of the next reccrd to be written to the queue (the PUT pointer), is set in the DCT to point after the last physical record present in the queue at emergency restart.

Fcllowing emergency restart, each physically recoverable destination will have been recovered tc reflect all PUT activity. All GET activity to each destination will alsc be recovered, except for the last record which had been read by an in-flight task against these destinations. This record may not have been completely processed prior tc uncontrolled shutdown and must be reread cn emergency restart.

Logical Recovery

Logical recovery results in restoring the GET and PUT pointers to their status at the completicn cf a logical unit cf work (LUW). When logical recovery is specified, reccverable destinations are periodically keypointed in the system activity keypcint, and their status is logged at the end of each LUW which uses each destination (that is, at a sync point or end of task).

This logging is deferred until the end of a LUW, in the event that an unccntrolled shutdown may cccur during the LUW. If this happens, no reccrd of the activity cf the in-flight task will appear on the system log. On subsequent emergency restart, the status of those intrapartition destinaticns cperated upon by in-flight tasks is automatically backed out to the status at the end cf the last LUW which operated on thcse destinations.

If an uncontrclled shutdown dces not occur, at the end of the LUW the status of all intrapartition DCT entries operated on during that LUW is lcgged as part of the sync point for the LUW. If subsequent uncontrolled shutdown occurs, the DCT status will be restored to that at the end of the last LUW fcr each destinaticn, or from the last system activity keypoint.

Protected Destinations

Destinations are protected for the duration of a logical unit of work, to prevent interleaved input and output activity. For protection, each destination is considered tc consist of two separate facilities. This permits one LUW to have exclusive control of the output facility and another LUW tc ccncurrently have exclusive ccntrol of the input facility. There is no conflict between these facilities unless the input LUW attempts to read the data records written by an output LUW while the output LUW is still active. In this case, the input LUW waits until completicn of the output LUW.

On emergency restart, the result cf logical recovery is for CICS/VS to back out, from the queues, the activity of thcse in-flight tasks which had not completed a logical unit of work at system termination.

Following emergency restart, automatic task initiation, terminal output and low or high pricrity processing information on recoverable intrapartition destinations are recovered.

EXTRAPARTITION DATA SET RECOVERY

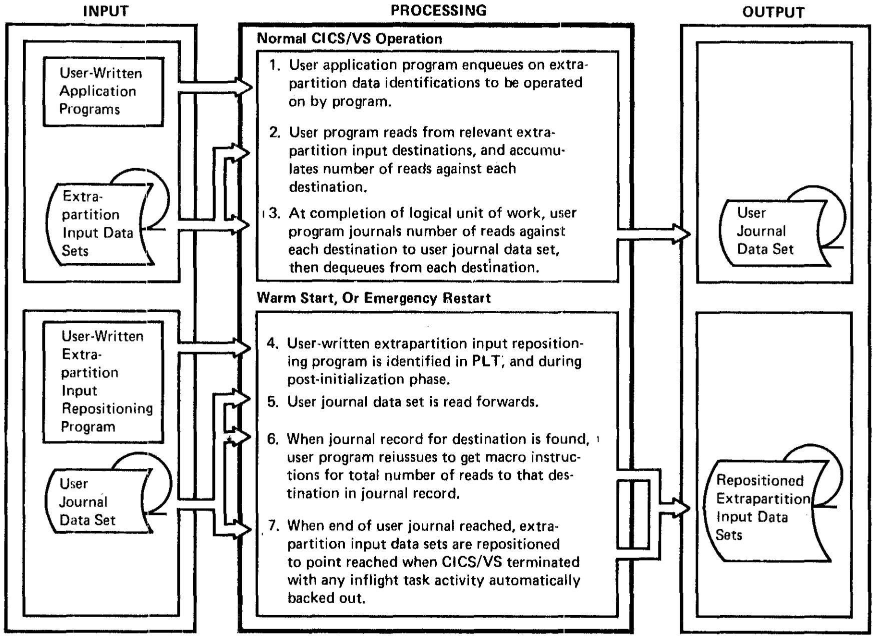
CICS/VS does not provide for recovery of extrapartition data sets. If this is significant to the online application, the system design team must develop procedures to enable that informaticn tc be recovered for continued execution cn restart, following either a controlled or uncontrolled shutdown of CICS/VS.

There are two areas which must be considered in recovery of extrapartition data sets:

- Input data sets
- Output data sets

Input Data Sets

The main information required on restart is the number of records processed up to the time of system termination. This may be recorded during processing using the journaling capability of CICS/VS, as described in the following paragraphs and illustrated in Figure 8-9.


INPUT
PROCESSING
OUTPUT
User-Written Application Programs
Extra-partition Input Data Sets
User-Written Extra-partition Input Repositioning Program
User Journal Data Set
Normal CICS/VS Operation
Warm Start, Or Emergency Restart
User Journal Data Set
Repositioned Extrapartition Input Data Sets

1. User application program enqueues on extrapartition data identifications to be operated on by program.
2. User program reads from relevant extrapartition input destinations, and accumulates number of reads against each destination.
3. At completion of logical unit of work, user program journals number of reads against each destination to user journal data set, then dequeues from each destination.
4. User-written extrapartition input repositioning program is identified in PLT, and during post-initialization phase.
5. User journal data set is read forwards.
6. When journal record for destination is found, user program reiussues to get macro instructions for total number of reads to that destination in journal record.
7. When end of user journal reached, extrapartition input data sets are repositioned to point reached when CICS/VS terminated with any inflight task activity automatically backed out.

Extended Description

3. If system terminates before user journal record is written for a destination, previous journaled information for that destination is used.	7. As inflight tasks did not complete a LUW no activity for those tasks appears on the user journal data set.

Figure 8-9. Extrapartition Input Data Set User Recovery (Logical Recovery)

Each application program which reads records from extrapartition input destinations should first enqueue to ensure exclusive access to those destinations. This will prevent interleaved access to those same destinations by other concurrently executing tasks, and so enable the user's extrapartition recovery technique to operate correctly. (This provides a similar capability to that described earlier in this chapter in "Protected Resources.")

Transient data GET macro instructions are then issued by the application program, to read and process extrapartition input records. The application programs accumulate the total of input records read and processed during execution for each destination. The total number of GETs is journaled by the program to a user journal data set, together with the relevant destination identifications. This journaling is only carried out at completion of a logical unit of work, which may be at end of task or at a user sync point, such as on a conversational terminal operation (see "Logical Task Synchronization" in this chapter).

Following output of the user journal record, the application program dequeues itself from the input destinations, to permit other application programs to access those extrapartition input destinations.

If uncontrolled shutdown occurs prior to this user journaling, no records will appear on the user journal data set for that logical unit of work, and the effect of that in-flight task is therefore automatically backed out on emergency restart. However, if the user journal record is written before uncontrolled shutdown, this completed input data set processing will be recognized on emergency restart.

On a controlled shutdown, CICS/VS will wait until the application program completes execution normally. This will enable the input data set activity to be recorded by the user for subsequent warm start.

On emergency restart following uncontrolled shutdown or on a warm start following a controlled shutdown, the following procedure may be utilized. This will reposition the extrapartition input data sets, to reflect the input and processing of their records during previous CICS/VS operation.

An uncontrolled shutdown does not permit a tape journal data set to be closed normally. This may be achieved through the use of the CICS/VS tape end-of-file utility program (see "Preparation of User Journals" in this chapter) prior to execution of the user recovery program.

A user-written extrapartition input recovery program may be identified in the PLT for execution during the post-initialization phase. This program reads the user journal data set forward. Each journaled record indicates the number of GETs issued to the relevant extrapartition input data set during previous execution of application programs. The same number of transient data GET macro instructions is issued again by the recovery program, to the same input destination as referenced previously (see Figure 8-9).

On reaching the end of the user journal data set, the intrapartition input data sets are positioned at the same point they had reached prior to initiation of tasks which were in-flight at uncontrolled shutdown. The result is the logical recovery of these input data sets with in-flight task activity backed out.

Output Data Sets

The user recovery of output data sets is somewhat different from the recovery of input data sets.

For a tape output data set, a new output tape should be used on restart. The previous output tape can be utilized if necessary to recover information recorded prior to termination.

To avoid the loss of data in tape output buffers on termination, it may be desirable to write unblocked records. Alternatively, data may be written to an intrapartition disk destination (which is recovered by CICS/VS on a warm start or emergency restart) and periodically copied to the extrapartition tape destination by an automatically initiated task. In the event of termination, the data is still available on restart to be recopied. (This is discussed further in the logical recovery technique presented below.)

If a controlled shutdown of CICS/VS occurred, the previous output tape is closed correctly, and a tapemark is written. However, on an uncontrolled shutdown such as on a power failure cr machine check, a tapemark will not be written to indicate the end of the tape. This may be achieved by execution of the CICS/VS tape end-of-file utility program pricr to use of that tape for subsequent recovery (see "Preparation of User Journals" in this chapter).

For a line printer output data set, if it is satisfactory to continue output from the point reached pricr to system termination, no special action need be taken. Hcwever, if it is desired to continue output from a defined point, such as at the beginning of a page, it may be necessary to use a journal data set. As each page is completed during normal CICS/VS operation, that fact may be noted by writing a record to a journal data set. Cn restart, the page which was being processed at the time of failure can be identified from the journal data set, and that page reprocessed to reproduce that same output again, from the beginning of the page. Alternatively, an intermediate intrapartition destinaticn may be used (as previcusly described) for tape output buffers.

Since extrapartition disk output data sets are normal sequential data sets, a difficulty exists in locating the pcint where last output was written and continuing output frcm that point. Two techniques for user reccvery are presented in the following paragraphs, to permit the end of the extrapartition data set to be located fcr subsequent continuaticn of cutput.

The first technique is suitable for physical recovery of the output data set. The end of the data set is located, and output then continues from that point with nc attempt to back out in-flight tasks. This technique assumes that the programs which will subsequently process this output data set can detect possible duplicate (or missing) records resulting from lack of backout on physical recovery and take appropriate acticn. This detection may be achieved by each extrapartition record having a consecutive sequence number, for example.

The second technique is suitable for logical recovery of the output data set. This also may utilize a ccnsecutive sequence number in each record (for subsequent program detection). The lcgical recovery capability offered by transient data intrapartition destinations is utilized for "spooling" of a logical section of data to an intrapartition destinaticn, and then subsequently copying each completed logical section to the extrapartition destination. If an uncontrolled shutdcwn cccurs, the in-flight transient data intrapartition task activity can be backed out.

These two techniques are discussed further in the following paragraphs. The physical recovery technique is illustrated in Figure 8-10.

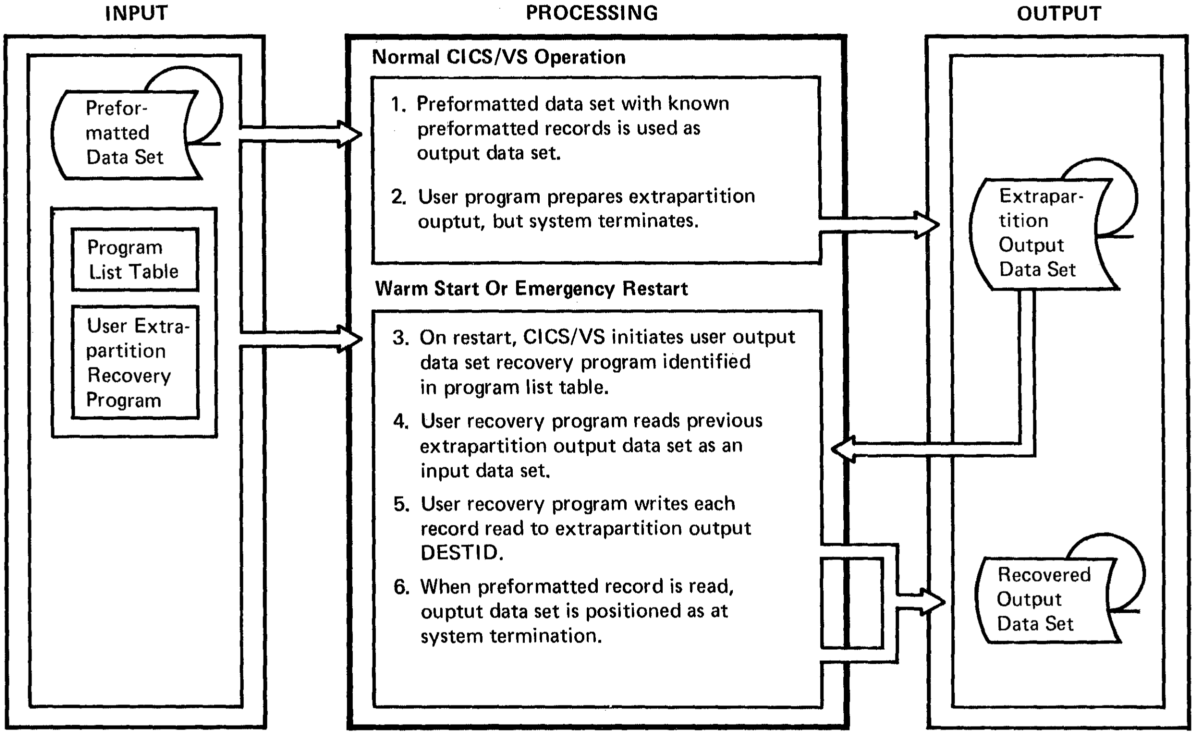

INPUT
PROCESSING
OUTPUT
Prefor-matted Data Set
Program List Table
User Extra-partition Recovery Program
Normal CICS/VS Operation
1. Preformatted data set with known preformatted records is used as output data set.
2. User program prepares extrapartition ouptut, but system terminates.
Warm Start Or Emergency Restart
3. On restart, CICS/VS initiates user output data set recovery program identified in program list table.
4. User recovery program reads previous extrapartition output data set as an input data set.
5. User recovery program writes each record read to extrapartition output DESTID.
6. When preformatted record is read, ouptut data set is positioned as at system termination.
Extrapar-tition Output Data Set
Recovered Output Data Set

Figure 8-10. Extrapartition Outpput Data Set User Recovery (Physical Reccvery)

- Physical Reccvery

On restart, a user program ir the progran list table is utilized in the post-initialization phase to read the previous output data set as an input data set. Reccrds read cn restart from the previous output data set are written to the output destination to be used cn restart. The result is the copying and reestablishment of a new output data set to the same status as the previous data set prior to terminaticn.

However, to terminate the copy operation, it is necessary to identify when the end cf the previcus output data is reached. This is best achieved by prefcrmatting the entire cutput data set prior to initial use by CICS/VS to kncwn record contents, such as records containing binary zeros. Accordingly, as the previous output data set is read by the post-initialization program as an input data set, each reccrd retrieved from that data set is checked to determine whether it is a preformatted record. When the first preformatted reccrd is retrieved from the input data set, the output data set has been updated to reflect the output location reached prior to termination. Subsequent output to that same extrapartition destination then continues from that point when CICS/VS restarts.

- Logical Recovery

  An alternative technique, which can be utilized in the event of either a controlled or an unccntrolled shutdcwn, takes advantage of the recovery suppcrt provided by CICS/VS for intrapartition destinations. This technique requires extrapartition data to be directed during normal CICS/VS operation instead of to an intrapartition destinaticn. (This requires applicaticn programs to prepare that data as variable-length records, as required by intrapartition destinaticns.)

  Such data is directed to an intrapartition destination, rather than an extrapartition destination. When a logical section of data (for example, a printer page) has been queued to that destination (based upcn its trigger level), a task is automatically initiated. This task reads those reccrds queued to the intrapartition destination, converts the reccrd fcrmat tc fixed-length if required, and then writes those records to the appropriate extrapartiticn destination. This transfer of data frcm intrapartition to extrapartition continues until all records currently queued to the intrapartition destination have been read. At this point, a transient data PURGE macro instruction may be issued to release those tracks already read, for subsequent reuse.

  Several dummy records may be written to the extrapartition destination, when end cf queue on the intrapartition destination has been reached. This forces a ccmplete blcck cf records in the extrapartiticn data set buffers (QSAM buffers for CICS/OS/VS) to be written out. Subsequent programs which read this data set must, of course, test for these dummy records and ignore them.

  An unccntrolled shutdcwn may cccur before the data currently queued on intrapartition has been transferred to extrapartition. If the transfer task is unable tc ccmplete normally, the effect of that in-flight task is backed out by CICS/VS on emergency restart. The data is still available cn the intrapartition destination, and can be subsequently recopied to the extrapartiticn destination following restart when the autcmatically initiated task is activated again. The user must positicn the extrapartition data set at the end of the last completed logical section (using a sequence number in the record, for example), before this recopying from the intrapartition destination (following emergency restart) is initiated.

USE OF JCURNAL CCNTRCL FCR SEQUENTIAL DATA SETS

As previously discussed, recovery of extrapartition data sets on warm start or emergency restart is a user responsibility. It can require a considerable amount of user implementation effcrt. One approach which may be considered fcr user recovery of tape or disk extrapartition data sets is based on the use of journal ccntrol by the user's applicaticn programs in place of extrapartition transient data. This requires modification of existing applicaticn programs and a possible data conversion step before processing. However, this effort can be offset by the additional support offered by journal control over transient data. The following discussion compares the different support offered by each facility.

- Record Format

  Journal contrcl suppcrts cnly variable-length blocked records with prefix contrcl informaticn preceding the data record. Extrapartiticn data sets, however, may be fixed-length or variable-length, blocked or unblccked, with nc prefix control information. Thus, journal control data sets must be converted if compatibility is desired with existing extrapartition data sets.

- Buffering

  CICS/OS/VS uses QSAM for extrapartition I/O and BSAM for journal control I/O. CICS/DOS/VS uses SAM for both. Journal control uses a sophisticated buffering technique which can be controlled at CICS/VS initialization time by using different versions of the generated journal control table (JCT) having varying journal buffer size and buffer shiftup values for each journal data set. (See *CICS/VS System Programmer's Reference Manual*, SH20-9004.)

- Output Buffer Flush

  Extrapartition I/O does not enable a task to force output of the buffer on task termination. Journal control enables a task to force buffer output at any time (see "Journaling" in this chapter). In addition, a task may close and open journal data sets for input or output at any time.

- Work File Capability

  Extrapartition data sets can only be supported as sequential data sets. Journal data sets offer both sequential and work file capability using journal control NOTE and POINT macro instructions.

- I/O Overlap

  Extrapartition I/O is synchronous and the task waits for I/O completion. In addition, journal control I/C can be synchronous or asynchronous to enable the task to continue processing while I/O is in progress.

- I/O Wait Time

  Extrapartition I/O results in a DOS/VS or an OS/VS WAIT being issued which causes the entire CICS/VS partition/region to wait. Journal control uses CICS/VS WAIT, which enables the partition/region to remain active and allows other CICS/VS tasks to process during the journal control I/O. The result is additional CICS/VS processing time if journal control is used.

- Volume Switching

  Extrapartition I/O does not support automatic volume switching. The master terminal operator can, however, dynamically open or close extrapartition data sets. Journal control supports one or two tape drives (or disk extents) for each journal data set with automatic volume switching. Disk extents are automatically reused, if insufficient space is allocated. Accordingly, if disk reuse is to be avoided, the total disk space allocated must be sufficient to contain the total journal data set contents.

- Concurrent Task Access

  Extrapartition transient data does not explicitly inhibit two or more tasks from accessing the same destination concurrently. To avoid the consequent data integrity problems (see "Protected Resources" in this chapter), each task must explicitly enqueue on the destination. There is no similar integrity exposure for journal control output, as each journal control record identifies (in its prefix) the originating task number. Journal control explicitly inhibits concurrent access by two or more tasks to an input journal data set. Other tasks will wait until the first task has finished its use of the data set, and close it.

- Recovery

  Journal data sets are preformatted using utilities supplied by CICS/VS (DFHTAP for tape and DFHJCJFP for disk). A utility is also supplied to write a tapemark at the point reached in a tape data set at uncontrolled shutdown. (See CICS/VS Operations Guide.) The journal control OPEN macro instruction enables an application program to open a journal data set for either input or output, and automatically position the data set at the point reached at uncontrolled shutdown to continue I/O against the data set from that point.

  Therefore, the system designer may wish to specify that journal control be used for sequential data sets, in place of extrapartition transient data.

INDIRECT DESTINATIONS

The use of indirect destinations for providing backup of intrapartition terminal destinations or extrapartition destinations has been discussed in this chapter. With the recovery of intrapartition transient data by CICS/VS on restart, all intrapartition indirect destinations are also reestablished to their status at the time of system termination.

However, if a user-written DCT modification program (as described for terminal backup in this chapter and also in Chapter 4) was used to modify indirect destinations, these user modifications will not be reestablished in the DCT on restart by CICS/VS, since it is not aware that any modification was made. It is the user's responsibility to journal information when each user DCT modification is made. This journaled information must then be utilized by user programs on system restart, to reestablish the same DCT modification in force at termination.

DUMP DATA SET RECOVERY

On restart, unless specific user action is taken, the dump data set is opened by CICS/VS as if it were a new dump data set. Subsequent CICS/VS dumps then override earlier dumps produced prior to termination. To avoid this situation, it may be necessary for the user, prior to restarting the online system, to print out all dumps taken up to system termination, by utilizing the CICS/VS dump utility program. Alternatively, a user post-initialization program may automatically attach the master terminal transaction CSMT which will switch to the alternative dump data set. Dumps prior to system termination are consequently retained on the original data set, and dumps following system restart are directed to the alternative dump data set. The original dump data set may then be printed out using the dump utility program from another batch partition at some later time, when required.

While the master terminal operator can switch the dump data sets by using the master terminal operation transaction (CSMT) on system restart, it is better to make this an automatic function of restart by specifying a program in the program list table used by the post-initialization phase, as described above.

CICS/VS PROGRAM LIBRARY RECOVERY

The CICS/VS program library is a private core-image library for CICS/DOS/VS, or a partitioned data set for CICS/OS/VS.

If no changes are made to the program library during CICS/VS execution, no special action need be taken to recover this library on system restart.

If updated versions of CICS/VS application programs were cataloged to the CICS/VS program library prior to system termination, and the PPT could not be updated dynamically by the master terminal operator to point to the new version of the program before termination, on system restart the PPT is automatically reestablished to point to the current version of each application program. Accordingly, no action need be taken during system restart for PPT maintenance, either by post-initialization programs or by the master terminal operator.

TRANSACTION RECOVERY AND RESTART

The following sections discuss message recovery and transaction restart considerations first for VTAM and BTAM terminals.

RECOVERY OF MESSAGES ASSOCIATED WITH VTAM TERMINALS

CICS/VS automatically logs input messages for transactions specified in the PCT as "message-protected," and logs committed output messages and responses indicating their subsequent positive receipt by the terminal. (See "Protected Messages" in this chapter.) Automatic logging only applies to message-protected transactions associated with VTAM terminals which can support message recovery. It is a user responsibility to carry out this journaling for transactions associated with BTAM terminals. Inquiry transactions (which do not update data sets) should not be specified as "protected" in the PCT.

The CICS/VS recovery utility program (RUP) identifies in-flight tasks during emergency restart and transfers logged input messages or committed output messages (for which receipt acknowledgment is outstanding) to temporary storage. (See Figure 8-11 and "Protected Messages.") A temporary storage message "cache" is defined for each terminal which had an in-flight task at uncontrolled shutdown. This message cache has a unique temporary storage DATAID of DFHMXXXX (where XXXX=four-character terminal identification). This cache contains either the input message for the in-flight LUW, or, if the LUW completed normally but definite receipt of a committed output message was not acknowledged by the terminal, the committed output message.

A committed output message can optionally be automatically retransmitted by CICS/VS on emergency restart to establish message resynchronization. Thus, committed output message integrity is preserved.

An input message for an in-flight LUW is not automatically reprocessed by CICS/VS on emergency restart. CICS/VS will back out all in-flight activity for that LUW. The decision to reprocess the input message is a user responsibility based on factors such as application requirements and security of information. A technique for user-activated transaction restart is discussed later in this chapter.

