IBM System/3
Communications Control Program
System Design Guide

Program Numbers:
- 5702-SC1 (Models 8 and 10)
- 5703-SC1 (Model 4)
- 5704-SC1 (Model 15)
- 5704-SC2 (Model 15)
- 5705-SC1 (Model 12)

Feature 6011/6012/6033/6070/6071

GC21-5165-1
File No. S3-36

IBM System/3
Communications Control Program
System Design Guide

Program Numbers:
- 5702-SC1 (Models 8 and 10)
- 5703-SC1 (Model 4)
- 5704-SC1 (Model 15)
- 5704-SC2 (Model 15)
- 5705-SC1 (Model 12)

Feature 6011/6012/6033/6070/6071

Second Edition (September 1980)

This is a major revision of, and obsoletes, GC21-5165-0 and Technical Newsletters GN21-5596, GN21-7969, GN21-5656, and GN21-5288. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change or addition.

This edition applies to the System/3 system control program versions listed below and to all subsequent versions and modifications until otherwise indicated in new editions or technical newsletters:

Version	Modification	Program Number	Feature Numbers	Model
16	00	5703-SC1	6033	Model 4
16	00	5702-SC1	6033	Models 8 and 10
06	00	5705-SC1	6070, 6071	Model 12
08	00	5704-SC1	6033, 6070, 6071	Model 15A, B, C
05	00	5704-SC2	6011, 6012	Model 15D

Changes are periodically made to the information herein; these changes will be reported in technical newsletters or in new editions of this publication.

Use this publication only for the purposes stated in the *Preface*.

Publications are not stocked at the address below. Requests for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

This publication is a guide to designing an IBM System/3 system that uses the Communications Control Program (CCP). The manual is primarily intended for customer system analysts/programmers and IBM system engineers. This publication is applicable to System/3 Models 4, 8, 10, 12, and 15. The following subjects are included:

Application Design Concepts: Describes the considerations for using various CCP program types, programming facilities, and programming concepts in designing applications to be run under CCP.

Direct Files: Describes the use of direct files in a CCP environment and describes techniques for designing direct files.

File Sharing: Describes the necessity for file sharing in a CCP environment, and describes file sharing capabilities and restrictions on the various System/3 models.

3270 Screen Design: Describes the human factors and CCP performance aspects of designing screen formats for the IBM 3270 Information Display System.

Use of Printers Under CCP: Describes the system design and performance considerations for using the system printer and terminal (remote) printers with CCP.

Task Chaining: Describes how the System/3 Model 15D CCP task chaining facility can be used in CCP system design.

Sort Under CCP: Describes system design aspects of using CCP/Disk Sort to sort files under CCP.

System Security/Integrity: Describes the special considerations involved in maintaining system security in a CCP environment. Describes procedures for ensuring the accuracy of information processed by the system and stored in the system files.

Queuing Theory: Describes the application of simplified queuing theory to CCP system design.

Performance Tips: Provides miscellaneous tips and techniques for improving CCP performance.

This publication references several specific terminals in the 3270 Information Display System. These references are for illustration only and not necessarily representative of all 3270 devices supported. For a list of 3270 devices supported by System/3 CCP, refer to the *IBM System/3 Communications Control Program System Reference Manual*, GC21-7620.

This publication is oriented to the typical CCP user: one who wants to use RPG II and 3270 terminals with the CCP Display Format Facility (DFF). The reader who plans to use 3270 terminals is assumed to have knowledge of 3270 operation and design, either from IBM education, previous reading, or experience. However, usefulness of the manual is not limited to RPG II/3270 users. Most of the design concepts are applicable regardless of the programming language and terminal type used.

This guide is intended for readers who have had CCP programming instruction and who have attended, or are attending, the CCP system design class. The reader is expected to use this manual to refresh his or her knowledge, as supplementary reading, and for subsequent reference. The reader is assumed to have previous experience and/or training in:

- Programming and operating System/3 with CCP
- System/3 disk file organizations and processing methods
- The particular terminal devices to be used

Related Publications

The publications related to CCP and System/3 data communications are listed in Appendix B. See *IBM System/3 Bibliography*, GC20-8080, for descriptions of other System/3 publications.

Contents

Chapter 1. Introduction

A communications-based system like System/3 with CCP, is made up of dissimilar elements (personnel, programming, devices) that operate at totally different speeds. These elements are:

- Terminal operators
- Terminals
- The communication facility
- The system operator
- The processing unit and associated I/O devices
- Application programs

These elements must perform well together to satisfy user requirements. Good system performance requires good system design.