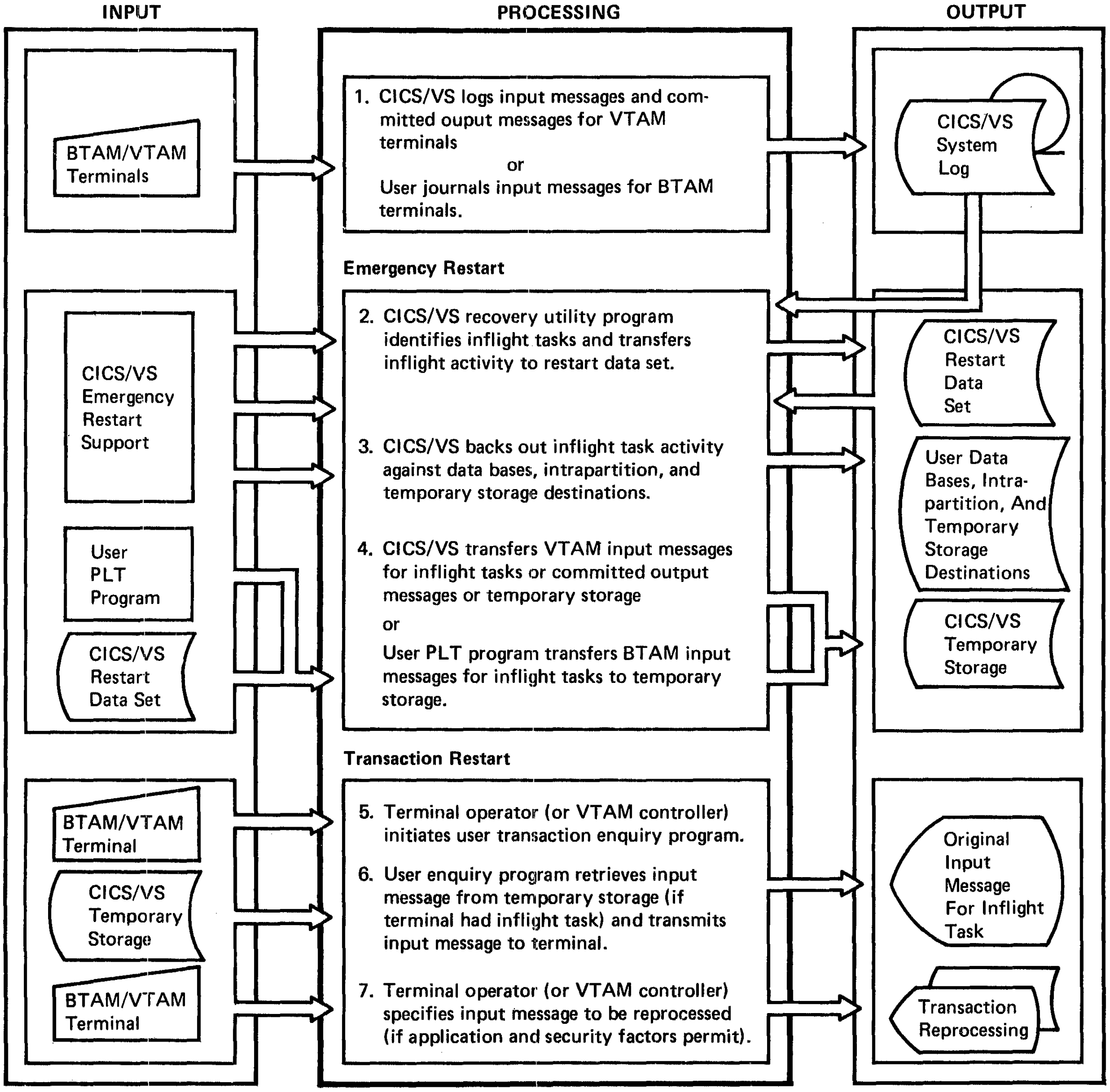

INPUT
PROCESSING
OUTPUT
BTAM/VTAM Terminals
1. CICS/VS logs input messages and committed ouput messages for VTAM terminals
or
User journals input messages for BTAM terminals.
CICS/VS System Log
Emergency Restart
CICS/VS Emergency Restart Support
2. CICS/VS recovery utility program identifies inflight tasks and transfers inflight activity to restart data set.
CICS/VS Restart Data Set
3. CICS/VS backs out inflight task activity against data bases, intrapartition, and temporary storage destinations.
User Data Bases, Intra-partition, And Temporary Storage Destinations
User PLT Program
4. CICS/VS transfers VTAM input messages for inflight tasks or committed output messages or temporary storage
or
User PLT program transfers BTAM input messages for inflight tasks to temporary storage.
CICS/VS Restart Data Set
CICS/VS Temporary Storage
Transaction Restart
BTAM/VTAM Terminal
5. Terminal operator (or VTAM controller) initiates user transaction enquiry program.
CICS/VS Temporary Storage
6. User enquiry program retrieves input message from temporary storage (if terminal had inflight task) and transmits input message to terminal.
Original Input Message For Inflight Task
BTAM/VTAM Terminal
7. Terminal operator (or VTAM controller) specifies input message to be reprocessed (if application and security factors permit).
Transaction Reprocessing

Figure 8-11. Message Recovery Transaction Restart

RECOVERY OF MESSAGES ASSCCIATED WITH BTAM TERMINALS

To provide for implementation of a user-written transaction restart program, each user task must journal its terminal input message as soon as it commences execution. This can be achieved either by each task LINKing to an installaticn-level, user-written transaction jcurnal prcgram, or by specifying this transaction jcurnal program in the PCT for each transaction code for which input messages are to be journaled. In this latter case, after jcurnaling the input message the user-written transaction journaling program examines the transaction code in the input message, and transfers contrcl to the relevant application prcgram.

Alternatively, journaling of terminal input messages may be carried out by user-written code in a terminal contrcl prcgram user exit.

The informaticn which wculd be included by the user in the transaction journal record is detailed belcw. Scme of this information is automatically provided in the system prefix by journal control. (See "Journaling" in this chapter.)

- Terminal identificaticn
- Operator identification
- Transaction code used to attach the task (extracted from the TCA)
- Time of day when the journal reccrd was writter
- Task pricrity frcm the task ccntrol area (TCA)
- Other informaticn required tc further identify that transaction

The above information is inserted in the system and user prefixes of the journal record, and the original input transaction from the terminal input area may be placed by the user's transaction journal program in the data section cf the jcurnal record.

The user-written transacticn journal program may journal terminal input messages tc a user jcurnal data set and/or the system log. Input messages written to the user journal data set can be used for audit purposes, if required. Input messages written to the system log will be transferred to the restart data set by the CICS/VS recovery utility program, for in-flight tasks when an uncontrolled shutdown occurred. This is carried out during emergency restart.

Following RUP processing, the in-flight activity of that task against data bases, intrapartiticn, and tempcrary storage destinations is backed out by CICS/VS. (See Figure 8-11.) User-written programs identified in the program list table (PLT) are then executed.

A user-written PLT prcgram shculd read terminal input messages from the restart data set, where they were transferred by RUP. This program should ccnstruct a temporary storage record, using the same temporary storage message cache format as that established by CICS/VS for VTAM terminals. This ccntains the terminal identification, transaction code, and task sequence number. This infcrmation can be extracted from the system prefix which was constructed for the record on the restart data set when it was criginally written by journal control to the system log. The program should flag the message as a task-originating input record and then write the input message (now in the VTAM temporary storage message cache format) to temporary storage using a DATAID of DFHMXXXX, where XXXX=four-character terminal identification.

The input message is then available to the user for reprocessing based on application requirements and security of information. By using the same fcrmat as that used fcr the VTAM temporary storage message cache, reprocessing cf input messages frcm both BTAM and VTAM terminals can be handled ccnsistently. (See Figure 8-11.)

As BTAM terminals are unable tc indicate definite receipt of specific output messages as can VTAM terminals, the VTAM ccncept of committed output messages is nct applicable to BTAM terminals. Consequently, output message journaling by the user cannct ensure output message integrity. User applicaticn programs should delay issuing BTAM terminal control writes until the completion of the logical unit of work. It is a user responsibility to identify to terminal operators on emergency restart the transactions which were in-flight at uncontrolled shutdown and subsequently backed out. This is necessary to ensure that the backed out transactions (whose application and security requirements permit) are reprocessed. The following is a technique for user-activated transaction restart following emergency restart.

TRANSACTION RESTART (BTAM AND VTAM TERMINALS)

At the completion of emergency restart, input messages for in-flight tasks which have been backed out are available in the temporary storage message cache for each relevant terminal. (See Figure 8-11.) An inquiry program should be written by the user so that terminal operators can determine whether their last transaction was fully processed prior to uncontrolled shutdown, or whether it was backed out on emergency restart. This program should issue a temporary storage GET macro instruction, using as the DATAID DFHMXXXX, where XXXX=four-character identification of the enquiring terminal. If that terminal had no in-flight task, an IDERROR error indication will be returned to the program. (See CICS/VS Application Programmer's Reference Manual.) If the task was in-flight (or had a VTAM-committed output message), a temporary storage record will be returned to the program. The program should check the temporary storage message cache record flag for presence of an input message, and then present that input message and associated information to the terminal operator. (See Figure 8-11.) The terminal operator can then decide whether the transaction is to be reprocessed. A reprocessing request by the terminal operator should only be accepted if the terminal operator has the necessary authority (based on security codes, or operator class in the TCT) to make that decision for the particular transaction. Processing then proceeds as if the transaction has just been entered from the terminal.

If the input message is associated with a VTAM programmable controller, the inquiry program may be automatically initiated by the controller after message resynchronization and recovery have been completed. The in-flight input message (transmitted back to the controller by the inquiry program) may be presented automatically to the relevant terminal operator for a reprocessing decision. Alternatively, if application and security considerations permit, the controller itself may automatically make the reprocessing decision and notify the enquiry program accordingly.

TERMINAL OPERATOR RESTART

The previously described technique enables the terminal operator to determine, on emergency restart, the status of transactions submitted prior to uncontrolled shutdown. By initiating that transaction inquiry program, the terminal operator is notified whether his last transaction was completely processed, or was in-flight. If it was in-flight (and therefore backed out), he is presented with the original input message to decide whether reprocessing is necessary. If a committed output message had not been received by his VTAM terminal prior to uncontrolled shutdown, it is retransmitted on emergency restart.

Thus, the terminal operator is presented with sufficient information on emergency restart to allow him to identify the point reached in previous communication with CICS/VS, and reestablish application processing activity.

TERMINAL ERROR RECOVERY

If a terminal error is detected, BTAM or VTAM attempts to recover from the terminal or data transmission error. If recovery is not successful, that indication is passed to the CICS/VS terminal abnormal condition program (TACP) for BTAM, or node abnormal condition program (NACP) for VTAM. The TACP or NACP analyzes the terminal error failure and may attempt additional error recovery. It then specifies various default actions (such as requesting that the error terminal be disabled, the error line be disabled, the error task be abnormally terminated, or the error line disconnected if it is a switched line). Control is then passed to a terminal error program (TEP) or node error program (NEP) to enable further application-dependent recovery to be taken. CICS/VS provides a sample TEP which can be utilized to generate a TEP tailored to the user's requirements (see *CICS/VS System Programmer's Reference Manual* for more detail). Alternatively, a user-written TEP or NEP can be utilized.

Refer to Figure 3-12 and "Terminal Error Recovery" in Chapter 3 for a description of procedures which may be coded in the terminal error or node error program by the user to provide, in the event of an unrecoverable terminal error, for:

- Additional transmission retry to overcome a terminal error

- Transmission of an output message to an alternate terminal

- Queuing of the output message to an intrapartition destination, for later transmission when the terminal error has been rectified

TERMINAL BACKUP

CICS/VS does not provide specific support for terminal backup. However, the use of the terminal error program or node error program enables various forms of backup to be implemented by the user. This may permit terminal messages to be received on a backup terminal, if an unrecoverable error occurred on the terminal which would normally receive those messages.

Refer to "Terminal Backup" in Chapter 4 for techniques used to provide backup facilities in the event of unrecoverable terminal errors. This topic describes the utilization of the terminal error or node error program, together with DCT modification through user programming to allocate backup terminals or devices.

TERMINAL RECONFIGURATION

The technique described in Chapter 4 for allocation of backup terminals permits considerable flexibility in terminal reconfiguration in the event of I/O errors (see "Dynamic Terminal Reconfiguration" in that chapter).

Through the destination control table (DCT) and indirect destinations, CICS/VS users can readily implement user-written terminal reconfiguration procedures. The terminal reconfiguration can only be achieved by user-written code to dynamically modify the DCT to reflect a configuration change. It is the user's responsibility to journal this DCT modification, and reestablish this modification (if necessary) on system restart, following system termination. (See "Terminal Backup" in Chapter 4, and "Journaling" in this chapter, for further discussion of this requirement.)

"Dynamic Terminal Reconfiguration" in Chapter 4 also illustrates how this DCT modification may be used to dynamically allocate different devices to receive applicaticn output at different times cf the day, based upcn application requirements.

Techniques for dynamic terminal reconfiguraticn in programatle controllers are also discussed in "Dynamic Terminal Reconfiguration."

DEVICE RECOVERY

CICS/VS does not specifically provide support for the recovery or backup of various devices. If devices such as tape drives, card readers, printers, or data sets fail during online operation, procedures must be designed, by the user, tc enable the online application to continue executicn.

Following failure of a card reader, transactions which would normally be entered from such a simulated sequential terminal may be entered from any other terminal (fcr example, from conversational terminals such as the 3270 Information Display System or 2740 Communications Terminal, cr from batch terminals such as the 3735 Programmable Buffered Terminal, the 2770 Data Communications System, or 2780 Data Transmission Terminal, depending upon vclume).

If a line printer fails, the output which would normally be directed to that printer may be directed to cther devices, depending upcn whether the printer is being used as a simulated sequential terminal output device, cr whether it is being used as an extrapartition output device.

If the printer is being used as a simulated sequential terminal output device, an alternative terminal can be allocated to receive that output using the techniques described in this chapter (and in Chapter 4) for terminal backup.

If the line printer is being used fcr extrapartition output, procedures similar tc those discussed in "Extrapartition Device Failure" in this chapter may be carried out.

If a tape drive failure occurs, procedures described for extrapartition device failure may also be applied.

To handle disk I/O errcrs, application programs should specify in the CICS/VS file control macro instruction that ccntrol is to be passed tc particular routines in the application program (or to a common disk error routine for the installaticn) on detection of various error conditions. These routines should free any file I/O areas or file work areas which are still allocated as a result cf the error situation and then attempt to retry the cperation. If the retry is not successful, the data set in question may be regarded as having "failed." Such data set failure may be as significant to the operaticn of the application as the failure of terminals cr devices as described above.

Depending upon the impcrtance of that data set, the design team may decide to carry a duplicate data set on a different pack. If a serious I/O error is enccuntered in the main data set, the duplicate data set may be utilized in place of the main data set.

When duplicate data sets are to be used, if the main data set is to be updated, deleted frcm, cr added tc cnline, then all updates, deletions, and additions must be applied to each ccpy of the data set. This introduces extra complexity when recovery procedures are designed, since all data sets must be kept in step in the event of abnormal terminaticn or program failure. If some of the data sets were updated and cthers were not, those data sets nct updated must be brought to the same status as the main data set, or the updates applied to the main data set must be backed out on restart. See "Data Set Backout" in this chapter. Complexity is also introduced in the handling of an unrecoverable I/O error on a write to a duplicate disk data set.

EXTRAPARTITION DEVICE FAILURE

CICS/VS does not provide support for recovery from extrapartition device failure. If an extrapartition device failure occurs, such as failure of a tape drive, line printer, or extrapartition data set on disk, use may be made of preplanned extrapartition backup devices as described in the following paragraphs.

Allocation of the extrapartition output to a backup device can be achieved by using indirect transient data destinations, and defining backup devices as additional destinations, which are normally open but never referenced. A user-written DCT modification transaction (see "Terminal Backup" in Chapter 4) may then be entered by the master terminal operator to access the relevant failing extrapartition destination and change that destination to refer indirectly to a particular backup destination, regardless of whether it is extrapartition or intrapartition.

If the backup device is an intrapartition terminal destination, the failing extrapartition destination indirectly points to that terminal destination, which in turn points to the actual terminal to be used to receive the extrapartition output. However, to enable an intrapartition device to be used to back up an extrapartition device, only variable-length records can be used for the extrapartition output.

If the backup device is another extrapartition device, the DCT modification program changes the failing destination to indirectly point to the backup extrapartition destination, for either input or output backup.

BATCH DL/I DATA BASE RECOVERY

DL/I DOS/VS and IMS/VS DL/I supply a number of batch utility programs designed to provide rapid recovery of a data base rendered unusable because of:

- Disk I/O errors
- Abnormal task termination
- System failure

DI/I data base recovery is designed to provide a rapid, accurate, and easy to employ means of restoring the contents of the physical data base after destruction. The recovery system is supported by four utility programs which provide for data base recovery from the error situations listed above.

- Recovery from disk I/O errors is achieved by the use of the:

  1. Data Base Data Set Image Copy: creation of dump images of data base data sets for backup purposes.

  2. Data Base Change Accumulation Utility: accumulation of data base changes since the last complete backup dump.

  3. Data Base Data Set Recovery Utility: restoration of data sets of a data base using a prior image copy (backup dump)

and the accumulated changes, together with changes made since the last accumulated change run.

- Recovery from abnormal task termination or system failure is achieved by the use of the:

    4. Data Base Backout Utility: removal of changes made to data bases by selected application programs.

These batch data base recovery utilities are discussed further later in this chapter. The recovery utilities do not support HSAM data bases, since HSAM does not support insertions (additions), deletions, or replaces (updates).

Utilities are not provided by DL/I ENTRY for data base recovery or data base backout.

During execution of batch application programs utilizing DL/I DOS/VS or IMS/VS data bases, any data base insertions (additions), deletions, or replacements (updates) are automatically written by DL/I to a DL/I system log tape. CICS/VS application programs which use DL/I online result in DL/I activity being written to the system log.

BATCH DL/I SYSTEM LOG

The DL/I system performs the function of logging to tape of all information associated with the modification of data within a data base. The DL/I system log is not required when a data base is loaded, or when a batch application program only retrieves data (that is, read-only). The log is required for all other cases.

The logging of data base modifications, additions, and deletions is done on a physical basis to facilitate a quick recovery procedure. Only DL/I calls that actually cause a change to be made to a data base are logged. Two sets of information, a "before" set and an "after" set, are logged for each modification made. This information is directed to the DL/I system log when DL/I is used offline or to the CICS/VS system log when DL/I is used online with CICS/VS.

The "before" information is that required by the CICS/VS recovery utility program when DL/I is online, or by the data base backout utility when DL/I is used offline. It is used to back out a partially completed update series and to restore a data base to some prior version. This is described further under "Batch DL/I Data Base Backout" and "Online DL/I Data Base Backout" in this chapter.

The "after" information is that required by the data base data set recovery utility to restore the data base from a previous backup copy. This is described further in "DL/I Data Base I/O Recovery" later in this chapter. It applies to DL/I I/O recovery in both batch and online environments.

In addition to the above, the DL/I system log contains:

- A CICS/VS application program scheduling record (for DL/I DOS/VS), written whenever a CICS/VS application program issues a successful DL/I scheduling call against a PSB that is update sensitive to one or more DL/I data bases. See IMS/VS DL/I Application Programming Reference Manual.

- A data set open record, written whenever a data set is opened.

- A CICS/VS application program termination record, written whenever a DL/I application program task issues a DL/I termination call, or terminates while still scheduled to DL/I (if a scheduling record was written for this task).

These three record types are used to facilitate data base recovery.

DL/I RECOVERY UTILITIES

Data base recovery is provided by both DL/I DOS/VS and IMS/VS DL/I. The information on the CICS/VS system log for online operation, or on the DL/I system log tapes for offline operation, and periodically created tape copies of the data sets representing the data base, are utilized by DL/I for recovery. The log information contains records which describe the modifications made to the data sets within a data base. The number of log tapes necessary for data base recovery depends upon the frequency of data base copy execution, the volume of data base modification, and the total usage of DL/I. Since information from a considerable number of log tapes may be necessary for data base recovery (all log tapes created since the last data base copy), a technique for accumulating all the latest changes to each specific data set in a data base is provided to permit faster recovery of data bases. This accumulation of the latest data base modification is performed by the data base change accumulation utility. Thus, the information necessary for data base recovery is contained within the three following sources:

1. Data base (data set) copy tape created when the data base was last dumped, through the data base data set image copy utility.

2. Data base change accumulation tape, created from log tapes available since the last data base copy creation, by the data base change accumulation utility.

3. Log tapes employed since the last data base copy creation and not incorporated into the accumulation tape. This includes at least the log tape in use when problems were encountered with the data bases.

The data base data set recovery utility program, which is the final stage of data base recovery, operates as an application program under control of the DL/I system. Recovery is done for individual data sets. In most cases, a data set is synonymous with a data base. This is true with HDAM and HISAM root-only data bases. However, most HISAM data bases consist of two data sets: a key-sequenced data set and an entry-sequenced data set. Thus, for HISAM, if the contents of one data set are destroyed, it is not necessary to recover the complete data base. Recovery is by direct physical replacement of data within a data set rather than by logical reprocessing of transactions.

DL/I Data Base I/O Recovery

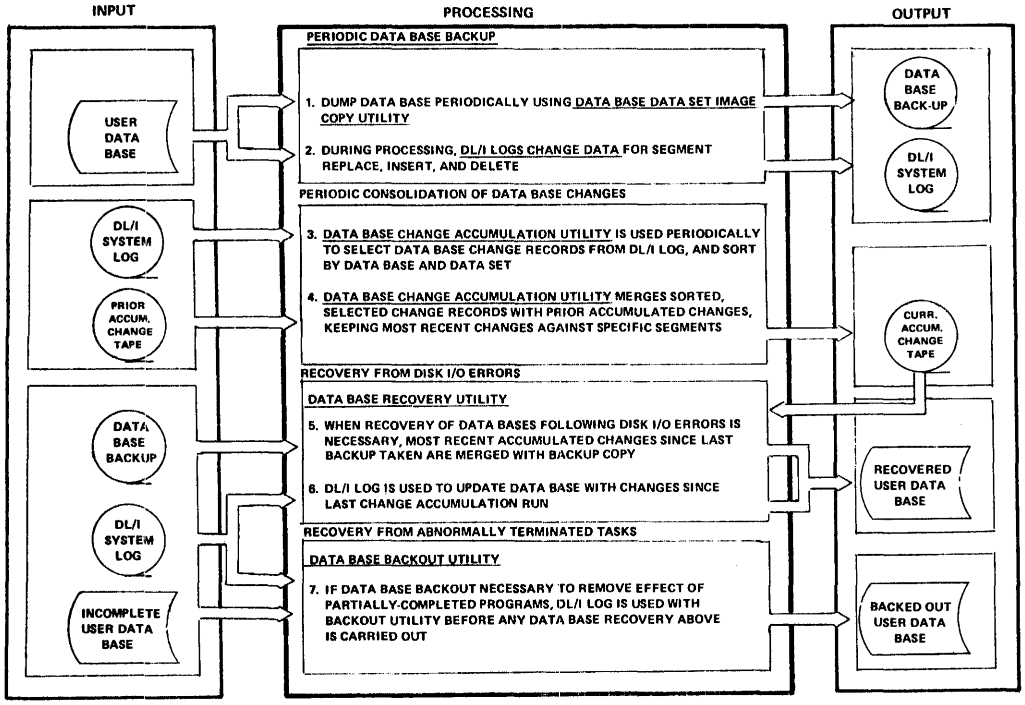
The following functions, illustrated in Figure 8-12, are required to accomplish data base recovery following an unrecoverable I/O error encountered on DL/I data bases used either offline or online with CICS/VS.

- Log change data for a segment replacement, insertion, or deletion, including the identification of the update segment. This function is performed automatically by DL/I for all data bases, using the DL/I system log for batch or CICS/VS system log for online.

- Dump the data set periodically to provide a backup copy, using the data base data set image copy utility.

- Select the change data base log records from the DL/I system log or CICS/VS system log and sort them in order by data base and data set. If the data set is key-sequenced VSAM, the sort is ordered by VSAM key. If the data set is entry-sequenced VSAM, the sort is by entry-sequenced data set relative byte address (RBA). Selecting and sorting are performed as part of the change accumulation data set creation by the data base change accumulation utility.

- Merge the sorted, selected changed records with the prior accumulated changes, keeping only the most recent data. Merging is performed as part of the change accumulation data set creation by the data base change accumulation utility.


INPUT
PROCESSING
OUTPUT
USER DATA BASE
DL/I SYSTEM LOG
PRIOR ACCUM. CHANGE TAPE
DATA BASE BACKUP
DL/I SYSTEM LOG
INCOMPLETE USER DATA BASE
PERIODIC DATA BASE BACKUP
PERIODIC CONSOLIDATION OF DATA BASE CHANGES
RECOVERY FROM DISK I/O ERRORS
DATA BASE RECOVERY UTILITY
RECOVERY FROM ABNORMALLY TERMINATED TASKS
DATA BASE BACKOUT UTILITY
DATA BASE BACK-UP
DL/I SYSTEM LOG
CURR. ACCUM. CHANGE TAPE
RECOVERED USER DATA BASE
BACKED OUT USER DATA BASE

1. DUMP DATA BASE PERIODICALLY USING DATA BASE DATA SET IMAGE COPY UTILITY
2. DURING PROCESSING, DL/I LOGS CHANGE DATA FOR SEGMENT REPLACE, INSERT, AND DELETE
3. DATA BASE CHANGE ACCUMULATION UTILITY IS USED PERIODICALLY TO SELECT DATA BASE CHANGE RECORDS FROM DL/I LOG, AND SORT BY DATA BASE AND DATA SET
4. DATA BASE CHANGE ACCUMULATION UTILITY MERGES SORTED, SELECTED CHANGE RECORDS WITH PRIOR ACCUMULATED CHANGES, KEEPING MOST RECENT CHANGES AGAINST SPECIFIC SEGMENTS
5. WHEN RECOVERY OF DATA BASES FOLLOWING DISK I/O ERRORS IS NECESSARY, MOST RECENT ACCUMULATED CHANGES SINCE LAST BACKUP TAKEN ARE MERGED WITH BACKUP COPY
6. DL/I LOG IS USED TO UPDATE DATA BASE WITH CHANGES SINCE LAST CHANGE ACCUMULATION RUN
7. IF DATA BASE BACKOUT NECESSARY TO REMOVE EFFECT OF PARTIALLY-COMPLETED PROGRAMS, DL/I LOG IS USED WITH BACKOUT UTILITY BEFORE ANY DATA BASE RECOVERY ABOVE IS CARRIED OUT

Figure 8-12. DL/I Data Base Recovery

- When recovery is necessary following I/O errors, read the most recent backup copy of the data set to be restored, and merge the accumulated changes (thereby reloading a partially restored data set). Read log tapes not included in the most recent accumulated changes, and update the data set to the point at which the error was detected. These functions are performed using the data base data set recovery utility.

All updates to the data set at recovery time may be applied from the log tapes rather than from the sorted accumulated changes and the log tapes if desired. Hcwever, cccasional data set dumps will reduce recovery time, by allowing all accumulated changes which precede the dump or reload tc be dropped from the sorted change accumulation tape.

Batch DL/I Data Pase Backout

When the status cf a data base is not clear because of a system failure or because the batch programs that were updating the data base terminated abncrmally, the data base backout utility may be used to back out the effect of those programs. This utility reads backward the log tapes created by the processing of the offending programs. Using the data base log records thus read, it restcres the data base to its status at the time abnormally terminated programs were orginally scheduled. It alsc creates a log tape that must be used as input to any future recovery operation.

Refer to the *DL/I Utilities Manual* for DL/I DOS/VS or the *IMS/VS Utilities Reference Manual* fcr further information on DL/I reccvery.

BACKUP DESIGN

Procedures have been described in this chapter for the backup of:

- Programs
- Terminals
- Data base devices
- Extrapartition devices

Facilities are provided by CICS/VS to allow the warm start (after controlled shutdcwn) or emergency restart (following an uncontrolled shutdown) of:

- Various CICS/VS system tables
- Tempcrary stcrage data identifications
- Transient data destinaticn ccntrcl table
- File ccntrcl data set backout
- DL/I data base backout

Procedures have also been described which can be utilized through user-written programs fcr:

- Transient data extrapartition data set recovery
- Transaction reinitiaticn cn restart

The function of the system design team is tc examine each of the areas identified above, tc determine whether the facilities available through CICS/VS and DL/I are satisfactory for backup and recovery, and to identify whether additicnal procedures or programs should be developed. Regardless of how simple, or complex, the online application may be, each of these factcrs must be considered tc determine whether a backup or recovery problem exists for the cnline applications.

BACKUP PROCEDURES

Standard backup procedures must be defined for each potential failure situation. For example, action by the master terminal operator to carry out program recovery, terminal recovery, or data set or disk pack recovery must be completely described.

The procedures to be carried out by persons such as the system operator in running DL/I data base recovery utilities must be defined, and any additional procedures for recovery of file control data sets and transaction restart (if used) must be identified.

The most critical time in an online application operation occurs when the system terminates abnormally. Because of the need to reestablish online application execution as quickly as possible, every person involved in the reestablishment of the CICS/VS application must know exactly what is required of him and must be well versed in the backup procedures defined by the design team.

To ensure proficiency in the use of these backup procedures, it is absolutely essential to carry out regular backup "fire drills," either at scheduled or unscheduled times, and evaluate the performance of each person involved in the backup. A wrong action made during system restart may completely invalidate the restart process and the data base recovery, with disastrous consequences for the online applications and the installation.

FAILBACK DESIGN

Fallback design refers to the design of the online applications such that information necessary to fall back to a lower level of operation is created during normal execution of the online application. An example of this may be the periodic printing of information to be utilized in a manual fallback environment for answering inquiries. As part of the backup procedures defined by the design team, each level of system fallback should be identified. For example, in the case of an installation having duplex CPUs, the switchover procedures to a backup CPU should be fully described.

However, even with two CPUs there is a possibility of both CPUs going down at the same time (resulting from a power outage, for example). Accordingly, the next level of fallback after a backup CPU also becomes unavailable is generally manual fallback procedures.

The design team should develop procedures for each level of fallback, and ensure that all personnel are completely versed in the implementation of those procedures.

CUTOVER DESIGN

The most critical period in the implementation of online applications is the time of initial cutover from the previous procedures used for those applications to the newly developed online application procedures. It is at this time that the online application is most vulnerable. Regardless of the amount of program testing and system testing carried out prior to cutover, bugs will invariably still be present in programs, either because of data inconsistencies in input of which no one was aware, or because of application and program requirements that were either poorly designed or incompletely tested.

In addition, regardless of the amount of operator training carried out prior to cutover, system operators, master terminal operators, and user terminal operators may not be confident of their ability to function correctly in an online application envircnment. They have not had an opportunity tc develop full confidence in their own ability, or in the ability of the designed system to satisfy different requirements.

Tc minimize the effect cf such problems arising at cutover time, a number of techniques may be used. The most commonly used and generally satisfactory technique is that of parallel processing. In effect, this involves the double processirg of information using the previous application procedures as well as the new online application procedures. In the event of sericus prcblems develcping in the new online applications, the previous procedures may still be utilized to ensure that necessary actions fcr the application are carried out. While this does involve duplicate processing, parallel processing provides a form of insurance against cutcver prcblems.

Accordingly, the cutover procedures must be designed before program development commences. This is necessary, as it may be required that additional facilities be designed into the system to provide for parallel processing, and alsc prcduce information which can be used to check the accuracy of online application processing against the accuracy of the prior application processing.

The irportance cf complete design for backup procedures, fallback procedures, and cutover prccedures cannot be too greatly emphasized. Unless these procedures are designed as part of the normal online applicaticn, the system eventually developed may be workable only when no unusual problems or device failures occur.

Lack cf backup, fallback, or cutover design will drastically affect the availability of the cnline system and application. Regardless of how good a terminal response is provided, the online application may be considered a failure if it does not exhibit high availability owing to poor backup, fallback, or cutcver design.

Chapter 10 outlines additional functions which must be carried out during online cutover and subsequent fcllowup, and which should be considered during initial cutover design.

CHAPTER 9. CICS/VS TESTING AND INTEGRATION

This chapter introduces techniques for testing and integrating CICS/VS application prcgrams tc produce an operaticnal online system. Experienced CICS users may wish to omit reading this chapter, except for the section "Sequential Terminals," which describes changes in this support in CICS/VS from that provided in previous versions of CICS.

The testing of individual CICS/VS application programs, and the subsequent integration and system test of the entire online applications, must be considered by the design team. CICS/VS facilities to assist in testing and system integration will now be discussed.

TRACING AND DEBUGGING

CICS/VS permits the execution of application prcgrams to be traced. Information relating to the use cf each CICS/VS management routine by the application program is recorded by CICS/VS in a wraparound trace table in the CICS/VS nucleus. In addition, application programs may utilize trace control macro instructions to insert their own trace entries in the table. When the end of the table is reached, trace entries at the beginning of the table are overwritten by the trace control prcgram. Trace activity may be turned on or off during CICS/VS execution by the master termibal operator. (See CICS/VS System Administrators Guide, SH20-9006.)

CICS/VS permits trace entries tc be time stamped and written to tape; a program supplied by CICS/VS lists this tape offline in a formatted form for debugging purposes. Programs may also be developed by the user to analyze the time-stamped trace entries for perfcrmance evaluation. This analysis may consolidate all of the trace entries relating to one task in a combined task report, cr may determine the elapsed processing times of all tasks initiated from a specific terminal or by a specific transaction code. These elapsed processing times may be statistically analyzed to determine the average, standard deviation, and 95 percentile values. Similarly, a statistical distribution of the peak number cf tasks in the system dynamic stcrage usage over a defined period of time can be developed by user analysis programs.

Analysis of the trace tape is invaluable for debugging purposes and for evaluation of the performance of existing CICS programs. Potential performance bottlenecks can be identified and conditions rectified.

In addition to utilizing trace entries, application programs may issue dump control macro instructicns to dump selected parts of applicaticn prcgrams, all the stcrage associated with a task, or the entire CICS/VS nucleus to disk. These dumps on disk can be printed at the end of testing, cr concurrently with testing if the dump data set is switched to the alternative dump data set. Refer tc "Program Backup" in Chapter 8 for more information relating to the use of the dump data set in this environment. During testing, the CICS Online Test/Debug Installed User Prcgram (Prcgram No. 5796-AE6) may be used. Source program errors identified by the Online Test/Debug program may be corrected online by using the CICS Scurce Program Maintenance Online Field Developed Program (Program Nc. 5798-BDT) to update a copy of the source program maintained on disk.

CICS/VS enables terminal transactions to be entered from a "simulated" sequential terminal such as a card reader, magnetic tape, or disk. The terminal output may be directed to a line printer, tape, or disk. Through the use of simulated sequential terminals, large volumes of test data may be prepared for what is effectively batch-type testing. No online terminals need be used or even be present on the system. Instead, a card reader and line printer, and/or magnetic tape, and/or magnetic disk may be used for input of simulated terminal transactions, and receipt of the output responses from the application programs.

A simulated terminal transaction may be delimited by a user-defined end-of-data code in a card, or by a code in a tape or disk record. Several card, tape, or disk records may be read by CICS/VS until this end-of-data code is recognized or the input area is full. Large terminal input messages may be simulated by CICS/VS for testing in a batch-type environment.

The use of CICS/VS terminal device-independence enables simulated terminal messages from card reader, tape, or disk "terminals" to be presented to application programs as if they had been read from online BMS-supported terminals.

The output responses from the application program are directed to the simulated sequential output terminal, which may be a line printer, another tape, or another disk data set. Again, these output messages are prepared by the application program as if they are to be transmitted to a particular BMS-supported terminal, and are then converted by CICS/VS device-independent routines for output to the appropriate simulated terminal.

Because testing is carried out in a batch-type environment using sequential terminals, all test data provided must anticipate the requirements of the application program for input. This is necessary, because there is no conversational capability available in testing with sequential terminals. Consequently, conversational error correction carried out by the application program must be simulated by successive sequential terminal transactions.

The CICS/3270 Simulator Field Developed Program (Program No. 5798-AXC) may be used to test 3270 device-dependent characteristics using sequential terminals.

SINGLE-THREAD TESTING

The first stage of testing involves the separate execution of each application program. Only one task is active at a time, and each program can be tested without considering its interaction with other concurrently executing programs. This is achieved by using only one simulated sequential terminal. As each transaction is read from that single terminal, the appropriate program is initiated and the transaction is processed as a separate task. When that transaction completes execution, the task terminates, and the next terminal transaction is read and initiated.

Single-thread testing allows the logic of application programs to be debugged without complications caused by the concurrent execution of other programs.

The next stage after all application programs have completed single-thread testing is multithread testing.

MULTITHREAD TESTING

Multithread testing is equivalent to single-thread testing, except that several sequential terminals are utilized for entry of simulated terminal transactions. Each sequential terminal will result in the initiation of one task at a time to process each transaction read from that terminal. Not until a transaction has been completely processed will the next transaction be read from that terminal and another task initiated. However, at the same time, tasks may be initiated to process transactions from the other sequential terminals. These programs from each sequential terminal execute in a limited multitasking environment, the number of concurrently executing tasks being equal to the number of simulated sequential terminals.

A series of simulated sequential terminals may be set up, either using several card readers and printers, or, more commonly, using many tape and disk simulated terminals.

An approach which may be utilized in this multithread testing environment is to take the test stream which was used for single-thread testing, and copy that test stream to each separate sequential terminal on disk or tape, at the same time randomizing the sequence of transactions in the separate copied test streams. A number of sequential terminals may then be made active to allow multitasking execution to be tested. The output from each terminal should be identical to the output for each transaction as a result of single-thread testing.

The CICS Network Activity Simulator Field Developed Program (Program No. 5798-CCH) may be used to assist in multithread testing with sequential terminals. Refer to "Related Publications" in the Preface for relevant publications.

SYSTEM TEST

System testing generally involves the progressive checking out of all program logic and application functions, to ensure that all elements of the online application, and all programs, fit together and execute correctly.

TOPDOWN TESTING

In Chapter 2, the technique of "topdown" programming was discussed. Briefly, this involves developing the programs at the highest level first, linking to lower level programs which are initially implemented by dummy programs. These dummy programs note the fact that control was passed to that lower level program and return control to the next higher level program. Consequently, the highest level programs are developed and tested first, and then successively lower level programs. The need to develop higher level test drivers (which is a problem with the traditional bottom-up coding and testing technique) is avoided, since the highest level routines provide the system integration and also function as test drivers for lower level programs. As program testing progresses, system testing is also carried out automatically. Eventually, programs at the lowest level are coded and tested.

Because the development and testing of programs proceed in a topdown fashion using the structured programming technique, the problems and delays involved in having to move back to lower levels to check the correction of bugs with the traditional bottom-up method of programming are not apparent. Highest level programs and the entire system integration are tested out first, and then successively lower level programs are tested. As bugs are found in lower level programs, they will usually only affect those lower level programs.

Topdown programming and testing ensure that program interfaces are fully defined, coded, and tested before lower level programs are developed. Consequently, the interfacing of all the different components of the online applications is made easier.

CHAPTER 10. CICS/VS PRODUCTION CUTOVER AND FOLLOWUP

This chapter discusses factors which should be considered by the system design team for cutover of the online applications to live operation, and their subsequent followup to evaluate the effectiveness of the operational applications. Experienced CICS users may wish to omit reading most of this chapter.

CUTOVER

As mentioned, the cutover period of online application development is the most critical period of all. Regardless of how thorough program and system testing may have been, bugs will invariably occur at this time, with possible disastrous consequences. Similarly, regardless of how thorough operator training has been, during the cutover period operators may not have sufficient confidence in themselves, or the system, and may make unnecessary mistakes.

Accordingly, to minimize the effects of cutover problems, it is generally recommended that cutover to the online application be made in parallel with normal prior application procedures. In this way, if serious problems develop, the prior application procedures can continue without disruption to the application. In addition, the results of the online application execution can be compared with the normal results of the prior application procedures for further evaluation of the effectiveness of the online system.

TERMINAL OPERATOR TRAINING

Complete training must be given to terminal operators prior to cutover, to ensure that they are completely familiar with all procedures required of them during online application execution. These procedures should be documented in a Terminal Operator's Manual, which should include standard CICS/VS terminal operating procedures and transactions (such as the CICS/VS operator signon procedure), as well as the particular terminal operating procedures and transaction formats to be used for the online applications. All application error messages should be fully identified in this manual, together with the cause of each error and the action required by the terminal operator to recover from that error situation. The use of a "HELP" transaction, to enable terminal operators to obtain more information relating to errors, may be considered in addition to, or as an alternative to, a detailed list of all error messages in that Terminal Operator's Manual. Refer to "Error Correction" in Chapter 3 for more detail.

Procedures should be defined for the terminal operator to enable him to recognize system, terminal, or line failure situations. The corrective action or backup procedures which he should follow must be completely described. The effectiveness of terminal operator training is measured not only when the system is running correctly, but also when problems occur, or when system downtime occurs. If the terminal operators are sufficiently well trained to recognize these situations, and know exactly what to do to recover from the situations so that there is no panic, terminal operator training has been adequate. As mentioned elsewhere in this publication, the effectiveness of online application design, development, and training can best be judged when error situations arise.

MASTER TERMINAL OPERATOR TRAINING

The master terminal operator must receive all the training given to the other terminal operators, in addition to extra training for general administration of the CICS/VS system and entry of all master terminal commands. The master terminal operator has the overall responsibility for the correct functioning of the system and must be able to recognize failure conditions quickly and take the necessary action.

It may not be immediately evident to him that a failure situation has arisen. Certainly if the CPU goes down, this situation is immediately evident. However, a similarly serious failure situation may occur because of continuous I/O errors on critical terminal lines, or on critical data sets or data bases. This may not be immediately apparent, and in fact the online application may be able to continue operation with that failure situation, even though it may be operating in a degraded mode. However, the degree of degradation owing to error recovery procedures may make the online application unworkable. In this situation the master terminal operator must be sufficiently well trained to recognize the problem and be aware of the corrective action he must take.

SYSTEM OPERATOR TRAINING

The system operator must be completely trained in CICS/VS initiation and termination, as well as in the CICS/VS warm start procedure.

Provision is made by CICS/VS to enable a tailored CICS/VS system to be generated. This tailored system may be initiated, or specific functions may be overridden or replaced, during system initialization. Different versions of tables may be utilized at system initialization to support different terminal configurations, data set configurations, transaction codes, or programs on different days if necessary, or in the event of failure situations. The master terminal operator specifies the particular initialization requirements to the system operator, if the standard application system is not to be initiated. The system operator should normally not be required to interact with CICS/VS except for system initiation. The master terminal operator normally specifies through a master terminal command when the system is to be terminated. The system operator is therefore able to concentrate on other concurrently executing batch applications.

Particular care should be observed by all personnel during cutover. With the parallel processing of prior application procedures with the online application procedures, personnel must be fully trained in the use of additional information produced by the online application to aid in parallel cutover, or in the checking of online application results against results from the prior application procedures.

FOLLOWUP

Once cutover has been successfully made, and the prior application procedures are stopped to enable the online application to assume full control, regular followup must be made. This followup comprises a number of factors, such as:

- Accuracy of operation
- Online application user satisfaction
- Statistics evaluation

  - Performance evaluation

- Accuracy of operation

  During parallel testing, the accuracy of the online application should be fully checked. Procedures should be developed during system design to allow this accuracy to be further spot-checked at various stages after cutover, to ensure that no changes have occurred in the applications or data to invalidate particular application programs or system design. Procedures should be initially designed into the online application to ensure that necessary information to allow this checking will be provided.

- Online application user satisfaction

  Regardless of how well an online application operates, or how satisfactory a terminal response is provided, the ultimate evaluation of the usefulness of those online applications comes from terminal users. As mentioned at the beginning of this publication, to ensure that application requirements are fully met, a member of each particular user department which will use the online applications should be present at certain phases during system design. Output produced by the system, and the information which the user department must be able to enter into the system, must be meaningful to that user. Proper participation of user departments during the initial design phase will ensure that the subsequently developed system meets their requirements.

  During the training of user terminal operators, any areas overlooked during the initial design phase may become obvious. Although this is very late in the development cycle, depending upon the modifications necessary and the urgency of providing those modifications, action may be taken at that time to increase the usability of the system to the terminal operators.

  The design team should be aware that, regardless of how well a system may be designed, and even with considerable user department participation during design, it is often not until user departments have had some operational experience with the online application that they realize that certain improvements may be desirable. Their understanding of their own requirements prior to cutover of the online application may differ from their requirements after they have had some experience with the online application.

  Modifications identified through actual terminal experience in a live environment may be incorporated into the system in a future version. The design, development, and testing of this future version must be carried out using the design techniques already discussed.

- Statistics evaluation

  Statistics provided by CICS/VS, either on individual request by the master terminal operator, or periodically on request, or automatically at system termination, should be accumulated historically. Through user-written programs to analyze these statistics or through use of the CICS/VS program supplied to edit periodic statistics, the performance of the system over a period of time may be evaluated. (See "Performance Monitoring" in Chapter 7.) For example, generally the transaction volumes in an inquiry application will rapidly increase once the system is operational, particularly if the inquiries are providing useful information to the user departments. This increase in transaction volumes may have an effect on the overall performance of the online system. Historical analysis of the CICS/VS statistics enables these situations to be identified, and action to be taken, before the overall performance is drastically affected.

- Performance evaluation

  Performance evaluation may be based upon the CICS/VS statistics taken at particular times or upon the historical statistics analysis described above. Using the information provided by the CICS/VS statistics, it may be desirable to change the allocation of various data sets on disk, to reorder some of the CICS/VS tables (such as the PCT or PPT, for example, to ensure that high activity transactions and programs are near the head of the table and so reduce the amount of table searching), or to increase the maximum number of tasks allowed to execute concurrently.

  In addition, it may be desirable to increase the total CICS/VS virtual partition size to enable more dynamic storage to be utilized. This may increase the CICS/VS working set and hence contention with other partitions for the available page pool, and may result in increased paging with a consequent effect on online performance. The master terminal operator may dynamically control the size of the CICS/VS working set, and hence real storage contention, by varying the MAXTASKS value if more than one task may be concurrently executed. Refer to "CICS/VS Working Set" in Chapter 7 for more detail.

  CICS/VS is generally given the highest priority partition, so that increased paging may be reflected in reduced throughput for concurrently executing batch partitions. If the amount of dynamic storage which must be allocated for CICS/VS increases significantly, so as to cause sufficient paging to seriously degrade lower priority batch partitions, it may be necessary either to reduce the number of batch partitions executing concurrently with CICS/VS, or to increase the real storage size of the machine and so increase the available page pool. (Refer to "CICS/VS Working Set" for further discussion.)

  The flexibility available to the CICS/VS installation through the use of virtual storage enables CICS/VS online applications to be tuned for satisfactory operation, provided the system design has been carried out intelligently and effectively. The ability of a virtual storage system, and of CICS/VS, to dynamically adjust to changing work loads enables satisfactory online performance consistent with batch performance to be achieved.

  When the available page pool is increased by the installation of additional real storage, that real storage is made available not only for extra online performance, but also for extra batch processing performance. Real storage in a virtual storage environment is a performance resource, rather than a critical machine resource without which the applications may not have been able to be executed.

CHAPTER 11. CICS/VS APPLICATION DESIGN

This chapter cutlines a number cf typical CICS/VS applications in several industries. Its objective is to identify common functions, data sets or data bases required, and the online and offline programs which may be needed. Guidelines are presented tc assist the user in the selection of the most appropriate data base support for his installation.

Because of differences in usage of applications across various installations, this chapter does nct attempt to develcp complete application designs. It identifies only the most common problems and approaches, and is intended as a starting point for more detailed application design by the user.

Only those applicaticn areas relevant to the user's own industry need be read in detail. Hcwever, review of cther industry applications can be useful in identifying solutions to application design problems.

As discussed in Chapter 1, the logical starting point for the system design of an online application is tc define the application requirements, objectives, and system flow, and outline the programs and data sets or data bases required. This phase is referred to in this publication as Application Design.

Obviously, it is not possible tc discuss all data processing tasks that lend themselves to cnline operaticn. However, by examining the design of a number of representative applications, a fundamental approach to application design can be defined. At the same time, requirements which are peculiar to a particular application, and which require specific attenticn during system design, can be identified.

The applicaticns discussed in this section are introduced conceptually in the *CICS/VS General Information Manual*. The following topics describe these applications in more detail and use these applications to illustrate various design approaches for systems and applications to be executed under the control of CICS/VS.

Specific design techniques relating to data base support are described for certain applications. In most cases these techniques are applicable nct only for the applications to which they refer, but also to most of the other applications discussed below.

MANUFACTURING INDUSTRY

In the manufacturing industry, a typical online application may provide the following functicns:

- Add new manufacturing crders
- Locate orders in prccess
- Provide order status
- Monitor manufacturing crders in the shop

This online application may be referred to as an online manufacturing work order system or as a production order and status reporting system.

CICS/OS/VS supports the IBM 3850 Mass Storage System. This system allows large data bases to be accessed online for applications with low transaction volumes which can tolerate low response times. (See "CICS/OS/VS - 3850 Mass Storage Operation" in Chapter 7.) Typical applications are the online residence of engineering drawing data bases and technical reports to be retrieved using the STAIRS/VS Program Product (Program No. 5740-XR1). See *STAIRS/VS General Information Manual*, GH12-5114, for additional information.

PRODUCTION ORDER AND STATUS REPORTING SYSTEM

This online application provides manufacturing management and shop supervision with accurate, comprehensive reports concerning:

- Order location
- Schedule viability
- Status condition

Figure 11-1 illustrates this online application. The application has the capability to:

- Add new manufacturing work orders
- Change existing manufacturing work orders
- Split a manufacturing work order
- Report the status of a particular work order
- Provide a general inquiry capability

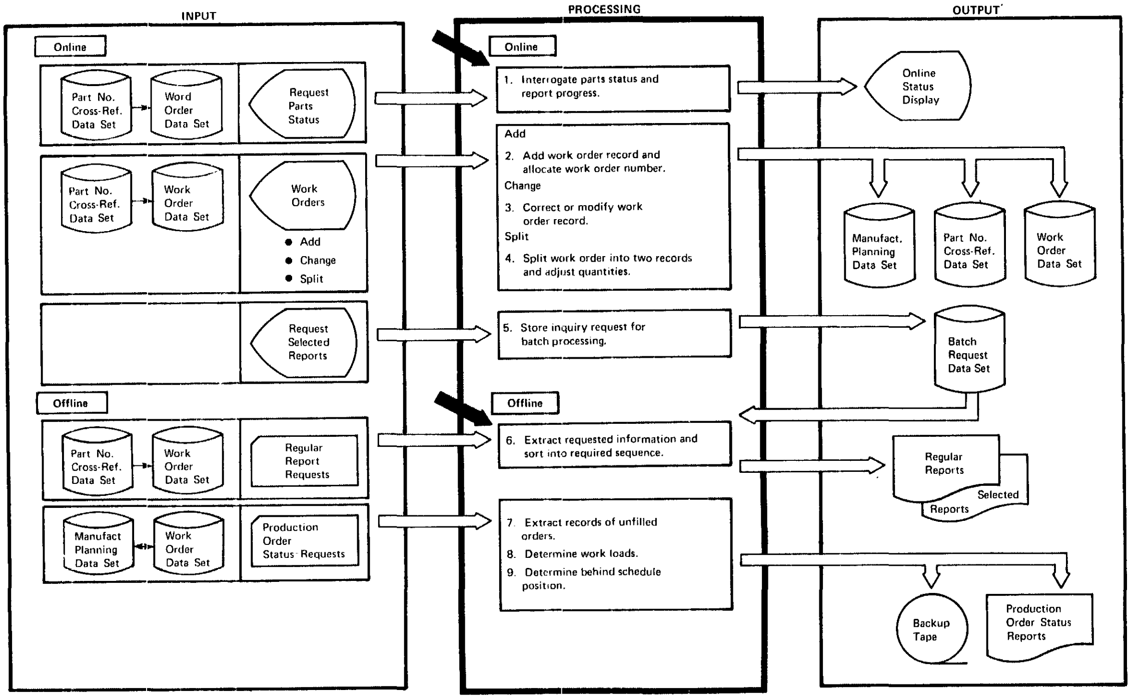

INPUT
PROCESSING
OUTPUT
Online
Part No. Cross-Ref. Data Set
Word Order Data Set
Request Parts Status
Part No. Cross-Ref. Data Set
Work Order Data Set
Work Orders
● Add
● Change
● Split
Request Selected Reports
Offline
Part No. Cross-Ref. Data Set
Work Order Data Set
Regular Report Requests
Manufact Planning Data Set
Work Order Data Set
Production Order Status Requests
Online
1. Interrogate parts status and report progress.
Add
2. Add work order record and allocate work order number.
Change
3. Correct or modify work order record.
Split
4. Split work order into two records and adjust quantities.
5. Store inquiry request for batch processing.
Offline
6. Extract requested information and sort into required sequence.
7. Extract records of unfilled orders.
8. Determine work loads.
9. Determine behind schedule position.
Online Status Display
Manufact. Planning Data Set
Part No. Cross-Ref. Data Set
Work Order Data Set
Batch Request Data Set
Regular Reports
Selected Reports
Backup Tape
Production Order Status Reports

Figure 11-1. Manufacturing Production Order and Status Reporting System

DATA SETS

Three data sets are required by this application. One contains the manufacturing work orders, which are the heart of the application. This data set contains a record for each manufacturing order. A typical record format for this data set is shown in Figure 11-2, which details various information to be kept for the manufacturing order, together with the last reported status of the order. Following this, there may be a variable number of status transactions against this order, other than the last reported.

Work Order No.	Split No.	Planning Dept.	Date of Issue	Lot No.	In-Work Date	Complete Date	Part No.	Initial Stores	Final Stores	Qty.

Authorization	Model No.	Next Assy. Part No.	Material Type	Quantity Complete	Last Status Of Work Order	Last Status of Work Order Numbers							
						No.	No.	No.	N	No.	No.	No.	No.

Figure 11-2. Manufacturing Work Order Record Format

The second data set is a part number cross-reference data set. This relates a part number to all open manufacturing orders for that particular part. Figure 11-3 shows a typical part number cross-reference record, which illustrates a multiple number of manufacturing work orders which use that part.

Part Number	Manufacturing Work Order Numbers										
	No.	No.	No.	No.	No.	No.	No.	No.	No.	No.	No.

Figure 11-3. Part Number Cross-Reference Record Format

The third data set is the manufacturing planning data set, which contains a record for each manufacturing order that has been planned and that will be released to the shop floor for work. Figure 11-4 shows a typical record format for this manufacturing planning data set. This data set is required for audit trail and control functions.

Program	Project	Part No.	Model	Unit	Release Number	Work Order No.	Quantity	Date/Time Sequence Number

Figure 11-4. Manufacturing Planning Record Format

The most suitable data base support for these three data sets will be discussed shortly. However, before a decision is taken on the particular data base support, the application design should broadly define the various programs which are to process the input from terminals for this applicaticn. Their function is shown in Figure 11-1.