SYSTEM PERFORMANCE

Performance requirements vary widely among CCP users, but there are quantitative and qualitative aspects of performance that are important to all users.

Quantitative Aspects

- System throughput
- Terminal response time
- Business response time

Qualitative Aspects

- Ease of use
- Expandability/changeability
- System security and integrity

System Throughput

System throughput is the total volume of work performed by a computing system over a period of time. Throughput is a primary concern of the CCP user and should also be the direct concern of the designer. Throughput is a measurable aspect of performance. The designer of an online system frequently measures throughput in terms of the number of transactions per unit of time (hour, day) that the system can handle. The users of the system may think of throughput as the number of invoices, orders, or inquiries the system can process over a period of time.

Terminal Response Time

To provide a viable system, the designer must balance system throughput requirements against terminal response time requirements. Terminal response time is the time interval from when the terminal operator enters data to the system until the keyboard is opened to permit more data to be entered. Response time requirements vary greatly among CCP users and even among applications for a single user. For example, a 10-second to 30-second response may be adequate for an inquiry application that is used occasionally, but a 1-second to 3-second response may be required for high-volume data entry and order entry applications.

Minimum response time results when there are no *queues* in the system; that is, there are no units of work waiting to be serviced by the different system facilities: disk, processing unit, communication lines, or terminals. Chapter 10, *Simplified Queuing Theory* further describes response time and queues.

Business Response Time

Business response time is the total duration of time required to satisfy a user (customer) transaction, inquiry, or request. A complete business response may require several terminal responses, but the entire response must be accomplished at one terminal even if other system resources are idle. Individual terminal response times surely contribute to good business response, but good system design plays the most important part.

Ease of Use

Ease of use is another important aspect of system performance. If use of the terminal is easy for the operator, if screens are carefully designed, if operator keying is kept to a minimum, if applications are broken into efficient, logical steps, and if the terminal operator is shielded from the internal workings of the system, the terminal operator can be more productive, make fewer mistakes, and thus contribute to improved system throughput.

Ease of use is important for other users also, such as the system operator and the application programmer. Clear and complete operating procedures should be defined and documented to govern the system operator's actions under all conditions. Recovery and backup procedures should be carefully considered in the system design and should be clearly documented for the system operator. Application programs should be written with clear, straightforward logic and should be well documented. Standard methods of return code checking and error handling should be considered throughout the design of the user application programs. Careful attention to program design will result in programs that have fewer errors and are easier to correct or modify.

Expandability/Changeability

Expandability and changeability are long-term measures of system performance. If a system design does not anticipate future expansion or changes, the system may initially perform adequately, but require a major redesign when facilities or functions are added. The system designer should consider effects on the system of an increased transaction load, system hardware changes, or additional applications.

System Security and Integrity

System security and integrity are other long-term aspects of performance. The concepts of system security and integrity include requirements such as:

- Transaction logging
- Audit trail
- Control procedures
- Data security
- Backup and recovery

These requirements must be an integral part of the system design. If system security and integrity are lacking, serious problems can occur when an audit is required, when there is a system failure, when an error occurs, or when an unauthorized person attempts to access and perhaps modify data in the system.

SYSTEM DESIGN

Because online applications usually have a significant, overall impact on the way the user organization is run, it is vital that the eventual users of the system–management, programmers, operators, and user departments–be involved in the design process from the beginning and that all of their needs and wants are considered. In many cases, a phased approach to design and implementation, with simple inquiry being the first online application, allows a gradual changeover from current methods, provides early success, and allows the system designer to gain experience that can be applied to later applications.

During the system design process, each facility and application should be justified on its own, either economically or as a necessary part of an integrated system. Otherwise, the design process can become unnecessarily complex and costly, and may result in a system that has more capacity than is required for the necessary functions and anticipated growth. It is sometimes better to use batch processing or manual methods for certain jobs and to use the online resources in areas where they are clearly justified.

The system design process must be guided by cost constraints and application requirements. To be successful, the system designer must understand the benefits that are expected from the online system and must gather a great amount of data concerning the proposed applications.

Benefits Offered by CCP

Online processing with CCP offers benefits that are not possible in manual or batch data processing systems. The following benefits might be analyzed differently in terms of importance to the business, ease of implementation, cost of facilities, measurability, and whether the benefits will be immediate or in the future.

Fast Access to Information

Inquiry applications can provide a fast answer for a customer who is on the telephone or in the office. Inquiry applications can provide improved customer service, reduced look-up time, and fewer computer printouts. Inquiry applications are a good place to start, since they are often easy to define and implement.