ONLINE PROGRAMS

Add Program - Adds New Manufacturing Orders

The add program generates a new reccrd in the manufacturing order data set for each new manufacturing crder. This record contains the information illustrated in Figure 11-2, less the status fields which are generated at a later time. In addition, the program places the manufacturing crder number on the part number crcss-reference data set.

Change Program - Changes Existing Manufacturing Crder

The change program allows the manufacturing order record to be corrected cr mcdified as required.

Split Program - Splits an Existing Manufacturing Order

It is often necessary tc split a manufacturing crder because of part quantity change requirements with respect to schedule. The split number uniquely identifies each split cf the crder. The split program splits the manufacturing order; that is, it generates a new manufacturing order record with a unique split number. It adjusts the quantity in the two manufacturing order records in accordance with total quantity.

Status Inquiry Program - Repcrts Parts Status

The status inquiry prcgram allows the shcp to report the progressive status of parts as they flcw to various operations within manufacturing. The status is interrogated, and the appropriate action is taken. A status field, which is alsc used in batch programs to produce status reports, is added to the manufacturing order reccrd.

Status inquiry prcgrams may handle the processing of inquiries in both online and batch envircnments. Requests for batch reporting are stored in a temporary data set tc be used in processing regularly scheduled reports.

OFFLINE PROGRAMS

The above programs enable the cnline applicaticn requirements described earlier tc be met. The application also requires a number of additional prcgrams tc select and fcrmat reports in the batch environment.

Report Writer Program

The report writer program selects and formats offline reports periodically. This prcgram sequentially processes the manufacturing order data set, selecting the desired manufacturing crder records for inclusion in the report. Inclusion is determined either on the basis of a request submitted by manufacturing cr on an exception repcrting basis.

Unload and Select Program

The unload and select program unloads the manufacturing order data set periodically and selects the applicable records for input to the production order status report program. As part of this program, a data set statistical report for data set reorganization is produced.

Load Program

The load program reloads the manufacturing order data set, using the backup created by the unload and select program as input. It eliminates any overflow and deleted records.

Production Order Status Report Program

Using data set information provided through the unload and select program, this program produces production order status reports. These reports provide information about all unfilled manufacturing orders in an effort to give the manufacturing department up-to-date information regarding work loads and/or behind-schedule position.

Selected Report Writer Program

The selected report writer program allows manufacturing personnel to vary the range and sorting sequence of several types of reports. For example, manufacturing may request a part number report for a specific department, and for a specific model and unit, and receive only those manufacturing order numbers which apply to those criteria. This program allows manufacturing to produce many unique reports.

DATA BASE SUPPORT SELECTION

The most suitable data base support for the online system should be selected at this time. The factors which can be taken into account in making this decision are described in "Data Base Selection Criteria" in Chapter 5.

DL/I Support

DBOMP (Data Base Organization and Maintenance Processor - Program Product (DOS) 5736-XX4) has been widely used as the main batch data base support in the manufacturing industry. However, it is not the purpose of this publication to discuss the use of DBOMP. DL/I products offer data base support which surpasses that provided by DBOMP in many respects, such as improved data independence and device independence. This data and device independence may result in significantly less program maintenance and data base maintenance than experienced with DBOMP.

The support of logical relationships and secondary indexing (see Chapter 5) by DL/I ENTRY, DL/I DOS/VS, and IMS/VS DL/I, assists in the conversion of existing DBOMP data bases to DL/I.

The Chained File - DL/I Bridge Program Product (Program No. 5748-XX3) can be used for conversion of chained file systems, such as DBOMP, to DL/I DOS/VS or IMS/VS DL/I. Refer to "Related Publications" in the Preface for relevant publications.

The manufacturing order data set may contain a multiple number of status segments (see Figure 11-2). Furthermore, the part number cross-reference data set (see Figure 11-3) contains a segment for each manufacturing order number associated with a particular part number. Since a part may be used with many different manufacturing orders, there may be a very large number of these segments associated with each part number. These data sets are particularly suited to DL/I. A typical logical structure for these data sets is shown in Figure 11-5.

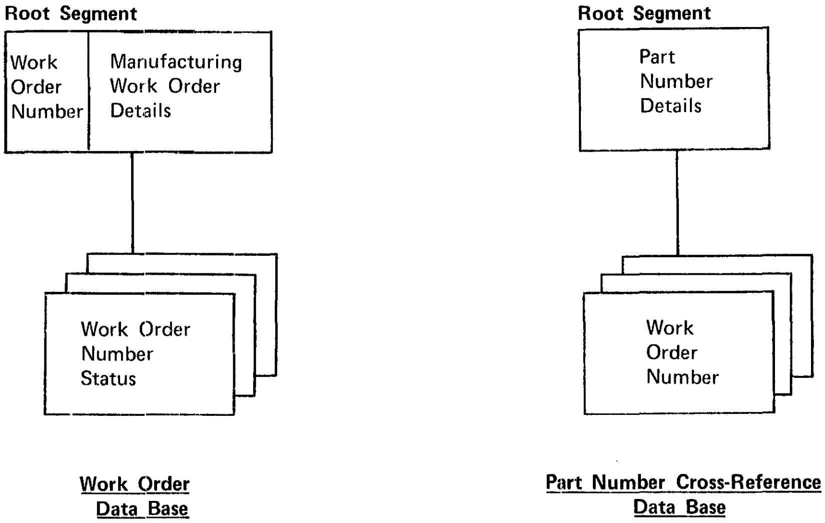


Figure 11-5. Manufacturing Data Base Logical Structures

The manufacturing planning data set (see Figure 11-4) has no dependent segments associated with it and does not require DL/I support specifically. However, DL/I can support it as a root-segment-only data base.

Since the segments in each data base previously described are fixed length, any of the DL/I products may be selected. Addition and deletion activity may be accumulated online in a CICS/VS file control data set, and applied to the data buffer offline.

Refer to "Sample Applications" in _DL/I DOS/VS General Information Manual_ for further discussion of the use of DL/I in the manufacturing industry.

CICS/VS File Control Support

DL/I is a suitable data base support for this online manufacturing application. However, if one of the DL/I products cannot be used, CICS/VS file control support should be selected.

The support of multiple dependent segments in the manufacturing order data set and the part number cross-reference data set can be provided by using the segmented record feature of CICS/VS file control. File control requires a specific number of segments to be defined for each segmented record. The basic information for both of these data sets is placed in the root segment of the segmented records (see Figure 11-6), and additional segments are defined to allow multiple dependent segments.

Root Segment		Dependent Segments								
Work Order No.	Manufacturing Work Order Details	Work Order No. Status Segment								
		No.	No.	No.	No.	No.	No.	No.	No.	No.

Figure 11-6. Manufacturing Work Order Segmented Record Format

CICS/VS file ccntrol can te used to provide a segmented record capability which approaches that of DL/I, but which requires more user programming and subsequent maintenance.

The manufacturing planning data set, which has a root segment but not multiple segments, dces nct require the segmented record feature for support by file control.

The next decision to be taken regarding the data base support for this application is the access methcd to be used ty CICS/VS file control for these data sets.

The three data sets may presently be supported ty either ISAM or DAM in a batch environment. However, because of the above need to use the segmented record feature of file control, the record formats of the tatch data sets will need to te changed for the cnline application. They may be converted and still te supported by ISAM or DAM if necessary.

At this stage it is wcrthwhile ccnsidering the effect of adding manufacturing orders online, changing manufacturing orders and possibly deleting segments or adding segments, or deleting entire manufacturing orders. Because this application reflects high activity of additions, updates, and deletions tc the online data sets, a more suitatle access method to use is VSAM. This enatles records tc te accessed either by key (key-sequenced VSAM), or by relative byte address within the data set (entry-sequenced VSAM), and also readily allcws for additicns to data sets while still maintaining good retrieval performance. Further, because of the ability of key-sequenced VSAM to physically delete segments or records from a data set, or increase or decrease the length of records in a data set, this application is ideally suited to the use of key-sequenced VSAM. However, if satisfactcry performance is to te achieved, VSAM should nct be used on systems with less than 144K of real storage.

If either DAM or ISAM is used, the overflow technique described later in this chapter fcr the banking industry can be utilized to increase the length of segmented ISAM or DAM reccrds. (This technique is described in more detail under "Segment Updating (DAM, ISAM, and Entry-Sequenced VSAM)" in "Segmented Records" in Chapter 5.)

BANKING INDUSTRY

The two principal online applications in the banking industry are:

- Savings Bank and Mortgage Loan System
- Customer information system (called customer information file)

SAVINGS BANK AND MORTGAGE LOAN SYSTEM

This online applicaticn enables saving bank deposits and withdrawals, and mortgage loan transactions, tc be entered frcm teller terminals located in various branches cf a bank. Such transactions are used to update savings bank and loan accounts immediately. Some of the advantages which result from this online applicaticn are:

- Imprcved customer service, enabling the bank's customers to utilize any branch of the bank
- Standardization of procedures across all branches
- Ability to extend all banking services to every teller terminal if required
- Timely account informaticn
- Assistance to tellers in maintaining teller tctals
- Possiblity of improved audit and control procedures

Figure 11-7 illustrates such an online savings bank and mortgage loan application. This application enables the following transactions to be entered:

- Savings bank deposits
- Savings bank withdrawals
- Mortgage loan transactions
- Inquiry transactions

Further, it provides additional facilities for audit purposes in the areas of online operation and offline operaticn.

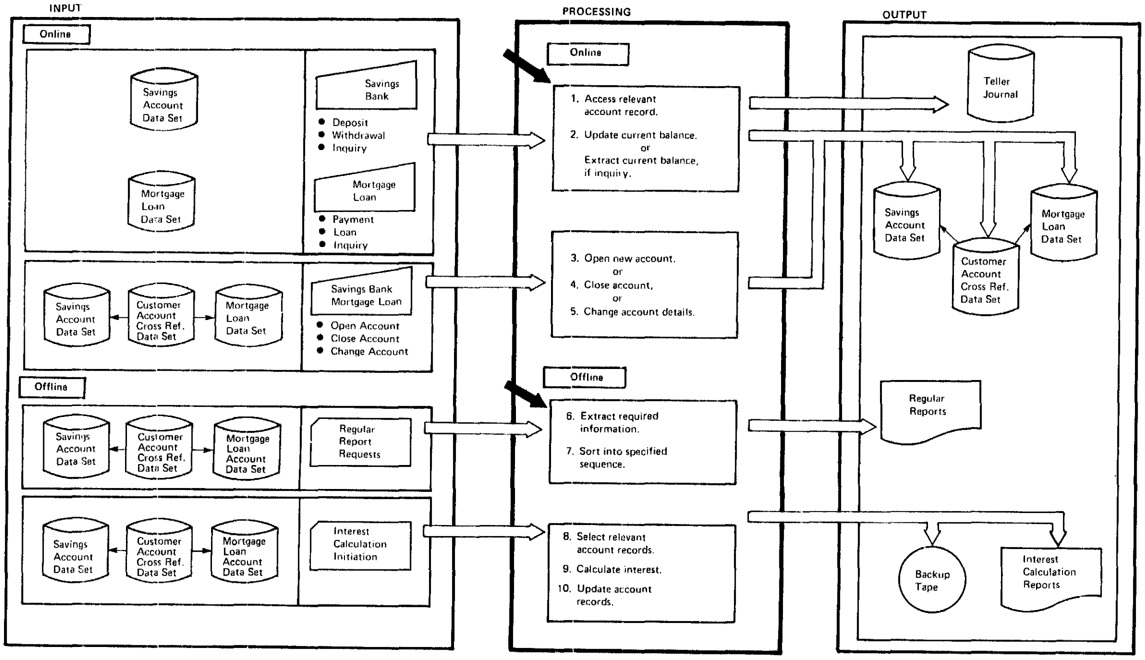

INPUT
Online
Savings Account Data Set
Mortgage Loan Data Set
Savings Bank
● Deposit
● Withdrawal
● Inquiry
Mortgage Loan
● Payment
● Loan
● Inquiry
Savings Account Data Set
Customer Account Cross Ref. Data Set
Mortgage Loan Data Set
Savings Bank Mortgage Loan
● Open Account
● Close Account
● Change Account
Offline
Savings Account Data Set
Customer Account Cross Ref. Data Set
Mortgage Loan Account Data Set
Regular Report Requests
Savings Account Data Set
Customer Account Cross Ref. Data Set
Mortgage Loan Account Data Set
Interest Calculation Initiation
PROCESSING
Online
1. Access relevant account record.
2. Update current balance. or Extract current balance, if inquiry.
3. Open new account. or
4. Close account. or
5. Change account details.
Offline
6. Extract required information.
7. Sort into specified sequence.
8. Select relevant account records.
9. Calculate interest.
10. Update account records.
OUTPUT
Teller Journal
Savings Account Data Set
Customer Account Cross Ref. Data Set
Mortgage Loan Data Set
Regular Reports
Backup Tape
Interest Calculation Reports

Figure 11-7. Savings Bank and Mcrtgage Lcan System

DATA SETS

Several data sets are used. One is the savings bank account data set, which contains a reccrd of each savings bank account and transactions against that account. A typical savings account record is illustrated in Figure 11-8, which shows the multiple occurrence of deposit and withdrawal transactions, for example, against that account. Reference may need to be made to previous deposit cr withdrawal transactions, particularly if those transactions were "nc-book" transactions (initiated withcut a passbook). When the passbook is next presented, these "no-book" transactions must be used to physically update the passbook with the previous "no-book" entries. The record format shown in Figure 11-8 endeavors to keep the transactions against a particular account in close proximity to the acccunt details, for optimum data set accessing time.

Account Number	Customer Number	Account Names	Current Balance	Stops-Flags	Transactions			
					Deposit	Deposit	Withdr.	Deposit

Figure 11-8. Savings Bank Account Record Format

The mortgage loan data set may be similar in structure and organization to the savings account data set, and may use a record format similar tc that illustrated in Figure 11-8. It is organized as a separate data set mainly tc physically separate savings bank and mortgage loan acccunts fcr subsequent batch processing.

The customer account crcss-reference data set is used to open new savings or loan accounts, or close acccunts frcm a teller terminal. Furthermcre, changes in custcmer details, such as a change in address, can be reflected in all the customer's accounts through the use of this data set.

While this data set is cf particular importance for a banking customer information system (see below), it is also useful in the savings and lcan applicaticn for the above reasons. Figure 11-9 illustrates a typical customer account cross-reference record format, which illustrates the multiple cccurrence of acccunt numbers for each customer.

Customer Number	Customer Name	Customer	Details	Address			Accounts				
				Line 1	Line 2	Line 3	Acct. No.	Acct. No.	Acct. No.	Acct. No.	Acct. No.

Figure 11-9. Customer Acccunt Cross-Reference Record Format

A teller journal data set may be used for audit and recovery purposes. Alternatively, the journaling facility of CICS/VS may be utilized by the user to direct teller activity information to a journal data set. If the 3600 system is used, the teller journal should reside on the 3601 diskette. This makes it available fcr both online and offline modes of operaticn.

In an online savings and loan application, each transaction entered from a terminal must be immediately recorded in the event of system failure, as the criginal transaction may no lcnger be available to be reentered. That transaction is represented by the customer at the teller window. Once he has made his transaction, he will take his passbook and leave. Apart from a hard-copy log of all transactions, which is provided on most savings bank and loan terminals, the teller may have no other record of the original transaction.

It is not reasonable to require the teller to reenter transactions from this hard-copy log in the event of a system failure. Instead, on receiving each banking transaction, the system should log that transaction to the teller journal data set, which contains a record for each teller in the bank. If the system goes down before the transaction is completed and added to the customer account record, it is available from the 3601 teller journal on restart and can be retransmitted for reprocessing. The question of how to handle transactions which could not be processed by the system because of system failure is discussed further under "Transaction Recovery and Restart" in Chapter 8.

Figure 11-10 illustrates a possible teller journal record format. As well as containing information relating to the last transaction from that terminal, this journal indicates the current status of the terminal which that teller is using, and maintains cumulative totals for that teller to enable him to balance his cash at the end of the day. This record format may be utilized in a separate teller journal data set on the 3601 diskette, or it may be written by the user to a journal data set utilizing CICS/VS journal control macro instructions.

Teller Number	Teller Status	Teller Totals			Terminal Itentification	Terminal Status	Last Transaction From Terminal
		Deposits	Withdrawals	Cash			

Figure 11-10. Teller Journal Record Format

ONLINE PROGRAMS

A number of online programs may be used in this application environment. These programs may be executed completely in the CPU or 3601, or may be executed partly in the 3601 and partly in the CPU. The most important of these programs are described in this chapter.

Savings Transaction Program - Processes Deposit and Withdrawal Transactions

Deposit and withdrawal transactions may be made in many ways. Some of the possible transaction types may be made as follows:

- With or without a passbook
- By cash, check, or cash and check
- By transfer of funds between accounts
- Against accounts with various stops or holds placed on them

All these transactions generally have the same input format, which may provide more or less information depending on the function required. Furthermore, they generally involve the same logical processing, except that some transactions may add to the current balance (deposits), other transactions may reduce the current balance (withdrawals), while other transactions may have to carry out certain tests first. Finally, the output response for each of these transactions is usually the same, but may require certain fields to be printed in certain columns of the passbook.

While it is possible to write a separate program for each individual type of transaction described above, another solution is to write a generalized program that uses a code entered with the transaction to define the particular transaction type, and then accesses various tables defined by the user to detail, for each transaction type:

- The input format
- Editing and testing to be carried out on the transaction
- Processing and information to be updated on data sets
- The output format to be used

By adopting a tabular approach to the definition of the requirements for each transaction type a user-written application program may be used with generalized logic to examine the information detailed in user tables for each transaction type and carry out the necessary processing. This application program may be developed as a generalized application program block (AP) for execution in the 3601 Controller. A generalized, user-written CICS/VS savings transaction program may then be used by all savings bank terminals for all savings bank transactions as previously described. The use of the 3600 and CICS/VS enables many teller terminals to use this same program concurrently through the use of the multitasking, dynamic storage allocation, and quasireentrant capabilities of CICS/VS and the multiprogram execution capability of the 3600. Through these facilities and a tabular processing approach, most efficient use can be made of the CPU processing capability and storage availablity, to maintain a satisfactory response time at the terminal. This technique is discussed further under "Tabular Structures" in Chapter 2.

Mortgage Loan Program - Process Mortgage Loan Transactions

The tabular approach described above may also be used for loan transactions, if desired. Some of the logic used for the savings bank program may be able to be used for the mortgage loan program. Alternatively, by adding to the savings bank tables to describe loan transactions, and by identifying transactions as relating either to savings or loan, one 3600 AP and one CICS/VS program may be used for both savings bank and mortgage loan transactions.

An advantage in combining these two applications into the one program, and using a tabular approach to describe each transaction type, is that reductions may be made in future program maintenance. If it is necessary to change either the input or the output format, or the processing or editing requirements for any transaction type, that change need only be made in the appropriate table in the 3601, which is separately assembled from the main savings and loan program. Furthermore, by deleting or adding entries in tables, transaction types can be easily removed or added to the application. If a generalized programming approach has been used for the savings and loan program, no coding changes may need to be made, unless a new application function is added which requires unique program code.

Change Program - Change Savings and Loan Accounts

This program is used to open and close accounts by adding or deleting records from the appropriate account data set (savings or loan) and adding or deleting account numbers from the customer account cross-reference data set. The program also may update the customer account cross-reference data set to reflect changes in address or name, for example. A further requirement of the change program is to add or remove stops (holds) to varicus accounts, or change the customer names for an account.

Use of the 3600 permits a subset of the account data set to be stored on the 3601 diskette. This normally would only identify account numbers that require special action, such as a stop on an account. A change to the accounts data set in the CPU by the change program should also be reflected in the subset of the accounts data set in the 3601.

Inquiry Program - Inquire Against Savings and Loan Accounts

This program enables information such as:

- Customer name and address
- Account names
- Stops and holds
- Current balance
- Interest

to be obtained from the varicus cnline data sets tc satisfy various inquiries.

Audit Prcgram - Log Information for Audit and Recovery Purposes

This program is executed in the 3600, and utilizes the user-defined teller journal, which ccntains a reccrd for each teller using the 3601 and logs each savings or loan transaction to this journal on it immediately after the transaction is received by it. In the event of system failure, a user AP can retrieve these transactions from the teller jcurnal, check to determine whether they have in fact updated the apprcpriate account (were not in-flight), and, if they were backed out on emergency restart, the AP can complete the necessary processing for those transactions. This audit function is part of the AP executed in the 3601 for savings and loan transactions.

This program records each change in status of the teller, such as "teller online," and also maintains current cumulative totals of all money handled by the teller during the day, including:

- Total cash withdrawals
- Total cash deposits
- Total check withdrawals
- Total check deposits
- Total balance

Supervisor Program - Provide High-Security Functions

In many cases it may be necessary to override certain functions in the savings and loan application. For example, a stop or a hold may be placed on an account, or an account may be rendered inactive for various reasons. It may be necessary for an authorized person in the bank to override this security provision of the application. An override capability is provided by the supervisor program and may be implemented either in the CPU or the 3601.

OFFLINE PROGRAMS

Several programs select and format reports in the batch environment. These programs are described below.

Selected Report Writer Program

The selected report writer program selects and formats regularly produced reports. This program sequentially processes both the savings account data set and the loan account data set, selecting the desired records for inclusion in reports. The inclusion is determined either on the basis of a request submitted by bank personnel or on an exception reporting basis.

Unload and Select Program

The unload and select program unloads the savings and loan account data sets on a periodic basis, and selects the applicable records for input to the interest calculation program. As part of this program, a data set statistical report for data set reorganization is produced.

Load Program

The load program reloads the savings and loan account data sets, using the backup created by the unload and select program as input. It eliminates any overflow and any deleted records. It may be run overnight, to reorganize data sets or data bases for the next day's online operation.

A subset load program must extract the necessary subset information from the data sets and transmit it to each 3600. A similar 3600 AP will reload the information on the 3601 diskette account data set if necessary.

Interest Calculation Program

Using record information provided by the unload and select program, this program calculates interest for various accounts.

Selected Report Writer Program

The selected report writer program allows banking personnel to vary the range and sorting sequence of many types of reports. For example, bank personnel may request a report of all loan accounts in arrears, or of all savings accounts at a particular branch. This program allows the bank to produce a wide range of unique reports.

Before considering the data base support for the data sets described above, the customer information system application is discussed. This banking application is sometimes referred to as a customer information file application.

CUSTOMER INFORMATION SYSTEM (CUSTOMER INFORMATION FILE)

A customer of a bank may have several savings accounts, checking accounts, mortgage loan accounts, and other accounts (such as bills of exchange, foreign exchange, stocks, and Christmas Clubs). Figure 11-11 illustrates a typical customer informationn system for banking.

CICS/OS/VS supports the 3850 Mass Storage System which can be used to maintain massive customer information system data bases online. It can be used for applications that have low transaction volumes and that can tolerate long response times. A typical 385C application in this industry is the online residence of historical current account (checking) transactions for subsequent enquiry. (See "CICS/OS/VS - 3850 Mass Storage Operation" in Chapter 7 for additional information related to the 3850.)

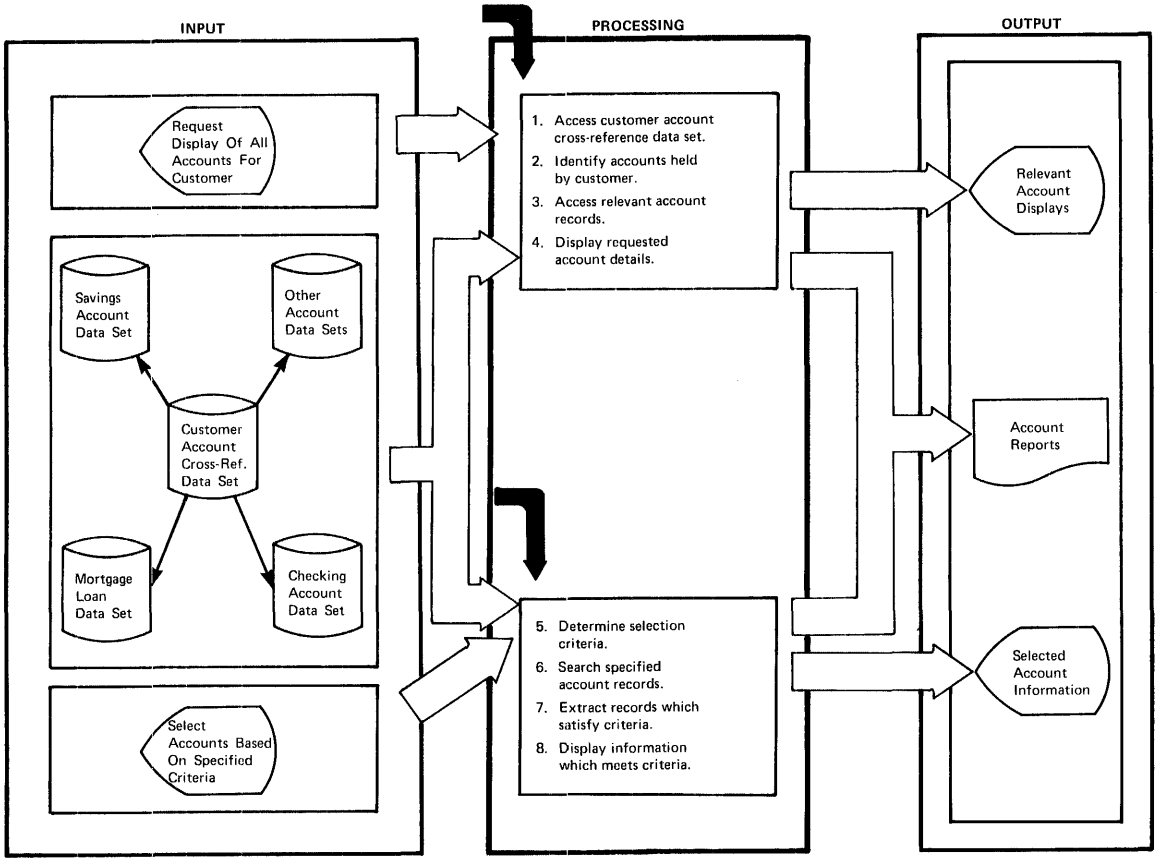

INPUT
PROCESSING
OUTPUT
Request Display Of All Accounts For Customer
Savings Account Data Set
Other Account Data Sets
Customer Account Cross-Ref. Data Set
Mortgage Loan Data Set
Checking Account Data Set
Select Accounts Based On Specified Criteria
1. Access customer account cross-reference data set.
2. Identify accounts held by customer.
3. Access relevant account records.
4. Display requested account details.
5. Determine selection criteria.
6. Search specified account records.
7. Extract records which satisfy criteria.
8. Display information which meets criteria.
Relevant Account Displays
Account Reports
Selected Account Information

Figure 11-11. Banking Customer Information System

In this application, all a customer's activities with the bank can be related. By relating all of a customer's accounts, access to any one account also makes other accounts identifiable and available. Thus a bank is better able to determine the involvement of each customer with the bank. Two of the advantages offered by such a customer information system are:

- Improved customer service

- Increased marketing opportunities for the bank in selling additional services to customers

By accessing the customer account cross-reference data set, information relating to the custcmer can be obtained, together with the account numbers of every other type of account held by a customer at that bank. By accessing the appropriate account data set, more detail may be obtained for each individual account.

Futhermore, by recording other information in each account record, such as the names of various people using a savings account or a checking account, the individual accounts held by each person can be identified by further reference tc the customer account cross-reference data set.

This information, enabling the various banking services used by a customer to be identified, may be used together with other information to open new marketing opportunities for the bank. For example, analyzing information such as the type of services used by customers in particular geographical areas, may identify those services which are most popular in a particular lccation. Furthermore, those services which are less popular can be identified, enabling action to be taken to improve the service cr to reduce costs by removing the service.

Another marketing opportunity which presents itself is to compare those customers in a particular location against a separate data base of all people living in that location, such as from an electoral roll. This enables the bank to identify people who are nct presently customers, thus allowing selective marketing to be carried out to attract more customers to the bank.

Another possibility is to identify various companies cr corporations which are customers of the bank, and then access information which is generally available in most countries identifying the main executives of those companies. If these executives are not presently customers of the bank, the bank may wish tc direct its marketing efforts toward attracting them as new customers.

The various approaches described are directed at improving customer services, reducing costs from unprofitable services, or attracting new customers to the bank. These approaches have always been available to the banking industry. However, the information which was necessary to implement them usually has nct been readily accessible. The use of an online customer information system, with all related customer accounts identified, makes this infcrmaticn available. CICS/VS enables such a customer information system to be readily implemented.

DATA SETS

The data sets used by a customer information system include the savings bank acccunt data set, the mortgage loan acccunt data set, the checking account data set, and other data sets (see Figure 11-11).

The checking account record fcrmat is illustrated in Figure 11-12. This would normally be created and updated offline daily by various check processing and clearing batch programs. It can be seen from the figure that the checking acccunt reccrd format is similar to that used for savings and loan acccunts. In particular, after the fixed checking account information in a root segment, there may be a multiple number of transactions which are associated with each acccunt recorded, each transaction representing a check written, or a check deposit made.

Account Number	Customer Number	Account Names	Current Balance	Stops-Flags	Transactions				
					Deposit	Check	Check	Charge	Check

Figure 11-12. Checking Account Record Format

Mortgage loan account data sets use a record format similar to the savings account data set illustrated in Figure 11-8.

The customer account cross-reference data set is used to contain customer details such as name and address, together with the account number of each account used by that customer as illustrated in Figure 11-9.

A customer information system is not as critically affected by system downtime as the savings and loan application described previously. Accordingly, a teller journal is not essential for this inquiry system and has not been included in Figure 11-11. The terminal operator can reenter his inquiry once the system has restarted, with little or no detriment to the application.

The programs which may be necessary in this application include, but are not limited to, the following.

ONLINE PROGRAMS

Account Reference Program - Identify Various Accounts Held by a Customer

This program accesses the customer account cross-reference data set for a particular customer and makes available at a terminal all account numbers used by that customer. As an option, the terminal operator may examine each individual account in detail.

This program particularly benefits from the terminal paging capability of CICS/VS. As each data set and account is accessed, all the information relating to that account may be retrieved and formatted as a separate terminal page for display if necessary. However, the program itself does not need to control the selection and presentation of pages by the terminal operator. The program formats each page on the assumption that it may be required and presents each page to CICS/VS. CICS/VS then displays each appropriate page as requested by the terminal operator. Refer to "Terminal Paging" in Chapter 3 for more detail.

Account Search Program - Retrieval Based On Selection Criteria

This program enables part or all of the specified data sets to be searched to locate records which satisfy various selection criteria. These criteria may be predefined, or defined by the terminal operator at the time of entering the retrieval request. For example, all checking account customers whose accounts have been below a specified minimum balance for a specified time period may be selected from the checking account data set. These selected records may be presented to the terminal operator, with records satisfying the selection criteria most closely presented first, followed by records satisfying the criteria less closely. In this way, the most delinquent checking accounts can be readily identified.

CICS/VS is particularly suited to this search requirement through the availablity of the built-in weighted retrieval function. This enables a key-sequenced VSAM data set to be sequentially scanned by CICS/VS, selecting records based upon user-supplied criteria. CICS/VS sequences records fcr presentaticn based upon the degree to which they fall between user-defined limits for selection. Refer to "Weighted Retrieval" in Chapter 5 fcr more detail.

DATA BASE SUPPORT SELECTION

Since both the savings and lcan application and the customer information system may ncrmally be simultaneously supported as online applications by CICS/VS, the selection of the apprcpriate data base support should satisfy bcth applications. The factors which should be considered in the selection cf this data base suppcrt are now discussed. The criteria that shculd be used in selection of the data base support are described under "Data Ease Selection Criteria" in Chapter 5.

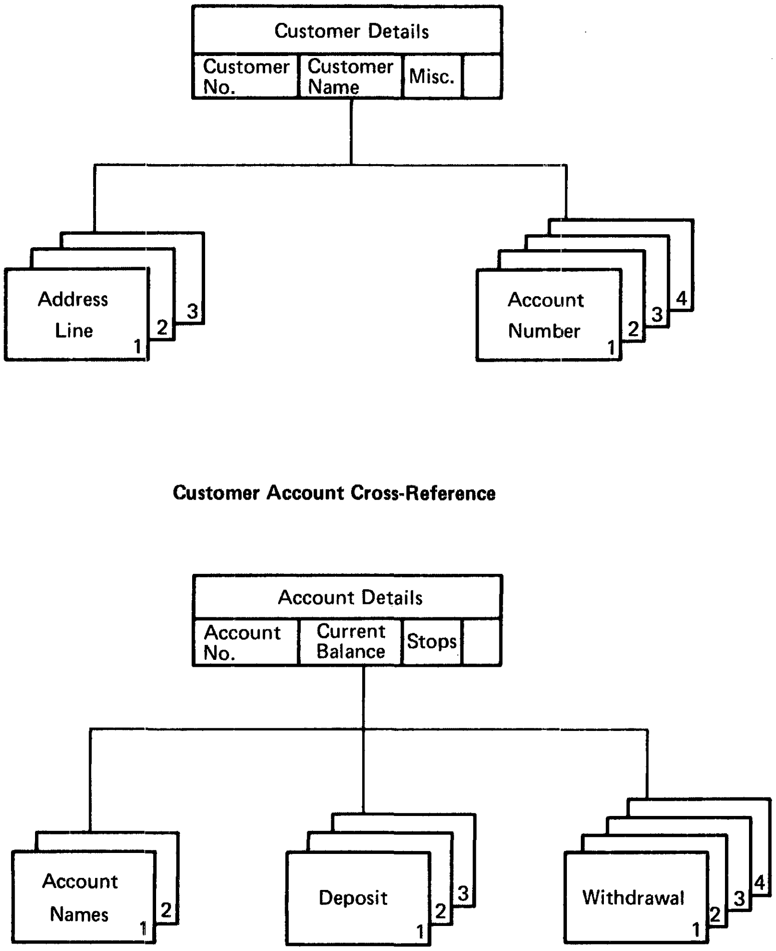
DL/I Support

Each of the data sets used by both the savings and loan application and by the customer information system exhibits a fixed amount of information relating to a particular account. This fixed information may be regarded as a root segment. The root segment may have a multiple number of dependent segments, such as various transactions for savings accounts, checking accounts, loan accounts, or cther accounts.

In the case of the customer account crcss-reference data set, the custcmer details wculd fcrm a roct segment, with each of the account numbers used by that customer being a dependent segment (see Figure 11-13). With the savings and mortgage lcan data sets, each account transaction would be a dependent segment as shown in Figure 11-13.

There may be any number of dependent segments (transactions or account numbers) fcr each roct segment. Because of this, these data sets are ideally suited to the use of DL/I as the data base support. DL/I enatles root segmert infcrmaticn or individual segment information to be retrieved on request by the prcgram and presented tc it for processing with nc ccncern by the prcgram for the physical location of the segments.

DL/I ENTRY, DI/I DOS/VS, and IMS/VS DL/I readily enable the addition, replacement, or deleticn of segments tc be achieved online. The root and dependent segments are fixed length, and consequently can be supported by any of the DL/I products. The support of logical relationships and secondary indexing by DL/I ENTRY, DL/I DOS/VS, and IMS/VS is particularly useful for the design of the customer information system data base. Refer tc "Sample Applications" in *DL/I DOS/VS General Information Manual* for further discussion of the use of DL/I data bases in the banking industry.



Customer Account Cross-Reference

Account Record

Figure 11-13. Banking Data Base Logical Structure

CICS/VS File Control

CICS/VS file control should be considered in this industry by CICS/DOS/VS users with real storage size equal to or less than 160K who do not wish to use DL/I DOS/VS, or by CICS/OS/VS users with real storage size equal to or less than 240K. In this instance, a limited capability similar to that provided by DL/I can be offered by the segmented record feature of CICS/VS file control.

The dependent segments (transactions) are generally small, typically no more than 30 bytes in most cases. Each 30-byte segment can be defined to file control as a separate 30-byte fixed-length segment. A large number of dependent segments (up to a maximum of 99) may be held as CICS/VS segments. For example, by defining ten bytes of segment indicator bit flags, 80 CICS/VS dependent segments can be defined. Each additional byte of segment indicator bit flags will allow a further eight dependent segments to be stored.

Because the segment indicator flags identify the presence or absence of CICS/VS segments, no disk storage is used (apart from the indicator flags) for segments which are absent in a particular data set record. Addition or deletion of dependent segments will change the length of CICS/VS segments. The normal requirement of this application is to add dependent segments, thus increasing the length of CICS/VS segments. However, this capability is not supported by ISAM or DAM.

Accordingly, the access method which should be selected is key-sequenced VSAM, which readily allows for the increase in segment length, or the deletion of segments with the ability to physical reuse that deleted storage. However, VSAM should not be used (for performance reasons) if the user's system has less than 144K of real storage.

If VSAM cannot be used, a technique may be utilized based on a direct access data set for addition or deletion of transactions through a backward chain. In this case, the relative location of the most recent transaction for an account is placed on the root segment of that account's segmented record (see Figure 11-14). This is the overflow technique described in Chapter 5 in the section "Segment Updating (DAM, ISAM, and Entry-Sequenced VSAM.")

Ideally, the teller journal should be a CICS/VS user journal data set, operated on by CICS/VS journal control macro instructions issued by the user. Alternatively, if this approach is not desired, the teller journal data set may be a direct access data set supported by DAM, or an entry-sequenced VSAM data set.

INSURANCE INDUSTRY

The two main online applications within the insurance industry are:

- Policy information system
- New-business policy entry system

The advantages of these online applications in the insurance industry are:

- The availability of policies to all offices of an insurance company
- The ability to maintain close control of the claims made by a customer against his policies
- The potential for reduction in cost of entering new-business policies
- The potential for improvement in accuracy of new-business policy entry

POLICY INFORMATION SYSTEM

A policy information system is analogous to a customer information system in the banking industry. This policy information system uses a policy data base which contains all the policies issued by the insurance company. All the policies held by each customer are identified and related to that customer. By identifying the customer, or a policy owned by that customer, all other policies belonging to that person may be identified. In this way, the customer's worth to the insurance company can be more fully assessed.

However, it is not sufficient to know only what policies are held by a particular customer. It is also important to know the claims which have been made against those policies and the current status of premium payments. Figure 11-14 illustrates a policy information system for insurance companies.

CICS/OS/VS supports the 3850 Mass Storage System which can be used to maintain policy information system data bases online. The 3850 supports data bases ranging from 35 to 236 billion bytes. It can be used for applications that have low transaction volumes and that can tolerate long response time. (See Chapter 7 for additional information related to the 3850.)

The insurance data base shown in Figure 11-14 enables information to be retrieved in a number of ways, such as by:

- Policyholder name and address
- Policy number
- Policy claims
- Policy renewals
- Agent's identification
- Line of business (Policy Class)
- Geographic territory

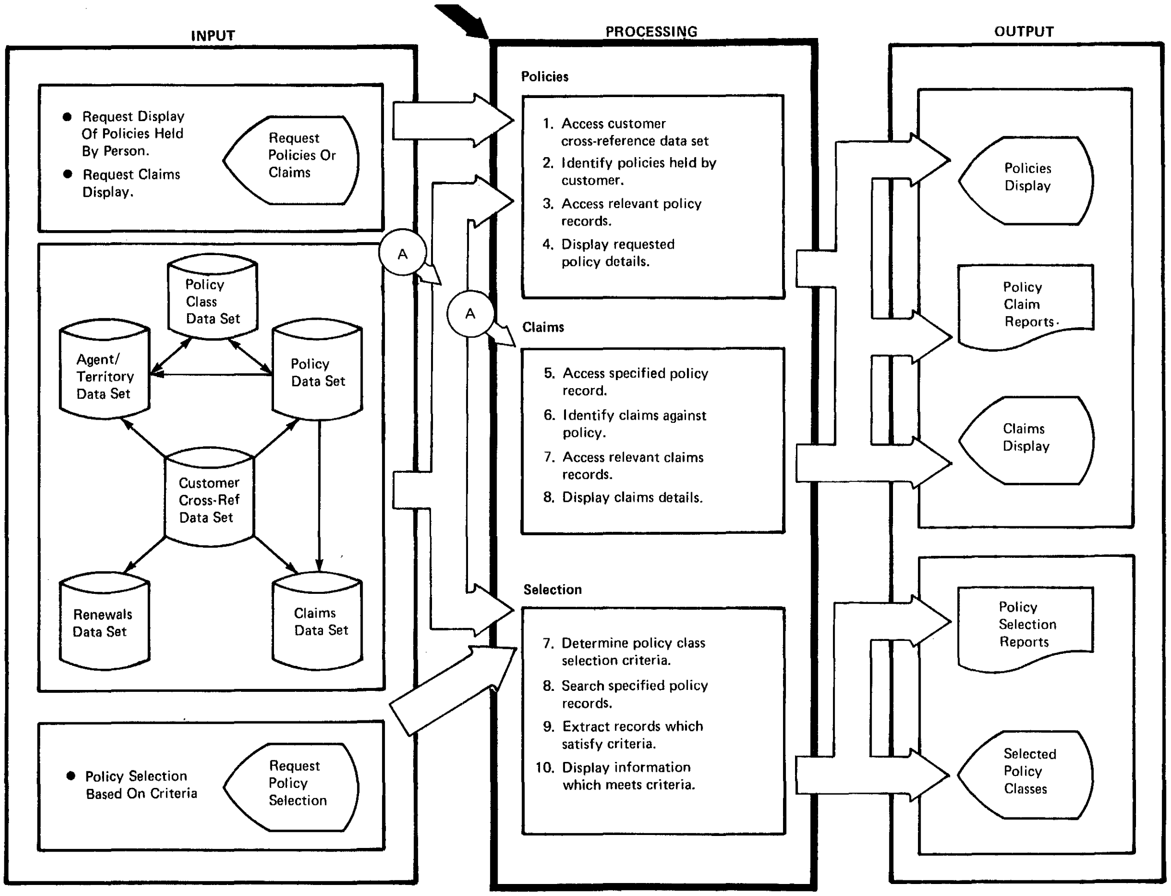

INPUT
PROCESSING
OUTPUT
● Request Display Of Policies Held By Person.
● Request Claims Display.
Request Policies Or Claims
Policy Class Data Set
Agent/ Territory Data Set
Policy Data Set
Customer Cross-Ref Data Set
Renewals Data Set
Claims Data Set
● Policy Selection Based On Criteria
Request Policy Selection
A
A
Policies
1. Access customer cross-reference data set
2. Identify policies held by customer.
3. Access relevant policy records.
4. Display requested policy details.
Claims
5. Access specified policy record.
6. Identify claims against policy.
7. Access relevant claims records.
8. Display claims details.
Selection
7. Determine policy class selection criteria.
8. Search specified policy records.
9. Extract records which satisfy criteria.
10. Display information which meets criteria.
Policies Display
Policy Claim Reports
Claims Display
Policy Selection Reports
Selected Policy Classes

Figure 11-14. Insurance Pclicy Infcrmation System

DATA SETS

The policy data set contains all the informaticn relating to each current insurance policy. A typical record format for this data set is illustrated in Figure 11-15.

Policy Number	Cus-tomer Code	Agent Number	Line of Business (Policy Class)	Issue Date	Cease Date	Premium Data	Activity		
							Claim	Claim	Renewal

Figure 11-15. Pclicy Record Format

The details of the policy are ccntained in a fixed section of the record, which may be regarded as a root segment. Following these policy details, there may be multiple claims which have teen made against that policy. These may te regarded as dependent segments of the policy root segment. Depending upon the type of policy, there may be also a multiple number of renewal segments, which represent the periodic payment cf premiums -- for example, monthly premium payments. By accessing a particular pclicy record, all the details relating to that policy, including claims and premium payments, are available for analysis.

Alternatively, claims and renewals may be supported as separate data sets, with the root segment of an account containing disk pointers to the most recent claim or renewal for that same account in the relevant data set. Each of the claims or renewals for an account is then chained backward through the data set.

The customer cross-reference data set contains each customer name and number, together with a multiple number of segments of the address, identification of agents who assist, and identification of the policies serviced by each agent. The code used to identify each customer may be a numeric code, a name code, or a phonetic code that is developed using the built-in phonetic conversion function of CICS/VS (see "Record Identification" in Chapter 5). Figure 11-16 shows a typical record format for the customer cross-reference data set.

Customer Code	Name	Address			Agent Number	Policies			Agent Number	Policies		
		Line 1	Line 2	Line 3		Pol. No.	Pol. No.	Pol. No.		Pol. No.	Pol. No.	Pol. No.

Figure 11-16. Customer Cross-Reference Record Format

The agent data set contains fixed information relating to the insurance agent, such as his name, employee number, and sales performance, followed by a multiple number of segments, one for each customer serviced by that agent. A typical insurance agent record format is illustrated in Figure 11-17.

Agent Number	Agent Name	Sales		Customer Numbers							
		Current	Year To Date	No.	No.	No.	No.	No.	No.	No.	No.

Figure 11-17. Insurance Agent Record Format

The territory and policy class data sets contain fixed information relating to the particular geographic territory or class of policy and may have a multiple number of segments identifying the customers in that territory or the policy numbers in that classification. By accessing this data set for a particular territory, all the customers in that territory can be identified. Further information about the policies owned by those customers is available from the customer cross-reference data set. Similarly, all the policy numbers in a particular policy classification can be identified and the information relating to those policies can be obtained from the policy data set.

ONLINE PROGRAMS

Having identified the various data sets used in a typical policy information system, the application programs should be defined. Some of the more common programs are described below.

Policy Display Program - Display Policy Information

This program displays policy details from the policy data set and formats these details in a page for display at the terminal. These pages are presented to the CICS/VS terminal paging routine (see Chapter 3) for subsequent display when requested by the terminal operator, as described below. In this way, all the policy details are made available for inquiry.

Customer Program - Display Customer Information

This program accesses the customer cross-reference data set, usually using customer name, and formats all the details relating to that customer (such as name and address) into a page for display at the terminal. Each policy owned by that customer is then retrieved from the policy data set by the system automatically initiating a policy inquiry, and the policy details are formatted into a page for display at the terminal. The terminal paging facility of CICS/VS (see Chapter 3) relieves the programmer of having to select and display each individual page requested by the terminal operator. Instead, CICS/VS terminal paging selects the appropriate page requested by the terminal operator from those pages supplied by the program and displays it on the terminal.

Claims/Renewals Program - Display Claims and Renewals

Based on specific policy identification, the details of the policy may be retrieved and the various claims made against that policy may be displayed, together with the renewal status of the policy.

Representative Inquiry - Display Representative Information

This program accesses the representative data set and formats representative details, such as name and sales performance, for display. Each client serviced by that representative can then be the subject of a policyholder inquiry if required.

Territory Inquiry - Display Policyholders in a Specific Territory

The territory data set is accessed, and territory details are displayed. The terminal operator should be permitted to display details relating to each, or alternatively a specific, policyholder in the territory.

Policy Class Inquiry - Display Policies of Particular Classification

This program is similar to the territory inquiry, except that the data set is accessed for a specific policy classification, and all the policies in that classification are displayed. A terminal operator then has the option to examine each of these policies in more detail with a policy inquiry.

NEW-BUSINESS POLICY ENTRY SYSTEM

This online application allows the entry of new-business policies into a policy data set of the insurance data base. Without such a system, the preparation of new-business policies would require highly experienced personnel to code the necessary information.

The use of online data entry under CICS/VS for this application enables the terminal operator to enter policy information in terms he is familiar with, and allows the computer to carry out much of the necessary coding and checking. In this way, it may be possible to use less experienced (and lower paid) personnel to prepare new business policies, with the computer handling the coding and checking of information for accuracy.

The 3790 Communication System enables much of this coding and checking to be done in a remote branch office either online, or offline for subsequent transmission to CICS/VS. The 3791 Controller contains 27.5 million bytes of disk storage which may be used to store sections of the data sets required for editing the 3790. Refer to "3790 Sessions" in this manual, and _Introduction to the IBM 3790 Communication System_, GA27-2767, for additional information about the 3790. A typical new-business policy entry system is illustrated in Figure 11-18.

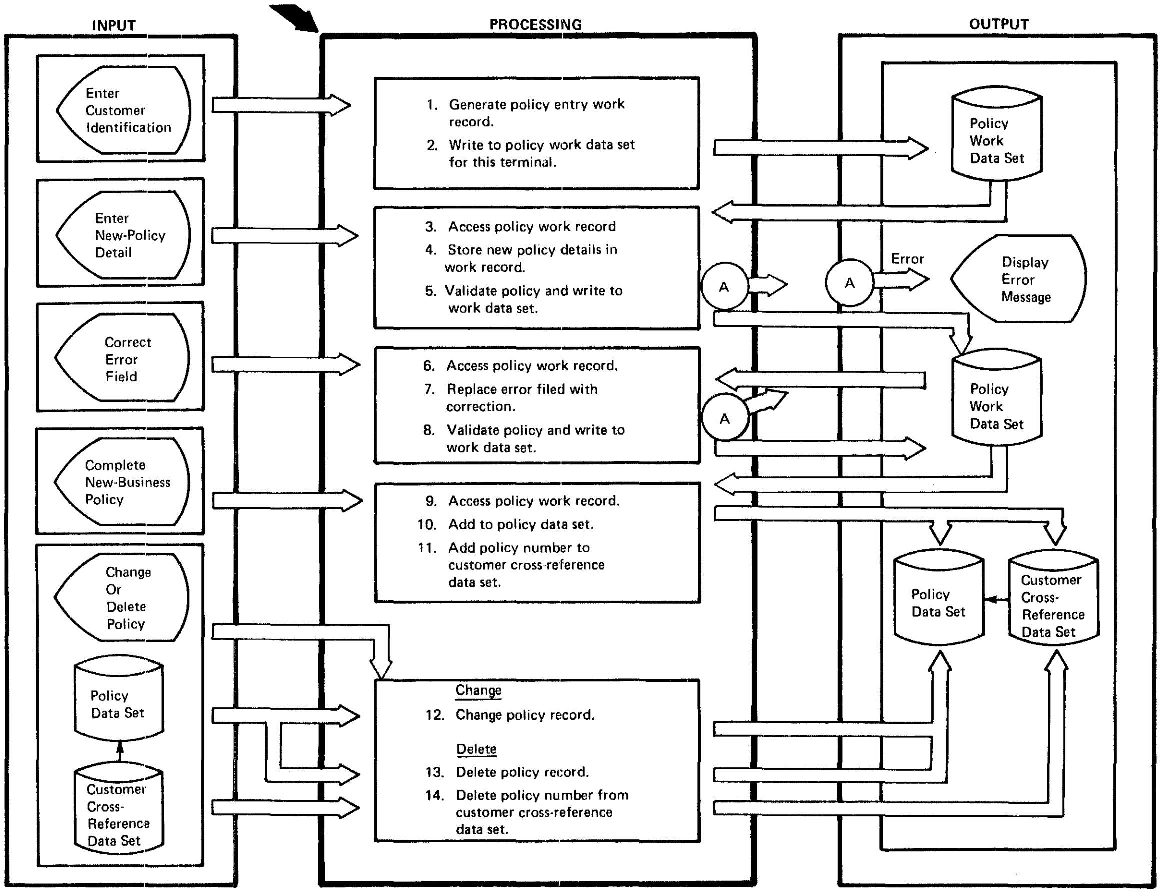

INPUT
PROCESSING
OUTPUT
Enter Customer Identification
Enter New-Policy Detail
Correct Error Field
Complete New-Business Policy
Change Or Delete Policy
Policy Data Set
Customer Cross-Reference Data Set
1. Generate policy entry work record.
2. Write to policy work data set for this terminal.
3. Access policy work record
4. Store new policy details in work record.
5. Validate policy and write to work data set.
6. Access policy work record.
7. Replace error filed with correction.
8. Validate policy and write to work data set.
9. Access policy work record.
10. Add to policy data set.
11. Add policy number to customer cross-reference data set.
Change
12. Change policy record.
Delete
13. Delete policy record.
14. Delete policy number from customer cross-reference data set.
A
A
A
Error
Policy Work Data Set
Display Error Message
Policy Work Data Set
Policy Data Set
Customer Cross-Reference Data Set

Figure 11-18. New-Business Policy Entry System

A considerable amount of information must be entered for each new business policy. Typically, the information entered from a terminal may approach, or in some cases exceed, 1,000 characters. If this approximately 1,000-character record is entered as one input transaction, the extent to which the computer can help in coding and checking is somewhat limited.