Improved Worker Efficiency

Because of the necessity for queuing and routing work through various departments, many manual systems are inefficient, inaccurate, and time-consuming. Online applications enable an organization to operate on information as soon as it is available, which results in greater worker productivity. Online applications can result in: less time spent waiting for information, less time spent transferring information from one area to another, less time spent restudying information that has been delayed in processing, fewer requests for additional information, and improved accuracy.

Reduced Job Complexity

Complex calculations, procedures, and detailed requirements can make a clerk's task highly subject to error. Applications such as order entry, pricing, and accounts receivable can require considerable system design effort because the system is handling many complexities, but the benefits of improved accuracy and improved customer service can be valuable.

Improved Resource Control

Applications such as inventory control in manufacturing or distribution industries and reservations in hotels can enable many interdependent users to access current information. Improved control of limited or time-dependent resources is often essential to the functioning of an organization. A well-organized system design effort is required to redirect the information flow around the centralized system.

Design Data

The CCP system designer must gather and analyze a large amount of data, including:

- Application descriptions
- Messages and line control activities
- Transaction descriptions
- Terminal locations
- Computer location
- Present communication network layout and description
- Data security requirements
- Man-machine interfaces
- Message formats
- Message destinations
- Traffic statistics
- Effective speeds of equipment and operators
- Transmission line error rates
- Rate of system growth
- Throughput objectives for both online and batch processing
- Terminal response time requirements–business response requirements
- Logical file information
- Type of error recovery required
- Other information, determined in part by the proposed applications

Some of this design data must be particularly detailed because of the important affect it has on network design and, therefore, the performance of the system:

- Message/transaction types
- Message/transaction length
- Messages/transactions per given time period
- Message/transaction input-output ratios
- Message/transaction priority classifications
- Response time requirements
- Peak traffic determination
- Growth projections

Traffic Peaks

Traffic peaks must be considered; it is not sufficient to total the work for a day and divide by the number of hours to get an average. For example, a plotting of the arrival of orders in a business might look like Figure 1. The system should be designed to handle peak loads. It may be possible for certain orders to be held off and entered during slack periods to level the peaks. For example, in Figure 1, all mail orders are received at 10 a.m. Perhaps the mail orders that cannot be processed in the morning can be held off until late in the afternoon so that telephone and walk-in orders can be processed on a real-time basis during peak periods. In no case should the average load be the maximum load capability of the system.

Regardless of whether design is done manually or with the assistance of specialized network design aids provided by IBM, the design data collected is needed to determine:

- Line and terminal loading
- Facility utilization
- Response time
- Queues in the system

Queuing Theory

An understanding of queuing theory (Chapter 10) helps the system designer to use the design data to estimate how much utilization to expect of a system resource and to evaluate a design at intermediate stages. However, resource utilization in a CCP system is not a simple consideration and few designers can depend solely on queuing analysis in designing CCP systems. Utilization of resources can change from minute to minute and there are so many variables that to pick any set and say *this is my system* is usually invalid. Also, reliable data may not be available early in the design stage.

CCP system design requires some experimentation and adjustments to the initial design to improve performance. Overdesign is often a desirable approach. For example, the designer can allow for more disk activity or communication line activity than he actually expects, or can purposely overestimate the number of transactions per hour that will be entered from terminals. Judicious reductions in the areas of overdesign can be the key to success of the design project.

IBM Design Aids

The CCP system designer should be aware of the IBM design aids available (such as performance analysis programs) and what benefits can be gained from each.

One such design aid for the Model 15D is the System Measurement Facility (program 5799-AYQ). When enabled, this facility assembles status reports of the operating system and selected I/O devices. This information is useful for workload balancing within the CCP system. Contact your IBM representative for information concerning this and other design aids.