However, if a conversational approach is taken to the entry of information, the computer can guide the terminal operator by requesting each section of policy information as it needs it. By adopting this conversational approach, the computer can request information in much the same way as it may be requested on a new-business policy form, and then take that entered information and code it based upon defined rules for each section of the policy entry.

The advantage a conversational policy entry approach offers is that it now becomes possible for inexperienced operators to transcribe information directly from new business policy forms, with the computer carrying out the necessary coding and checking of information. Furthermore, in the event of an error being detected in the input, only that error field need be reentered.

To achieve this conversational policy entry, the policy information received from the terminal is used to progressively build up a complete new-business policy record. The policy work data set is used for this purpose. When the entry of the new-business policy is started, the policy information is written to the policy work data set. As subsequent policy information is received from the terminal for this new policy, the record written to the policy work data set for that policy is retrieved, updated with the additional information, and written back. In this way, the new business policy record is gradually built up with each section of policy information being fully edited, validated, coded, and written to the data set before the next section is obtained from the terminal operator.

On completion of entry of the policy information, the new-business policy record is transferred to the policy data set, and the policy number is added to the customer cross-reference data set for the customer who has taken out the new policy.

DATA SETS

The policy work data set contains a record for each new-business policy entry terminal. In the event of system failure, the policy work data set records all policy information entered by each terminal operator up to the time when a failure may have occurred. On system restart, it enables previously entered policy information to be retrieved from the data set such that the terminal operator can continue with the policy entry, close to the point he had reached when the system failure occurred.

This policy entry recovery facility is made possible through the use of conversational entry of policy information. If all the policy information was entered as one large input transaction of approximately 1,000 characters, and the system went down toward the end of entering policy information, it would be necessary to rekey all that policy information again when the system is restarted. However, using conversational entry, only the last conversational section of the policy may have to be rekeyed on restart.

The customer cross-reference data set and the policy data sets are the same as described above for the policy information system.

ONLINE PROGRAMS

Policy Entry Program - Enter Policy Details

This program accepts policy information from terminals, validates that information based on established editing and checking procedures used by the insurance company, codes the validated information, and writes it to the policy work data set. The response returned to the terminal indicates the information to be next entered for the policy. Each subsequent transaction received from the terminal is also validated and coded, then added to the information for that policy in the policy work data set.

While this program has been identified as a policy entry program, it is in fact a series of program modules, each module used to validate and code each separate terminal transaction which is used conversationally to build up the entire policy. Each program may utilize temporary transaction codes (refer to "Task Initiation" in Chapter 3) to identify the program to process the next section of the policy, without requiring the terminal operator to enter a transaction code.

Correction Program - Correct Error Fields

This program accepts a terminal transaction which is identified as the correction of a previously entered error field. It retrieves the previous error information that was logged to the policy work data set, inserts the relevant corrected field, and revalidates the policy information. If the corrected field passes the validation procedure, it is inserted into the new-business policy record and written back to the policy work data set.

Change Program - Change Existing Policies

This program accesses policy records either in the policy work data set, or in the policy data set, to change specified policy information.

Delete Program - Delete Policies

This program accesses the policy data set and customer cross-reference data set to delete specified policies.

Claims and Renewals Entry Program

An extention to this new-business policy entry system enables the entry, validation, and addition of claims and renewal information to the policy data set. This uses a similar design approach to that described above, except that the amount of information to be entered may be supplied in a single input transaction, rather than in a conversational transaction environment as for new-business policies. A temporary data set such as the policy work data set is not necessary. Instead, after editing and coding, the claims or renewal information may be added directly to the policy data set.

DATA BASE SUPPORT SELECTION

The criteria that should be used in selection of the data base support are described in "Data Base Selection Criteria" in Chapter 5.

The policy data set, agent data set, policy class data set, territory data set, and the customer crcss-reference data set contain fixed infcrmation, followed by a variable amcunt of related information, such as:

- Claims and renewals fcr a particular policy
- Policy numbers for a particular customer
- Custcmers for a particular insurance agent
- Customers for a particular territory
- Policy numbers for a particular policy classification

DL/I Support

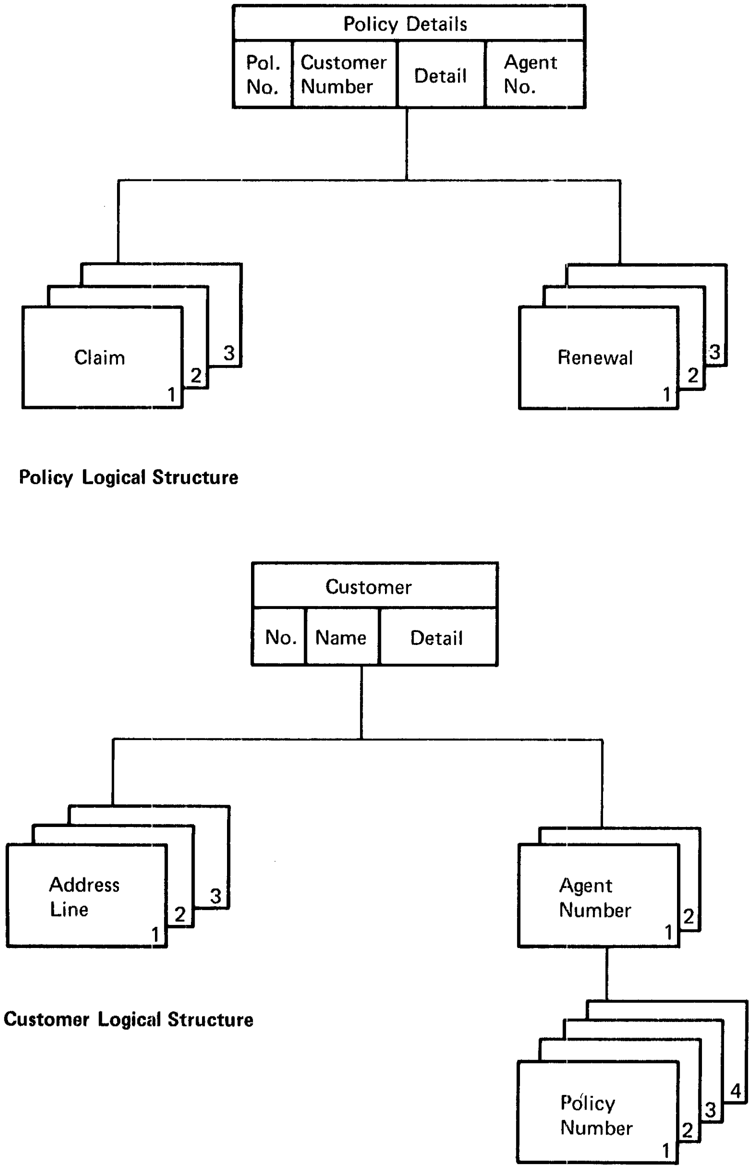
Because of the above considerations, the ideal data base support in this industry is DL/I. The multiple occurrence cf related informaticn described above can be supported as dependent segments, with the fixed information in the various data sets as root segments (see Figure 11-19). As these segments can be effectively made fixed-length, any of the DL/I products may be used.

DL/I ENTRY, DL/I DOS/VS, and IMS/VS DL/I suppcrt logical relationships and secondary indexing, which enable indexes tc be created and used by DL/I such that dependent segments may be directly accessed. An example of the use of such secondary indexing is the creation of DL/I indexes for directly accessing claims based on a claim number.

CICS/VS File Control

DL/I has many advantages, particularly in the area of reduced maintenance, over CICS/VS file ccntrol. However, if DL/I will not be used for varicus reasons, the segmented record feature of CICS/VS file control may be used to advantage. As previously discussed for the manufacturing and banking industries, the dependent segments may be defined as separate file ccntrol segments.

The policy information system is an inquiry applicaticn, and does not involve changes cr additions to the data base. However, the new-business policy entry system will result in additions to the policy data base and the customer crcss-reference data base. In this case, the length of file ccntrcl segmented records will be increased, or in the case of deletion of policies, will be decreased. Consequently, the most suitable access method tc provide this capability is key-sequenced VSAM, provided the user's system has at least 144K of real storage for satisfactory perfcrmance. If ISAM or DAM is used, the overflow technique described in "Segment Updating (DAM, ISAM, and Entry-Sequenced VSAM)" in Chapter 5 may be utilized to increase the length of segmented DAM or ISAM records. However, this technique requires additional prcgramming.



Policy Logical Structure

Customer Logical Structure

Figure 11-19. Pclicy Infcrmation System Logical Structures

MEDICAL INDUSTRY

In the medical industry, a typical online application is the control and maintenance of informaticn relating to all of the patients in a hospital or clinic, in a patient information system.

PATIENT INFORMATION SYSTEM

This application enables all activity such as visits, diagnoses, and treatments, between a patient and a hospital or clinic, to be recorded as that patient's history. When a patient is first admitted to a hospital or a clinic, his personal details can be entered from an online terminal directly intc the patient information system. On each subsequent visit, information can be added to his history -- all visits made by the patient, the diagnosis made on each visit, and the medical or surgical treatment received; a complete history can be developed. An example of such a patient infcrmaticn system is illustrated in Figure 11-20.

CICS/CS/VS supports the 3850 Mass Storage System to permit massive patient information system data bases to be maintained online. The 3850 provides online capacity to large data bases for applications with low transaction volumes which can tclerate lcng response times. (See "CICS/OS/VS - 3850 Mass Storage System Operation" in Chapter 7.) A typical 3850 application in this industry is the online residence of medical literature and research data. Use may be made of the STAIRS/VS Program Froduct (Program No. 5740-XR1) to retrieve relevant medical literature based upcn a keywcrd criteria search. See the STAIRS/VS General Information Manual (GH12-5164) for additional information.

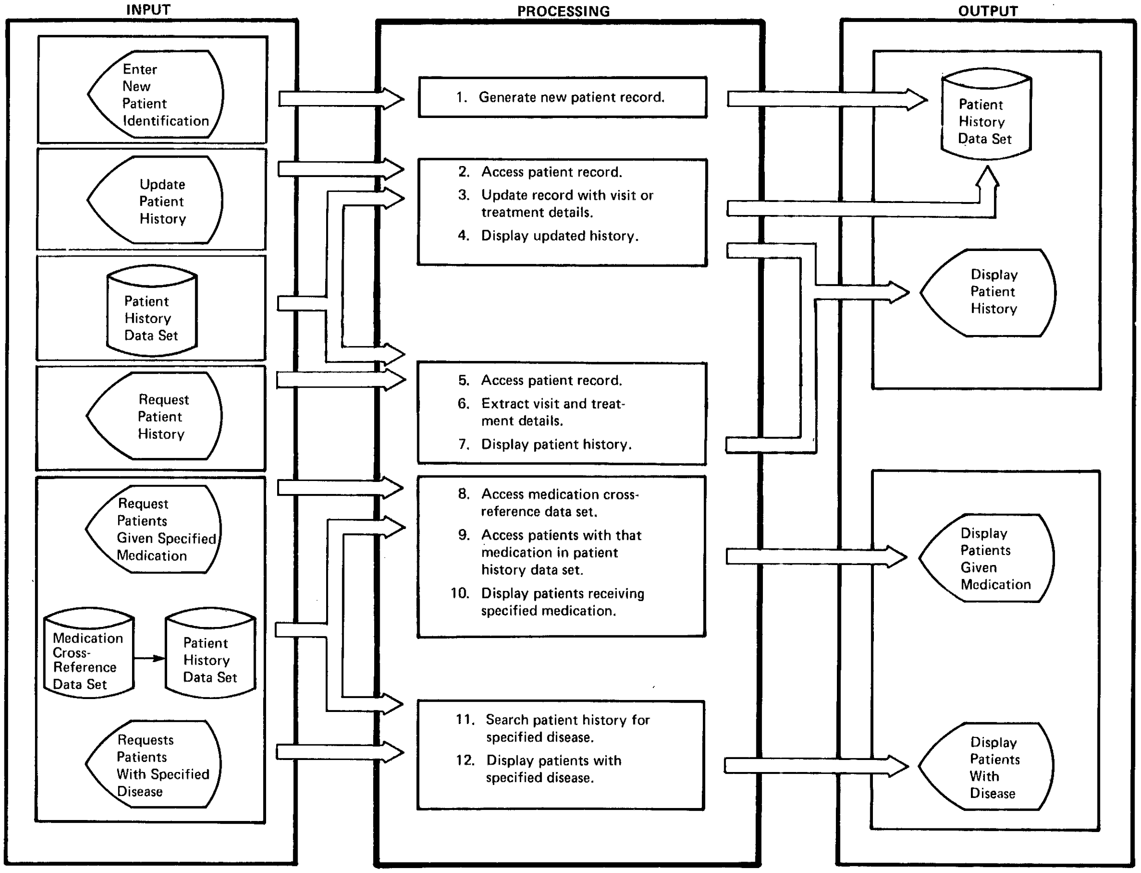

INPUT
PROCESSING
OUTPUT
Enter New Patient Identification
Update Patient History
Patient History Data Set
Request Patient History
Request Patients Given Specified Medication
Medication Cross-Reference Data Set
Patient History Data Set
Requests Patients With Specified Disease
1. Generate new patient record.
2. Access patient record.
3. Update record with visit or treatment details.
4. Display updated history.
5. Access patient record.
6. Extract visit and treatment details.
7. Display patient history.
8. Access medication cross-reference data set.
9. Access patients with that medication in patient history data set.
10. Display patients receiving specified medication.
11. Search patient history for specified disease.
12. Display patients with specified disease.
Patient History Data Set
Display Patient History
Display Patients Given Medication
Display Patients With Disease

Figure 11-20. Patient Infcrmaticn System

The patient history develcped over a period of time using the patient information system can be made available by means cf the computer to any authcrized person. Ccnsequently:

- Doctors may be made aware of all visits, diagnoses, and treatments relating to a patient.

- All handling and treatment of a patient can be identified.

- A record of all patient activity with the hospital or clinic can be made available to the accounting department for billing purposes.

In addition to recording information, relating to a particular patient, the computer can also record elsewhere the fact that the patient has received particular medication. It is then possible to determine which patients have received certain medication within a certain time period, for example.

DATA SETS

The patient history data set contains information identifying the patient, such as patient number, name, address, and personal details. Associated with this patient may be a multiple number of segments of information, each segment relating to a particular visit or treatment. A typical patient history record is illustrated in Figure 11-21.

Patient		Patient	Doctor	Address			Patient History				
No.	Name	Details	Number	Line 1	Line 2	Line 3	Visit	Diagnosis	Treatment	Treatment	Visit

Figure 11-21. Patient History Record Format

The medication cross-reference data set contains details relating to each particular medication, with a multiple number of segments identifying the patients who received that medication and the dates administered. Using this data set, all the patients who received particular medication at a particular time can be identified. By accessing the patient history data set, further information can be obtained relating to various patients. A typical medication cross-reference record format is shown in Figure 11-22.

Medication Identification			Patients Administered								
Number	Description	Details	Patient No.	Date	Dose	Patient No.	Date	Dose	Patient No.	Date	Dose

Figure 11-22. Medication Cross-Reference Record Format

A diagnosis data set and a doctor data set, if required, may be used to identify all patients who contacted a particular disease, or who were or are treated by particular doctors. The format of these data sets is almost identical to that of the medication cross-reference data set.

ONLINE PROGRAMS

Add Patient Program - Add New Patient to Patient History

This program accepts information relating to a new patient and adds that information to the patient history data set.

Patient Update Program - Update Patient History with Visits, Treatments, and Diagnoses

This program is used to add information relating to a particular patient visit, or treatment received by the patient, to the patient history data set. Over a period of time, a complete historical record is built up for each patient.

Patient History Inquiry Program - Display Patient History Information

This inquiry program accesses the patient history data set and formats that history into pages to be displayed on request by the terminal operator. These pages are presented by the program to the CICS/VS terminal paging routine (see Chapter 3), which saves them and displays them for the terminal operator in the sequence requested.

Medication Inquiry Program - Display All Patients Receiving Certain Medication

This program accesses the medication data set and identifies each patient who received specified medication over a specified time period. Further information relating to those patients may be obtained by a patient history inquiry if required.

Diagnosis Inquiry Program - Display All Patients with a Specified Disease

This program accesses the diagnosis data set and identifies all patients who contracted a specified disease. Further information can be obtained about these patients by means of a patient history inquiry.

Doctor Inquiry Program - Display All Patients Relating to a Specified Doctor

This program accesses the doctor data set and identifies all patients treated by that doctor. Further information about these patients may then be obtained by a patient history inquiry.

DATA BASE SUPPORT SELECTION

At this stage of the application design, the data base support to be used can be determined. Refer to "Data Base Selection Criteria" in Chapter 5 for a discussion of the factors which should be considered in this selection.

DL/I Products

The record formats of the patient history, medication, diagnosis, and doctor data sets (see Figures 11-21 and 11-22) exhibit multiple occurrences of information, dependent upon a root segment. These record formats indicate that DL/I is the ideal data base support for this application.

Generally, the segments may be made fixed-length, which allows DL/I ENTRY to be used. Variable-length segments may be used by DL/I DOS/VS and IMS/VS DL/I for information such as names and addresses.

Figure 11-23 shows a typical DL/I logical structure for the patient history data base.

Information describing patient's name, address, and physical characteristics is contained in the patient detail root segment. Each patient can have many visit and dependent diagnosis segments.

The support of logical relaticnships and secondary indexing by DL/I ENTRY, DL/I DOS/VS, and IMS/VS is useful in the design of the patient information system data tase.

Refer to "Sample Applications" in *DL/I DOS/VS General Information Manual* for further discussicn on the use of DL/I in the medical industry.

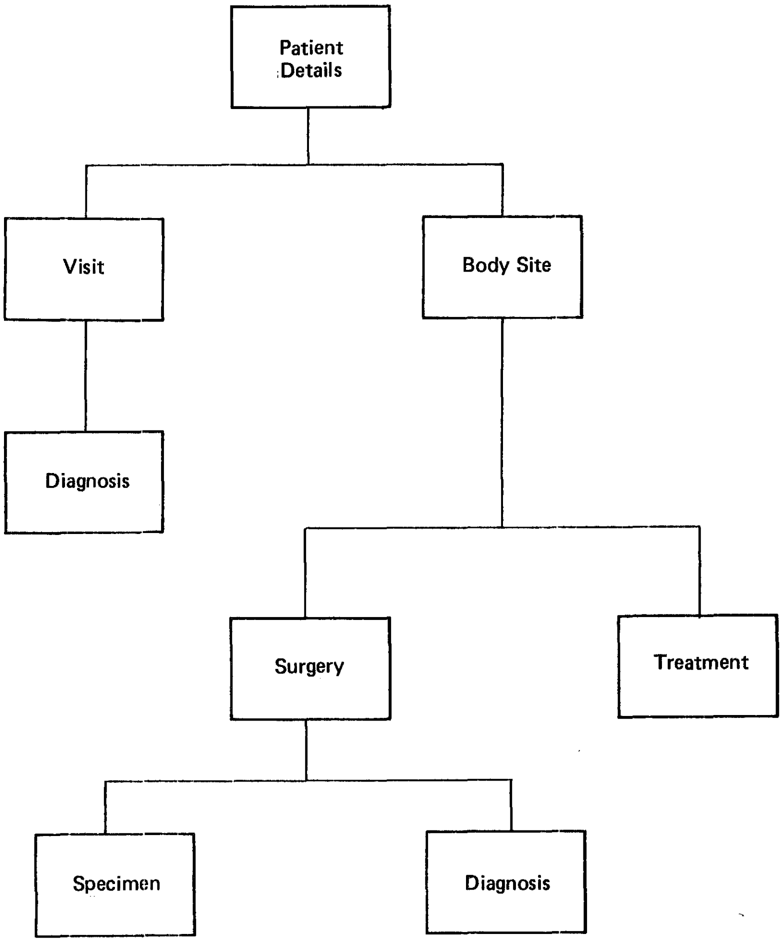


Figure 11-23. Patient History Lcgical Structure

CICS/VS File Control

The segmented reccrd feature cf CICS/VS file ccntrol may be used to support the multiple occurrerce cf dependent segments. Because of the requirement for increasing the length cf segments as information is added to a record, the most suitable access methcd to be used is key-sequenced VSAM.

For machines with less than 144K bytes of real storage, the use of VSAM is not advised, for performance reasons. ISAM and DAM may be used instead as the access method support. In this case, the overflow technique described in the section "Segment Updating (DAM, ISAM, and Entry-Sequenced VSAM)" in Chapter 5 may be used.

PHARMACEUTICAL INDUSTRY

A typical online application in the pharmaceutical industry is the entry of orders from pharmacists for various products, and the filling of those orders in the pharmaceutical company's warehouse. This application is generally referred to as a pharmaceutical order entry system.

PHARMACEUTICAL ORDER ENTRY SYSTEM

In this industry, orders for products are generally made by product name rather than by product number. Products such as aspirin may be supplied by many manufacturers in different strengths and in various quantities. Thus, an order taker must have a very good knowledge of the company's range of products so that he can readily identify the particular product ordered. Figure 11-24 illustrates a typical pharmaceutical order entry system. In this aplication, generally,

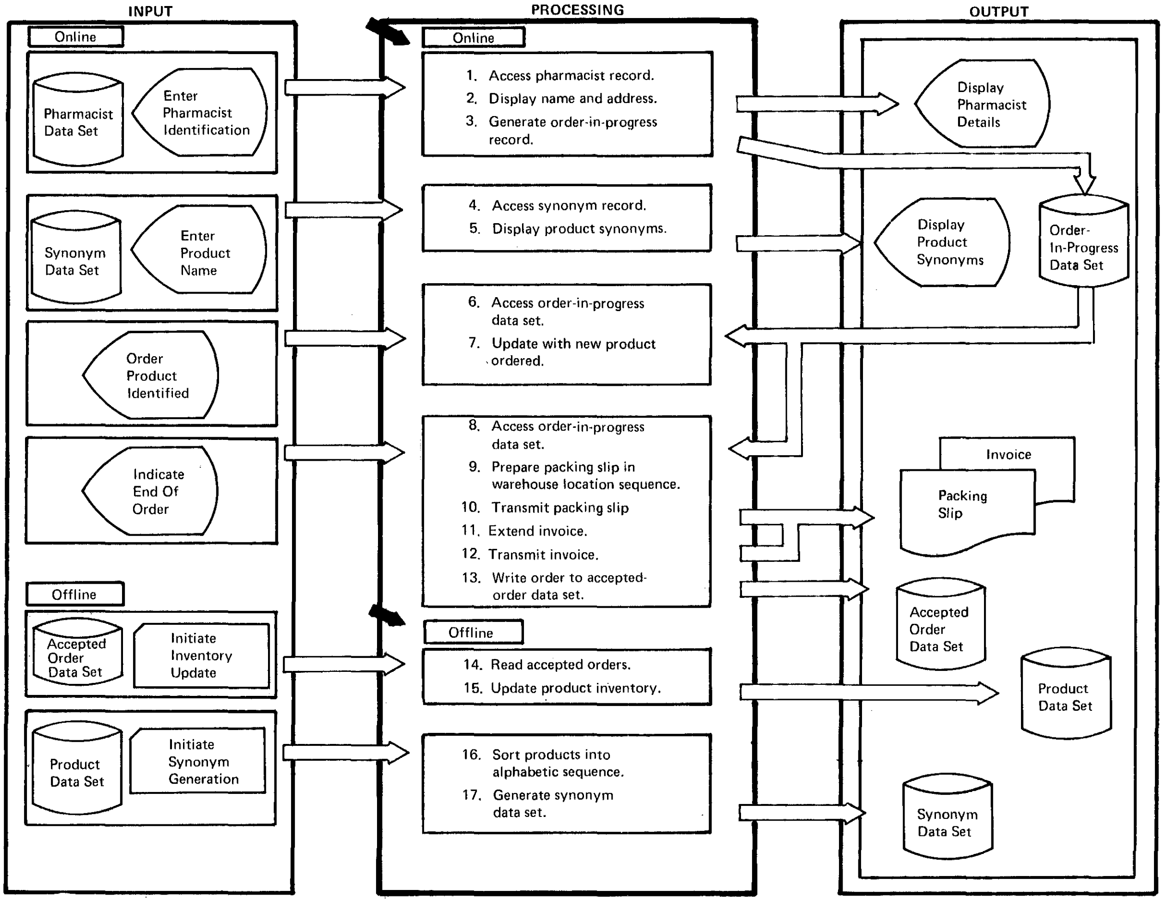

INPUT
Online
Pharmacist Data Set
Enter Pharmacist Identification
Synonym Data Set
Enter Product Name
Order Product Identified
Indicate End Of Order
Offline
Accepted Order Data Set
Initiate Inventory Update
Product Data Set
Initiate Synonym Generation
PROCESSING
Online
1. Access pharmacist record.
2. Display name and address.
3. Generate order-in-progress record.
4. Access synonym record.
5. Display product synonyms.
6. Access order-in-progress data set.
7. Update with new product ordered.
8. Access order-in-progress data set.
9. Prepare packing slip in warehouse location sequence.
10. Transmit packing slip
11. Extend invoice.
12. Transmit invoice.
13. Write order to accepted-order data set.
Offline
14. Read accepted orders.
15. Update product inventory.
16. Sort products into alphabetic sequence.
17. Generate synonym data set.
OUTPUT
Display Pharmacist Details
Display Product Synonyms
Order-In-Progress Data Set
Invoice
Packing Slip
Accepted Order Data Set
Product Data Set
Synonym Data Set

Figure 11-24. Pharmaceutical Order Entry System

orders are placed by pharmacists over the telephone, with the telephone operator keying into the terminal each product ordered. The name of the product ordered is entered on the terminal, and the computer accesses all products similar to that entered (that is, synonyms) and displays each of the syncnyms at the terminal for selection by the operator.

In this industry, a pharmacist's order is normally accepted regardless of whether or not there is sufficient stock on hand to fill that order. An cnline inventcry status check is normally not carried out. However, it is important that packing slips for such orders be passed tc the warehouse as quickly as possible. Ideally, the computer should prepare all the products in each packing slip in warehouse location sequence for faster picking of products. This packing slip, together with an extended invoice if preinvcicing is to be implemented, can be transmitted to a terminal in a warehouse very shortly after completicn of entry of the order.

DATA SETS

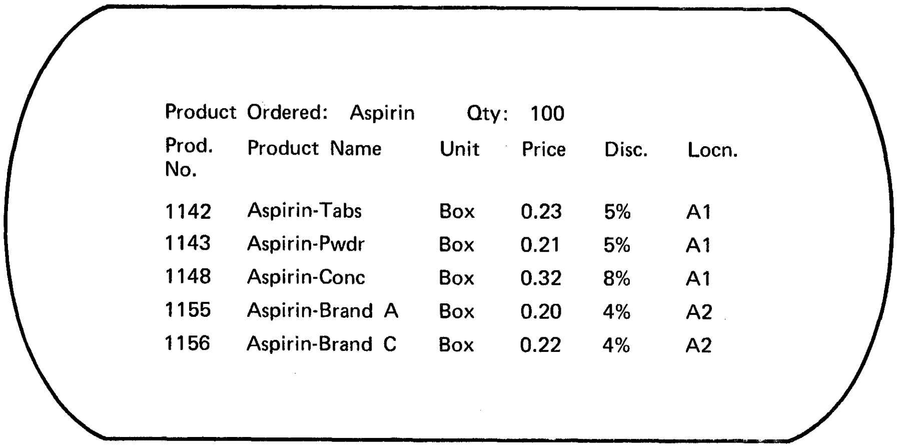
The synonym product data set is a special online version of the product data set (which is used only in the batch environment). If the product data set is used online to identify various product synonyms, a file access may be needed for each separate synonym. Because there may be twenty or mcre synonyms fcr each product crdered from each terminal, accessing the prcduct data set online represents a significant (and unnecessary) file accessing lcad. This online accessing is unnecessary, as the cnly purpose is to identify synonyms and not to update inventory levels. In this industry, inventory is normally updated overnight by batch prcgrams.

The synonym product data set is created offline from the product data set (see Figure 11-24), by extracting frcm each product record the product number and name into a separate data set. (A typical product data set record format and product data base DL/I logical structure are shcwn in Figures 11-31 and 11-32, respectively, for the distribution industry. These figures are also relevant to the offline pharmaceutical product data set.) Informaticn needed for invoice calculations, such as unit price, unit size, discounts, and warehouse location, is also extracted from the product data set to make up a synonym record. Syncnym reccrds are scrted into alphabetical sequence on the product name and are processed sequentially in a batch environment to generate display images of synonyms based on the first four to seven characters of the name (depending upcn installation requirements). The display images cf synonyms are used to generate the synonym product data set with the first characters of the product name (that is, the syncnym name) as the key. Figure 11-25 shows a typical synonym product record format and the resultant synonym display.

Using the product name entered frcm the terminal, the first four to seven characters of the name are used to access the synonym product data set and to retrieve one or several synonym displays, each containing many synonym prcducts. The operator then identifies the appropriate product. That product identification can be entered with the next product to be crdered, together with details of that next product.

By using the synonym prcduct data set, the amcunt of data set accessing which is necessary is dramatically reduced, with a subsequent potential improvement in online performance and response time.

Synonym	Product Line 1						Product Line n					
Key	No.	Descr.	Price	Disc.	Locn.	Unit	No.	Descr.	Price	Disc.	Locn.	Unit



Product Ordered: Aspirin Qty: 100

Prod. No.	Product Name	Unit	Price	Disc.	Locn.
1142	Aspirin-Tabs	Box	0.23	5%	A1
1143	Aspirin-Pwdr	Box	0.21	5%	A1
1148	Aspirin-Conc	Box	0.32	8%	A1
1155	Aspirin-Brand A	Box	0.20	4%	A2
1156	Aspirin-Brand C	Box	0.22	4%	A2

Figure 11-25. Synonym Reccrd Format and Display

The pharmacist data set ccntains information relating to the pharmacist, as shown in Figure 11-26. This data set is accessed at the start of each order to identify the pharmacist for credit purposes, account history, or disccunt.

Pharmacist		Credit	Postal Address			Ship-To Address			Account
No.	Name	Limit	Line 1	Line 2	Line 3	Line 1	Line 2	Line 3	History

Figure 11-26. Pharmacist Data Set

The order-in-progress data set is used as a wcrk data set. Each product crdered by name, together with its quantity, is entered conversationally and the apprcpriate product name is selected from the various synonyms. This crdered product is then written tc the order-in-progress data set (see Figure 11-27). As further products are ordered, the order-in-prcgress reccrd is updated until the ccmplete order has been received. This is necessary in the event that the

Order	Cust.	Product 1						Product n					
No.	No.	No.	Descr.	Unit	Price	Disc.	Locn.	No.	Descr.	Unit	Price	Disc.	Locn.

Figure 11-27. Order In-Prcgress Reccrd Format

pharmacist may wish to change part of his order, or cancel particular products or the entire order before he has completed the order.

Once the order has been completed, it is then transferred from the order-in-progress data set to the accepted order data set. This is effectively a direct access file which will be processed overnight as a sequential data set to update the product data set inventory. It has the same record format as the order-in-progress data set (see Figure 11-27).

ONLINE PROGRAMS

Order Start Program - Accept Pharmacist's Identification

This program accesses the pharmacist data set, displays the pharmacist's name, address, and credit information, if required for confirmation, and writes an order-in-progress record.

Order Entry Program - Accept Product Orders

The product name and quantity to be ordered are accepted from the terminal and used to access the synonym data set. The highest synonym key of each cylinder may be held in a table in storage, if required for faster access, to allow the synonym displays for a product to be retrieved directly. Details relating to the product, such as unit price, discounts, and warehouse location, are part of the synonym record (see Figure 11-25) and may be displayed for the operator, if desired.

Order Finish Program - Indicate Completion of Order

This program indicates that the entire order has been completed, at which time the order-in-progress record is transferred to the accepted order data set, to be used offline overnight to update the product inventory in the product data set. This program then initiates the warehouse location sequencing program.

Warehouse Location Sequencing Program - Sequence Products into Warehouse Location

This program is initiated automatically by the system at the completion of an order. It sequences the orders into warehouse location, based upon the product's location extracted from the synonym data set (see above). On completion of the warehouse location sequencing, control is passed to the invoice calculation program.

Invoice Calculation Program - Extend Invoice

This program takes the order when it has been sequenced into warehouse location. Using the unit price, discount, and size supplied by the synonym record (see Figure 11-25), it extends each line item and then calculates the total invoice.

Warehouse Transmission Program - Send Packing Slip and Invoice

The order, sequenced into warehouse location, is formatted into the warehouse packing slip and transmitted to a printer in the warehouse. Following this, the extended invoice is formatted and transmitted to the same printer so that the packing slip and invoice may be packed with the ordered products for preinvoicing.

CFFLINE PROGRAMS

Synonym Data Set Creation

As described earlier, this prcgram accepts the product data set sorted into alphabetical sequence and then formats display images of synonym products based on the first four to seven characters of the product name, for retrieval cnline from the synonym product data set. Information such as the unit price, unit size, applicable discounts, and warehouse location is also extracted for inclusion in the synonym record, as shown in Figure 11-26.

At the same time, a synonym display image can be printed to be used by terminal operators in the event of system downtime, for manual backup and identificaticn of prcducts ordered.

Inventory Update Program

The accepted crder data set created online is read sequentially by this program and used to update the stock levels cn the product data set. Depending upcn the frequency of activity of various products on this product data set, the accepted order data set may be sorted into the same sequence as the product data set before execution of the update program, for better perfcrmance. As a result of the product update, various inventcry reports may be produced, and back orders may be recorded on a back-order data set.

DATA BASE SUPPORT SEIECTION

The selection of data base suppcrt for this application can now be considered. Refer to "Data Ease Selection Criteria" in Chapter 5 for a discussion of the varicus factcrs relevant tc this selection.

DL/I Products

In most cases with this applicaticn, the data sets contain simple record formats which may be equally well supported by either DL/I or CICS/VS file ccntrol. In this irstance, the decision on data base support will generally be dictated by future installation direction rather than cther considerations. While this particular application does not need all the facilities of DL/I, the principal advantage of data independence with DL/I, and hence reduced maintenance of the data base, is an important reason for using it. Refer to "DL/I Products" for the distributicn industry later in this chapter, for a discussion of the use of DL/I to support the product and custcmer data bases. The techniques described are alsc applicable to the pharmaceutical industry.

CICS/VS File Control

If the data base support decisicn is to use file contrcl, either DAM, ISAM, or VSAM may be selected. The order-in-progress data set would normally be supported as a DAM or entry-sequenced VSAM data set. The accepted orders data set may be supported as a sequential data set, or as a direct access data set created sequentially. The pharmacist data set would typically be ISAM or key-sequenced VSAM. The synonym data set may also be ISAM or key-sequenced VSAM. The product data set used in a batch environment may be either a direct access data set or a keyed data set such as ISAM or key-sequenced VSAM. However, the user should recognize that VSAM shculd not be utilized cn systems with less than 144K bytes of real stcrage, for performance reasons.

While the product data set does contain some descriptive information (such as the product name), this data set need not use variable-length records unless disk storage is at a premium. In the case of the pharmacist data set, however, the name, postal address, and possible separate ship-to address may indicate significant disk storage savings by defining the pharmacist data set as a variable-length segmented record data set. The pharmacist's name, each address line, and the ship-to address may be made variable-length segments, and, through the presence or absence of segments, a variable number of address lines may be supported if necessary. Refer to the discussion of segmented records in Chapter 5 for a typical segmented customer record. This segmented record is useful also for a pharmacist data set.

DISTRIBUTION INDUSTRY

A typical online application in the distribution industry is order entry and invoicing. This application is similar in many respects to the pharmaceutical order entry system described above, but differs in the area of stock status checking.

ORDER ENTRY AND INVOICING SYSTEM

In this industry, products are generally ordered by product number. Orders may be accepted over the telephone, in person, or by mail. A customer number or account number is used to identify the person making the order. This customer identification is used to access a customer data set to obtain information such as customer name and address, ship-to address, and current credit rating.

The significant differences between this application and the pharmaceutical order entry application are discussed in the following.

Since each item is ordered by product number and quantity, the computer is able to access the product data set directly to obtain the product name, unit price, relevant discounts, and quantity-on-hand. The quantity-on-hand may be updated immediately to reflect acceptance of the order. In the event of an insufficient quantity-on-hand, the terminal operator may elect to either:

- Accept the quantity available

- Cancel that order item

- Cancel the entire order

depending on the customer's requirements.

On completion of the order, the computer can be used to produce an extended invoice to be transmitted to the warehouse, together with a packing slip listing products in warehouse location sequence. Furthermore, it is possible to produce an invoice as confirmation of the acceptance of the order if required, by also transmitting a copy of the invoice to the terminal. This order confirmation may result in improved customer service, and is of particular interest for orders placed in person by the customer. In the case of the orders placed by telephone or mail, the confirmation invoice may be mailed to the customer if required, depending upon the delivery time of the actual products ordered.

The advantages which result from an online order entry system similar to the one described above are:

- Improved customer service
- Improved credit control
- Up-to-the-minute stock status availability
- Potential reduction in inventcry levels
- Removal of the need for stock clerks, pricing clerks, and checking clerks
- Preinvoicing, and efficient warehouse picking

Figure 11-28 shows a typical crder entry and invoicing system.

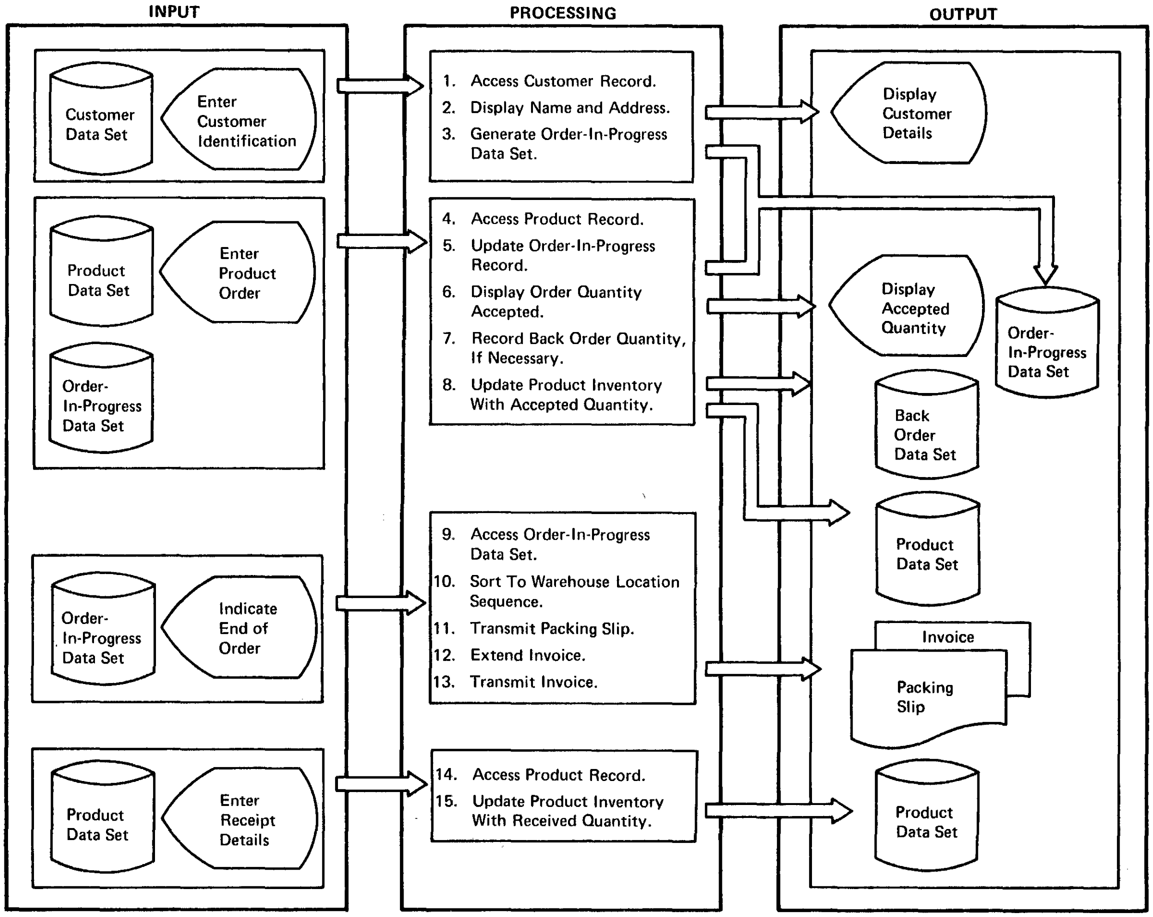

INPUT
PROCESSING
OUTPUT
Customer Data Set
Enter Customer Identification
Product Data Set
Enter Product Order
Order-In-Progress Data Set
Order-In-Progress Data Set
Indicate End of Order
Product Data Set
Enter Receipt Details
1. Access Customer Record.
2. Display Name and Address.
3. Generate Order-In-Progress Data Set.
4. Access Product Record.
5. Update Order-In-Progress Record.
6. Display Order Quantity Accepted.
7. Record Back Order Quantity, If Necessary.
8. Update Product Inventory With Accepted Quantity.
9. Access Order-In-Progress Data Set.
10. Sort To Warehouse Location Sequence.
11. Transmit Packing Slip.
12. Extend Invoice.
13. Transmit Invoice.
14. Access Product Record.
15. Update Product Inventory With Received Quantity.
Display Customer Details
Display Accepted Quantity
Order-In-Progress Data Set
Back Order Data Set
Product Data Set
Invoice
Packing Slip
Product Data Set

Figure 11-28 Distribution Order Entry and Invoicing System

DATA SETS

The data sets used in this application are analogous to those described for the pharmaceutical industry. The crder-in-progress data set is used to temporarily hcld the crder until completicn, in the event of a change in the order or possible cancelaticn of a particular item or the entire order. The ccmpleted order is then transferred to the accepted order data set. The record format for the accepted order data set and the crder-in-prcgress data set is shown in Figure 11-29.

Order	Cust.	Product 1						Product n					
No.	No.	No.	Descr.	Unit	Price	Disc.	Locn.	No.	Descr.	Unit	Price	Disc.	Locn.

Figure 11-29. Accepted Order and Order-in-Progress Record Formats

The customer data set contains similar information to that described for the pharmacist data set and is illustrated in Figure 11-30.

Customer		Credit	Postal Address			Ship-To Address			Account
No.	Name	Limit	Line 1	Line 2	Line 3	Line 1	Line 2	Line 3	History

Figure 11-30. Customer Record Format

In this application, products are generally identified by product number. The product data set can be placed online, allowing the stock status to be immediately updated on acceptance of each line item ordered. Figure 11-31 illustrates a typical product record format.

Item No.	Item Name	Unit Size	Price Per Unit	Dis-Count	Ware-House No.	Stock Loca-tion	No. of Items In Stock	Date of Last Change	Reorder Point	Supplier No.	No. Of Items On Order	No. Of Items Delivered	Date Of Order	

Figure 11-31. Product Record Format

The reorder data set is used to record orders to be placed on suppliers when the stock status of a product reaches its reorder point. It may be a separate data set, or part of the product data set (see "DL/I Products" below, and Figures 11-31 and 11-32).

The back order data set is used to record back orders placed because of insufficient stock. It may be a separate data set, or part of the product data set (see "DL/I Products" under "Data Base Support Selection", and Figures 11-31 and 11-32).

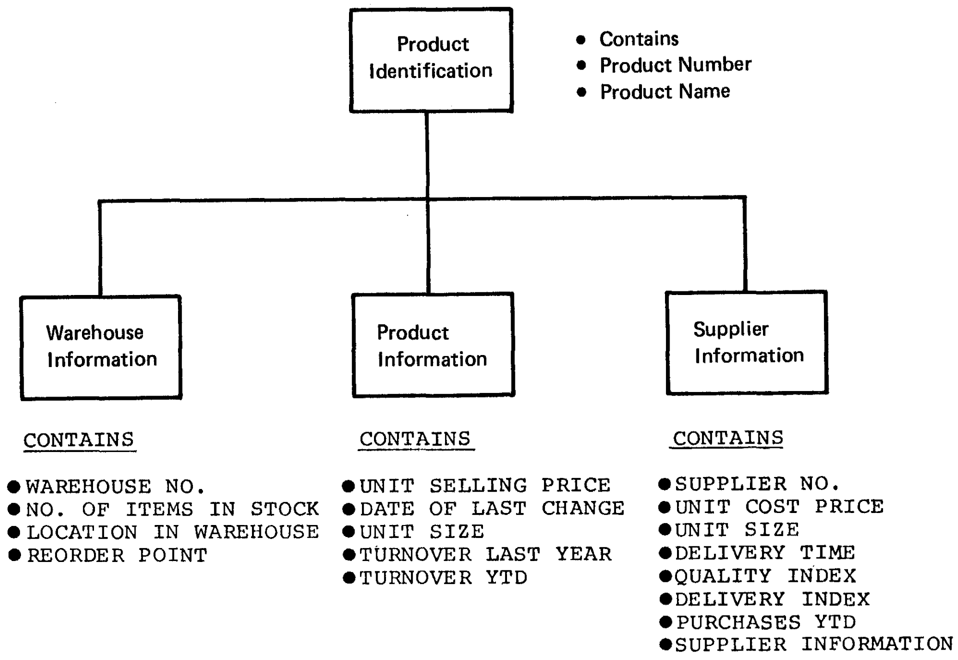


Figure 11-32. Product Data Base Logical Structure

ONLINE PROGRAMS

Order Start Program - Start New Customer Order

This program accesses the customer data set based on customer number and displays the customer name, address, ship-to address, and credit rating for confirmation. An order-in-progress record is then created for this terminal.

Order Entry Program - Accept Product Orders

This program accepts products ordered by product number and quantity, accesses the product data set, updates stock status, determines whether the reorder point has been reached, and creates a reorder record if necessary, then adds to the back order data set if the complete quantity ordered could not be satisfied. The line item ordered is then used to update the order-in-progress record for that terminal.

Order Finish Program - Signal Completion of Order

This program indicates the completion and acceptance of the order. The program transfers the order-in-progress record for this terminal to the accepted order data set. The warehouse location sequencing program is then automatically initiated by the system.

Warehouse Location Sequencing Program - Sequence Products into Warehouse Location

The products in the accepted order record are sequenced on the warehouse location field, obtained when the product information was initially retrieved from the product data set. This sequenced record is then written back to the accepted order data set, and control is passed to the invoice calculation program.

Invoice Calculation Program - Extend Invoice

This program extends each line item of the order, based on unit price and product and customer discounts extracted from the relevant data sets during entry of the order. The sequenced order and calculated invoice are then transmitted to the warehouse.

Warehouse Transmission Program - Transmit Packing Slip and Invoice to Warehouse

The order, in warehouse location sequence, is prepared as a warehouse packing slip and transmitted to a printer in the warehouse. The calculated invoice is then formatted and transmitted to the same warehouse printer for preinvoicing.

If a confirmation invoice is to be sent to the terminal operator, a copy of the warehouse invoice is also transmitted to a printer located near the terminal operator.

Receipts Program - Update Stock Status with Receipts

This program accepts transactions from the warehouse indicating the receipt into inventory of products ordered against suppliers. The program accesses the product record, increases the stock status, and reduces the reorder quantity in the reorder data set based upon the quantity received.

The increased product quantity may be used online to satisfy back orders in the back order data set, if necessary, or these back orders may be processed offline.

RETAIL STORE SYSTEM

An extension of the previously described order entry and invoicing application applies to the retail store environment. Here, purchases made by customers are entered to directly update inventory levels, calculate the purchase amount (with discounts and sales tax), print a receipt, and calculate change. The need to print a warehouse packing slip and invoice is bypassed. The customer selects the required product from the store, and the receipt becomes the customer's and the stores record of the purchase transaction.

The 3650 Retail Store System is specifically designed for this environment. The 3650 system comprises a 3651 programmable controller with 9.3 million bytes of disk storage, a 3653 Point of Sale Terminal, a 3275 Display Station, a 3284 Printer, and a 3657 Ticket Unit. Refer to the 3650 Retail Store System Introduction, GA27-3075, for further information.

CICS/VS permits conversational sessions to be established from 3653 or 3275 terminals, and also supports a pipeline session for rapid authorization of credit transactions. An application program session can be established for communication between user application programs in CICS/VS and in the 3651. (See "3650 Sessions" in Chapter 3 for further information.)

DATA BASE SUPPORT SELECTION

The most appropriate data base support for the applications is discussed in the following. Refer to "Data Base Selection Criteria" in Chapter 5 for a discussion of the factors relevant to this selection.

DL/I Products

The advantage of DL/I in this application permits information relating to stock availability for a product (across several warehouses) to be readily identified. Sales information is included in the logical structure for the product data base (see Figure 11-32), together with the stock availability in various warehouses. Back order information for each product may be maintained across different suppliers, using the same logical structure.

DL/I may also be utilized to support the customer data base. Refer to "DL/I Products" in Chapter 5 for a typical customer data base logical structure.

CICS/VS File Control

The order-in-progress and -accepted order data sets will generally be direct access DAM or entry-sequenced VSAM data sets. This also applies to the reorder and back-order data sets which, together with the accepted-order data set, are created sequentially online, but may be retrieved in random fashion.

The customer data set will generally be ISAM or key-sequenced VSAM, while the product data set may be either DAM, ISAM, or VSAM.

Depending upon the size of the customer data set, disk storage savings may be achieved by using the segmented record feature of file control and defining the customer record as variable-length, with a variable number of variable-length segments for name, postal address, and ship-to address, for example.

If satisfactory performance is to be achieved, VSAM is not recommended for computers with less than 144K bytes of real storage. In this instance, use DAM and ISAM as the access methods.

LAW ENFORCEMENT INDUSTRY

A common online aplication in this industry is a police information system. In such a system, a data base containing all information relating to criminals is established and maintained.

POLICE INFORMATION SYSTEM

This online application is dependent upon the quick availablity of information and the establishment of logical relationships between items of information. The data base used to provide this necessary information includes records of criminals, alias names, personal characteristics, convictions, and _modus operandi_. Information relating to various crimes and suspects is also recorded. Personnel at online terminals must be able to access this data when given any of several items of information, such as name, alias, modus operandi, and convictions. The ability to maintain up-to-date, accurate information about various crimes, criminals, and suspects is a valuable record-keeping function. A typical police information system is illustrated in Figure 11-33.

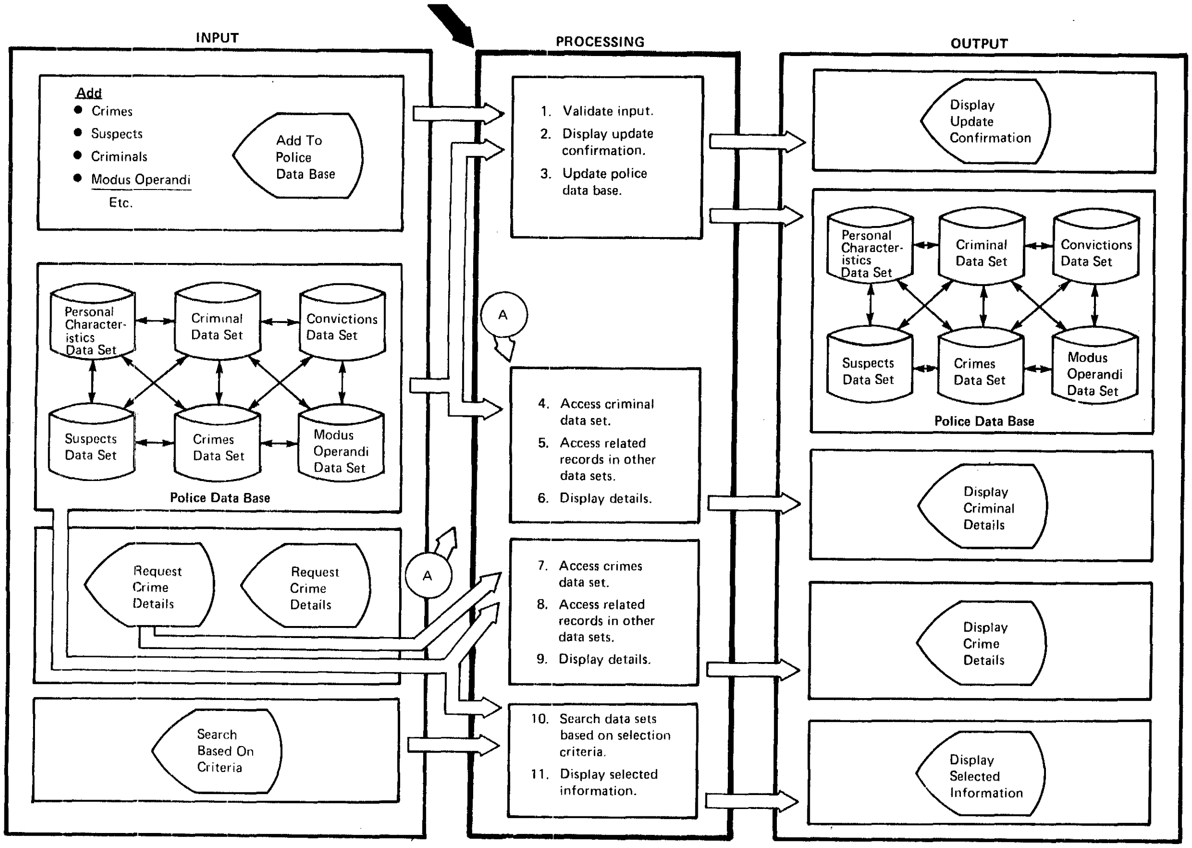


Figure 11-33. Police Information System

The computer offers its most significant advantages not only in establishing and maintaining a law enforcement data base, but also in the analysis of this information. For example, possible relationships between particular crimes and the modus operandi and characteristics of various criminals can be identified. Used in such a way, an online police information system enables the facts relating to a particular crime to be used in retrieving all information relevant to those facts; the online system then becomes a powerful law enforcement tool.