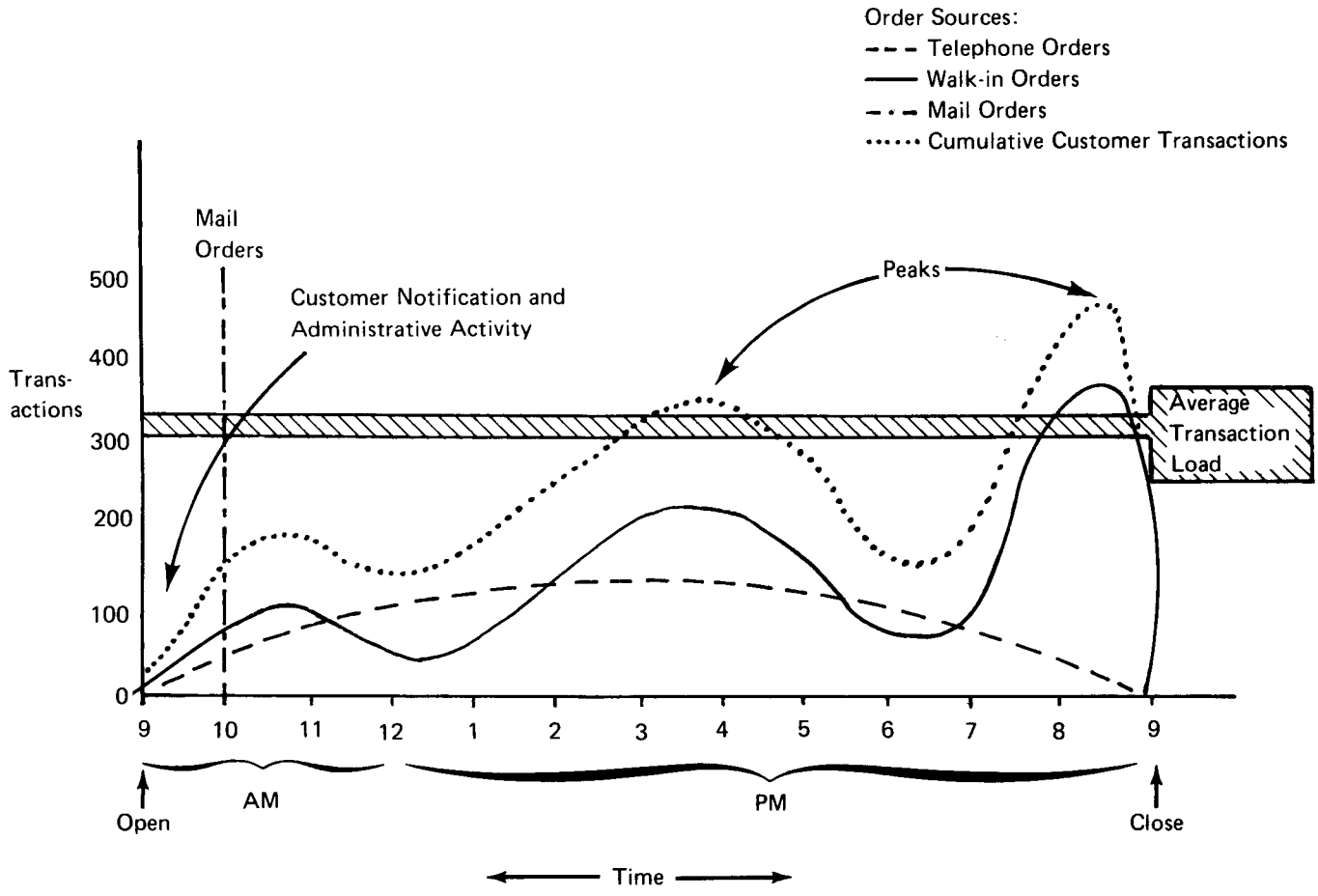


Figure 1. Transaction Peaks

Chapter 2. Application Design Concepts

This chapter presents broad concepts of implementing applications in an online terminal or work station environment.

ESTABLISHING APPLICATION GOALS

An application designer must establish specific goals for applications. In addition to specific application goals, the following general goals should be considered for all applications:

- Ease of use

  Design decisions should be made in favor of ease of use, not ease of coding or design. The person who uses the terminal should see the system as an easy-to-use tool. He expects to receive guidance throughout the application in the form of messages and definitions of the various options permitted. However, once an operator understands his task well, he should not be burdened with replying to repetitive messages nor with reading long text to understand where he is in the application. For example, if a 3270 display is being used and various function keys are allowed, present the user with a legend of the functions in a specific area of the screen where the user can read it or easily ignore it if he is familiar with the application. Error messages should also be confined to *one* area of the screen. High intensity and the audible alarm can be used to draw attention to the message. Use default options where possible to eliminate repetitive keying. For example, if special discounting is allowed but seldom used, let the operator specify when he wants it, but default to normal when he specifies nothing. See Chapter 5, *3270 Screen Design* for further information concerning operator ease of use.

- Logical and simple flow

  Whether you solve the application problem with one program or many programs using PRUF (program request under format), the operator must never have to understand how the application was coded. If you design an application using several programs, lead the operator through the sequence of programs by including the name of the next program to be called on the display.