The STAIRS/VS Program Product (Program No. 5740-XR1) may be used in this environment. Refer to "Related Publications" in the Preface for relevant publications.

Both STAIRS/VS and CICS/OS/VS support the use of the 3850 Mass Storage System, which can be used to satisfy the massive online data base requirements of this application. The 3850 supports data bases ranging from 35 to 236 billion bytes online. It can be used for applications with low transaction volume which can tolerate long response times. (See Chapter 7 for further information on the 3850.)

A related application which can utilize the online storage capability of the 3850 is the legal profession. The 3850 permits legal reports, cases, research material, and legislation to be maintained online for search and retrieval based upon keyword criteria using STAIRS/VS.

DATA SETS

Either of two completely different approaches may be taken toward this application in the definition of data sets.

Figure 11-33 shows a criminal data set with separate data sets for criminal personal characteristics, convictions, aliases, and *modus operandi*. Similarly, separate data sets are shown for crimes and suspects. Figure 11-33 illustrates the logical relationships between these data sets, which are shown schematically by lines (representing pointers) from one data set to a logically related record in another data set. This data base structure is oriented toward the use of CICS/VS file control indirect accessing.

A similar data base capability may be provided through the use of DL/I. This is illustrated in Figure 11-34, which shows the logical structure of two data bases, a criminal data base and a crimes data base.

Information in the criminal data base includes personal characteristics, aliases, *modus operandi*, and convictions as dependent segments, of which there may be multiple occurrences for each type of segment.

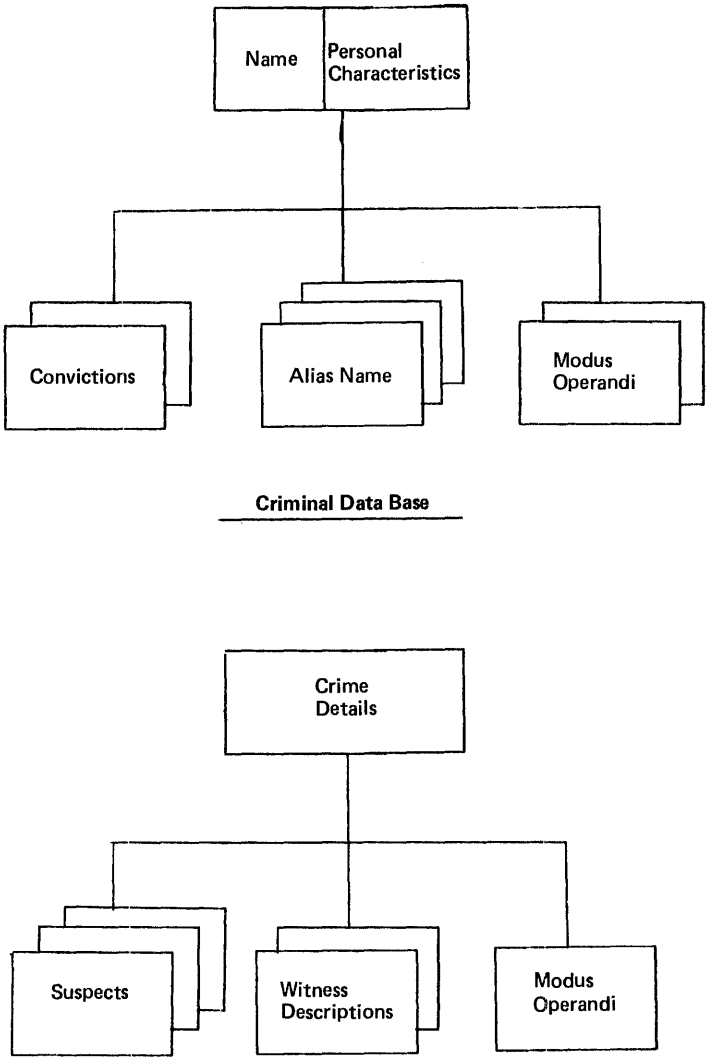
The crimes data base contains information relating to various crimes. It contains segments describing *modus operandi*, suspects, and descriptions, for example.

The selection of the most appropriate data base support is discussed following the description of the various online programs.

ONLINE PROGRAMS

Criminal Inquiry Program - Display Information Relating to Criminals

This program accesses all of the information relating to a specified criminal, including personal characteristics, aliases, *modus operandi*, and convictions, and formats that information for display as a series of pages. Those pages are presented to the CICS/VS terminal paging routine, which displays them on the terminal in the sequence, and as requested, by the terminal operator (see "Terminal Paging" in Chapter 3).



Criminal Data Base

Figure 11-34. Police Data Base Icgical Structures

Crimes Inquiry Program - Display Crimes Informaticn

This program accesses all of the information relating to specific crimes and suspects and prepares that information as a series of pages for presentation by the CICS/VS terminal paging routine tc the terminal operatcr, on request.

Add Program - Add Information to the Data Base

This program adds information relating to criminals, crimes, and suspects to the police data base, or changes existing information, thus enabling the data base to be dynamically updated and maintained online.

Selection Program - Select Information Based on Various Criteria

This program searches the available information relating to criminals, crimes, and suspects, selecting records which most closely meet various criteria supplied by the terminal operator. For example, the description and characteristics of a suspect may be used to search the personal characteristics and alias information, to determine if suspects are known criminals. Alternatively, the *modus operandi* of a particular crime may be used to search the *modus operandi* of all criminals to identify likely suspects.

The CICS/VS built-in weighted retrieval function (see "Weighted Retrieval" in Chapter 5) is particularly suited for this selection process. However, it can be utilized only for VSAM data sets.

DATA BASE SUPPORT SELECTION

As discussed previously, two distinctly different data base approaches may be used for this application. Refer to "Data Base Selection Criteria" in Chapter 5 for a discussion of the factors relevant to the most appropriate data base support for this application.

DL/I Products

Because of the volatile nature of information in this application, it is particularly suited to DL/I. The continual changes which are made to this data base, and the possible need to change the physical organization of this data base from time to time, make it best supported by DL/I, with its data independence and hence reduced program maintenance. Refer to "Segment Reference Design Factors" and Figure 5-34 in Chapter 5, for a discussion of logical structure design for this application. Figure 11-34 illustrates a logical structure of a typical police data base.

If the police data base contains a considerable amount of textual information, advantages may be gained by the use of variable-length segments with DL/I DOS/VS and IMS/VS DL/I. However, if disk storage capacity is not a significant factor, the textual information may be placed in fixed-length segments.

The secondary indexing capability of DL/I is particularly suited to this application, enabling indexes to be created by DL/I for direct retrieval of dependent segments, instead of by hierarchical retrieval through a logical structure.

This facility is supported by DL/I ENTRY, DL/I DOS/VS, and IMS/VS.

The support of logical relationships by DL/I ENTRY, DL/I DOS/VS and IMS/VS permits the design of sophisticated data bases for this application. Refer to "Logical Relationships" and Figures 5-35 and 5-36 in Chapter 5, for further discussion.

For retrieval of DL/I segments, an important consideration in the use of the file control indexes described above is the CICS/VS built-in weighted retrieval function. This powerful facility is available only for searching VSAM files, but cannot be used directly on DL/I data bases, unless the data base is a root-segment-only data base and of standard VSAM format. In the case of the police information system, the criminal and crimes data bases have a number of dependent segments. Consequently the weighted retrieval function cannot be used directly on these DL/I data bases.

However, through the use of VSAM for the file control indexes described above, the built-in weighted retrieval function may be used for searching these indexes and selecting relevant segment keys. The segment names related to selected keys may then be used to access the DL/I data base directly, by means of user-developed code.

Alternatively, if file control indexes are not desired, the search facilities offered by the weighted retrieval function may be readily implemented by user coding, searching the DL/I data base directly.

The query and selection facilities offered by STAIRS/VS may be used with data bases created and maintained by STAIRS/VS.

CICS/VS File Control

As described above, file control indexes may be created to identify all the DL/I segment names associated with a particular key. File control can also be used to support a police data base, with separate data sets created for criminals, personal characteristics, aliases, modus operandi, convictions, crimes, and suspects.

For example, the criminal data set contains all the information relating to a criminal, together with pointers to related records for the criminal in other data sets (see Figure 11-34).

Chaining techniques with file control, discussed in Chapter 5 and elsewhere in this publication, may be utilized if required. However, in many cases they will require extra user coding, and will introduce additional work for the creation and maintenance of these data bases. Because of the volatile nature of information in this application, and the multiple occurrence of dependent information, the use of DL/I rather than file control for this application will result in a significant reduction in user coding, data base creation, and maintenance, and a similar reduction in program maintenance when it is necessary to modify the data base. The availability of DL/I data base maintenance and recovery utilities, and the advantage of data independence with DL/I, is invaluable to this application.

UTILITIES INDUSTRY

A typical online application in this industry is a customer information system (CIS) as shown in Figure 11-35. It contains information relating to the utility companys customers, for example, name, address, type of account, current account balance, cash payments, service order history, merchandise installment contract, and meter histories.

CUSTOMER INFORMATION SYSTEM

The customer information system (CIS) is designed to improve the efficiency, quality, and accuracy of information and procedures used principally by the service departments and customer activities departments and, to some extent, the gas service, gas sales, electric service, electric sales, appliance service, and appliance sales departments where they exist within individual utilities.

The customer information system consists of bcth online and offline programs which are designed tc provide:

- An inquiry capability into the customer master data set

- A data entry capacity so that service orders or adjustments to the data base can be entered online

- Control of data entry activities, including the ability to monitor service crders frcm creaticn completicn

- A historical record of all inquiry or service order activities for online display at all times

CICS/OS/VS supports the 3850 Mass Storage System. This provides massive online data bases ranging frcm 35 to 236 tillion[1] bytes. It can be used for applications with low transaction volume which can tolerate lcng respcnse times (see Chapter 7).

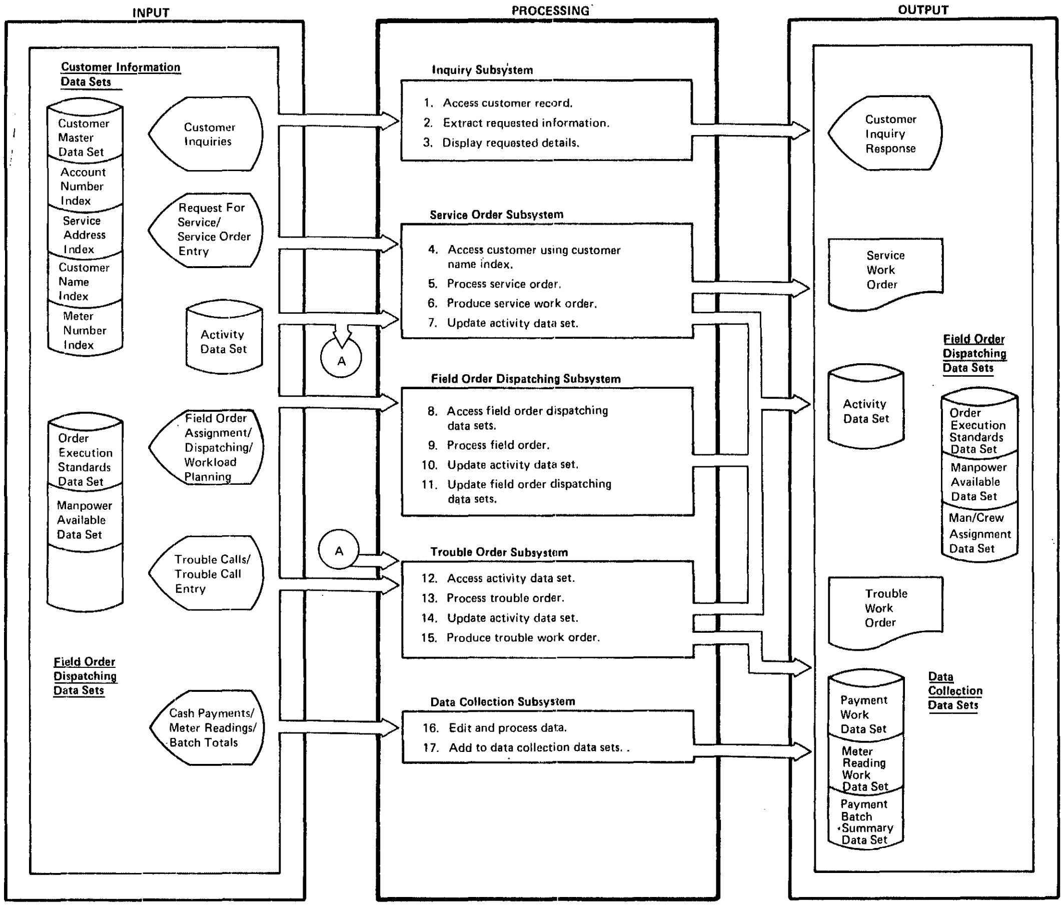
DATA SETS

The main data set used by this application is the custcmer data set. A typical customer reccrd is illustrated in Figure 11-36.

Customer Master Data Set

The primary CIS system data set is the customer master data set. It is made up of segments of information, each of which is a group of related fields (see Figure 11-36). Each record need contain only those segments that are required by that acccunt. For example, not all accounts contain a deposit segment, since deposits are not required from all customers. Also, each segment that is present can contain optional fields. Fcr example, the infcrmaticn in the meter segment can vary from account tc acccunt. These segments can be fixed, dependent upon the requirements cf each particular utility. There are basic segments of information that must exist, but no two companies will use the identical ccntent. However, the basic concept and record format should be the same in mcst companies. The control segment is always in every custcmer master record. Infcrmation in this segment allows CICS and data set maintenance modules to determine the specific format, contents, and relative lccation of each data segment in a given customer master record.

[1] One tillion equals 10^9.


INPUT
PROCESSING
OUTPUT
Customer Information Data Sets
Customer Master Data Set
Account Number Index
Service Address Index
Customer Name Index
Meter Number Index
Customer Inquiries
Request For Service/ Service Order Entry
Activity Data Set
Order Execution Standards Data Set
Manpower Available Data Set
Field Order Assignment/ Dispatching/ Workload Planning
Trouble Calls/ Trouble Call Entry
Field Order Dispatching Data Sets
Cash Payments/ Meter Readings/ Batch Totals
A
Inquiry Subsystem
1. Access customer record.
2. Extract requested information.
3. Display requested details.
Service Order Subsystem
4. Access customer using customer name index.
5. Process service order.
6. Produce service work order.
7. Update activity data set.
Field Order Dispatching Subsystem
8. Access field order dispatching data sets.
9. Process field order.
10. Update activity data set.
11. Update field order dispatching data sets.
Trouble Order Subsystem
12. Access activity data set.
13. Process trouble order.
14. Update activity data set.
15. Produce trouble work order.
Data Collection Subsystem
16. Edit and process data.
17. Add to data collection data sets.
Customer Inquiry Response
Service Work Order
Activity Data Set
Field Order Dispatching Data Sets
Order Execution Standards Data Set
Manpower Available Data Set
Man/Crew Assignment Data Set
Trouble Work Order
Data Collection Data Sets
Payment Work Data Set
Meter Reading Work Data Set
Payment Batch Summary Data Set

Extended Description

2. Extract following information, when requested:
   - Billing status
   - Billing history
   - Merchandise
   - Payments
   - Credit status

5. Process following requests:
   - Order entry
   - Order pringing (same day or future)
   - Order completion

9. Field order processing:
   - Order scheduling
   - Order assignment
   - Order dispatching
   - Workload statistics

13. Trouble order processing:
   - Trouble calls
   - Trouble processing

16. Data collection processing:
   - Cash payments
   - Meter readings
   - Control totals

Figure 11-35. Utilities Custcmer Information System

CIS Activity Data Set

The CIS activity data set is used to maintain all information concerning a customer account that is not yet contained in the customer master data set. The primary function of the data set is to maintain pending service crders. When a service order is taken, it is added to the data set as a pending crder. When the service order has been executed, completicn infcrmation is added. The completed service order is removed from the data set in a batch maintenance operation for offline programs. These offline prcgrams process the data and update the customer master data set.

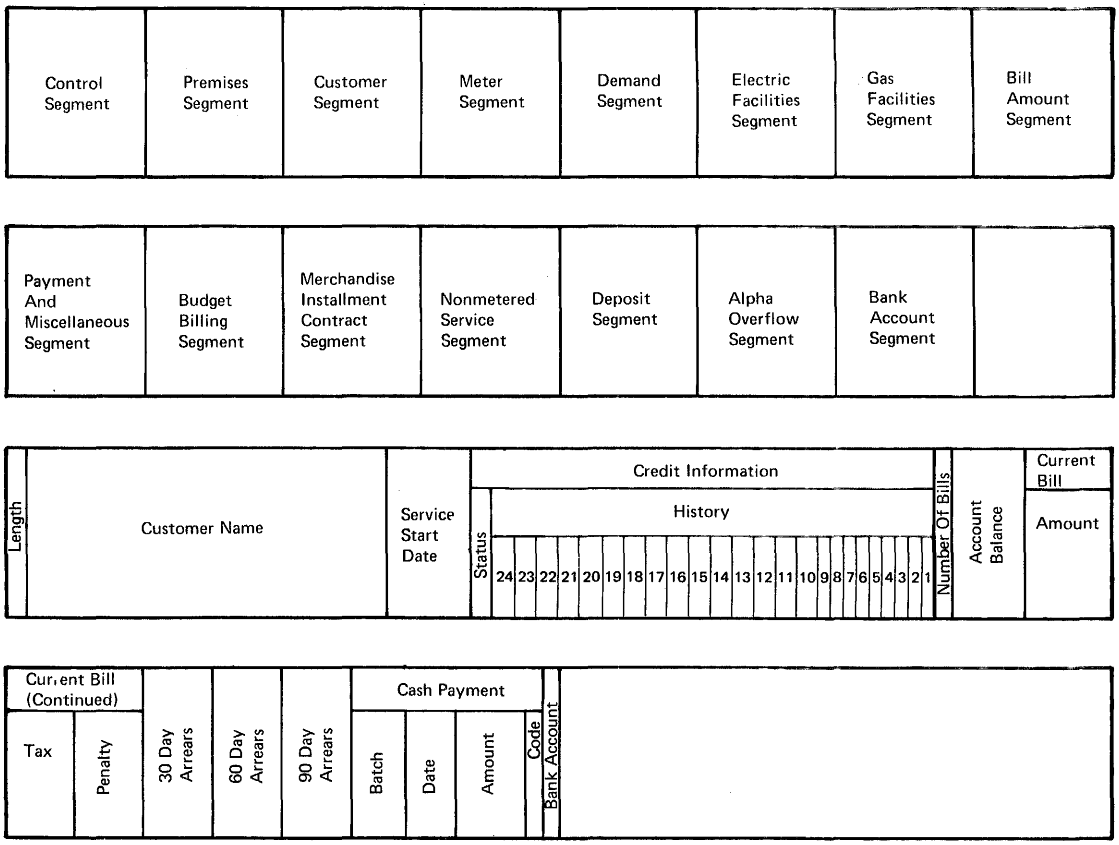


Figure 11-36. Utilities Customer Record Format

Support Data Sets

Additional data sets are normally included in CIS which support operations, but are not primary operational data sets are the customer master data set and the CIS activity data set. The primary among these can be termed manpower information data sets. These data sets contain serviceman/crew assignment information. Their primary functions include: (1) to assist in scheduling service order execution dates, (2) to assist in the assignment cf service orders to a specific serviceman or crew, and (3) to assist in the dispatching of service orders.

ONLINE PROGRAMS

The online programs which are used in this application are described shortly. The application cnline functions are performed by modules which are linked to cne another randomly, depending upon the processing required by any given online transaction. The functions performed are as fcllows.

ONLINE PROGRAMS

- Inquiry programs -- provide the ability to query the CIS data base for billing, payment, account status, outstanding service work, and credit status and collecticn information.

- Service order programms -- provide the ability for online entry of such service orders as turn-cn, turn-off, meter order, name correction, billing adjustment, the printing of service orders, and the entry cf order ccmpleticn information.

- Field order dispatching programs -- provide the ability (1) to assign service orders to a service-man/crew cnline, (2) to dispatch and maintain the status cf service orders online, (3) to inquire into the data base to obtain workload summary information, and (4) to change basic manpcwer data in the data base online.

- Trouble order programs -- prcvide the ability to enter trouble information online and tc print this informaticn on a hard-copy terminal in the trouble dispatching center.

- Data collection programs -- provide the ability to enter cash payment and meter reading information into the system from the remote locations.

Notation Program - Note Special Account Information

This program allows special instructions to be entered from the terminal and added to the customer reccrd, for subsequent use in processing that record.

Search Program - Select Custcmer Accounts Based on Specified Criteria

This program searches through all or a specified section of the custcmer data set, retrieving all records which most closely meet specified selection criteria. Fcr example, all account records in a particular street, cr building, cr with certain appliances may be selected for display. Alternatively, all records whose accounts are in arrears by more than a specified amount for a specified period of time can be selected and displayed. This program can also utilize the CICS/VS built-in weighted retrieval function (see "Weighted Retrieval" in Chapter 5) for selection cf records meeting specified criteria from key-sequenced VSAM data sets.

DATA BASE SUPPORT SELECTION

The customer data set ccntains a variety of infcrmation which may be present or absent in different records. Either DL/I or CICS/VS file control can be used. Refer to the section "Data Base Selection Criteria" in Chapter 5 for a discussion of the factors relevant to the most appropriate data base support.

DL/I Products

A customer data base may be set up as illustrated in Figure 11-37, showing the multiple occurrence cf dependent segments such as address segments, installment contracts, payments, or gas and electric facilities.

Furthermore, the additicn of special notation information to the customer record may be achieved by adding special notation segments to the account history segment, with as many special notation dependent segments as necessary.

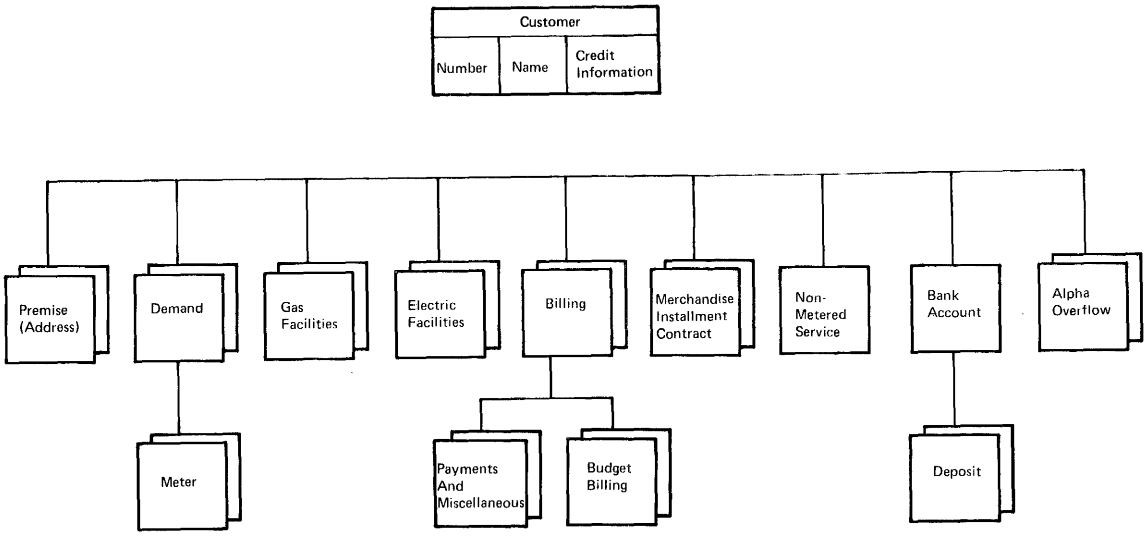


Figure 11-37. Customer Data Base Logical Structure

While the use of variable-length segments is advantageous if IMS/VS DL/I or DL/I DOS/VS is used, fixed-length segments with DL/I ENTRY are also feasible.

CICS/VS File Maintenance

The customer information system (CIS) data tase shculd be built around the use of VSAM. The high-response nature of CIS and the large volume of transaction processing that cccurs offline make disk prccessing efficiency critical. Cffline programs usually process several cycles daily, representing 15 to 25 percent of the entire customer master record. File maintenance and building runs often take several hours per day and any small degradations can be multiplied into performance prcblems. Usually, a sequential or skip sequential file organization technique built around VSAM can test handle the requirements of CIS.

VSAM should also be used to support CIS activities. Usually, little direct file maintenance of the customer master reccrd is performed online, tut rather is handled offline. Use of VSAM provides a compatible data tase organizaticn compromise between online and offline operation.

Information within each record cf the customer master file is organized using the CICS segmented record feature, which allows data to be grouped by frequency of use, function, and logical relationship. Accordingly, it is possitle to retrieve an entire record or selected segments of a given record as apprcpriate.

To permit the addition of segments online, such as by adding pending order segments, the access method used should te key-sequenced VSAM, which allows the record length tc be increased or decreased as required. However, VSAM should not te used (if satisfactory performance is to be achieved) cn systems with real storage less than 144K.

SH20-9002-2

CICS/VS System/Application Design Guide Printed in U.S.A. SH20-9002-2

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)