- Concurrent utilization of system resources

  Dedicating most of the resources of the system to one user at the expense of other concurrent users can cause dissatisfaction. Some examples of this are: allowing an application to be written as a large program, so that other users cannot load their programs because memory is not available; specifying NOSHR (no share) for files on the PROGRAM assignment statement so that a long running program has a file dedicated to it at the expense of other users who need access to the same file; running a program that loops or does complex calculations and thus makes excessive use of processing unit instruction cycles (cycle bound). Consider that applications will be run at the same time and that other applications may be added to the system at a later time.

  Concurrent utilization of resources may increase the response time of the terminal. But response time is not the most important indicator of performance. For example, consider the following:

  An application is designed using two methods. The first method uses one large program that remains in memory. Two operators are employed and response time averages 1.5 seconds each time the enter key is pressed. One hundred characters are keyed for every enter key operation.

  The second method of design uses a series of three programs, each small. The programs are loaded as required. The average response time is 2.5 seconds because program loads must take place. With the first method, there is no memory remaining for other concurrent applications; in the second, there is. Since all keying is done *offline*, that is, between program loads, the processing unit has processing time and main storage available to service other users. Hence, the total work accomplished may very well be greater in the second case.

- Minimum scheduling

  The main advantage of online applications is that the system can be responsive to the end user–it can do work when he has work to do. If there are fewer terminals than operators or more applications than operators, obviously scheduling has to be done. However, an application design where there are sufficient operators and terminals available, but because of the design they cannot all have access to the system, should be considered unacceptable.

- Efficient design

  When the first application is designed for a system, the probability that more applications will be added later–some known and some unforeseen–must be considered. This requires writing efficient code, designing files for the fastest processing, and using as little main storage as possible. A batch program is usually judged by whether it does the job and produces accurate results. Online jobs must be judged by those factors but also by size, modularity, function, flexibility and response time. The key to efficient design is that the system resources must be shared, and rationing the resources from the beginning will ensure that the maximum potential for system growth can be realized.

CCP APPLICATION PROGRAM TYPES

In order to use CCP most efficiently, the application designer must understand the CCP application program types, classified on the basis of:

- The number of requesting terminals the program can handle–one, or more than one.

- The number of different kinds of functions or transactions the program can handle–one, or more than one. (A transaction is defined as the entry of some unit of data, the processing of that data, and the return of some response or answer.)

Specific coding examples of each type of program are available in the *CCP Programmer's Reference Manual*, GC21-7579.

SRT (Single Requesting Terminal) Program

An SRT program is one that can service only one requesting terminal on each execution of the program. Each time the program is requested, CCP must load that program into main storage. If two terminals request the same program, each will have its own duplicate copy of the program in main storage.

An SRT program can communicate with other terminals while servicing a requesting terminal, but these would be data terminals and would be attached to the executing program either by being specified in the assignment set or by being acquired by the program using an acquire terminal operation code. In most cases, an SRT program handles one terminal, which is the requester.

MRT (Multiple Requesting Terminal) Program

An MRT program can service more than one requesting terminal on each execution of the program. An MRT program, like an SRT program, is initiated by a requesting command terminal. However, any subsequent requests for the program by other terminal users cause the additional requesting terminals to be attached to the program already in main storage, rather than to a separate copy. The maximum number of users of the program is specified in the assignment set. Thus, CCP controls the number of concurrent users of the MRT program.

When a terminal attached to an MRT program is finished processing, it is released from the program under either program or operator control. If other terminals are attached, they continue to process. If there are no other terminals attached at this time, the program ends (the program logic must set on LR in RPG II). Another request for the program causes it to be again loaded from the object library.

The system designer should consider using an MRT program to minimize the impact of program loading when a program is called often and must be loaded into main storage frequently during the course of a CCP run. Program load time is dependent on the access time of the specific disk drives used with the system; for example, a program load from a 5444 is slower than from a 3340. See Chapter 10, *Simplified Queuing Theory* for further information about how to determine when program residence in main storage is high enough that an MRT approach should be considered.

Single Function and Multiple Functions

In addition to the SRT/MRT distinction, the system designer must understand the distinction between single function and multiple function programs.

If a program can only handle one kind of function, the program is a single function program. The characteristics of a single function program are:

- Program logic determines when the program terminates.
- Coding is straightforward and efficient.
- There is duplicate code if more than one copy resides in main storage.

If a program can handle multiple functions, the program is a multiple function program. The characteristics of a multiple function program are:

- Either the operator or program logic can determine when the program terminates.
- Some coding logic is required to determine which function is to be performed.
- There is unused code in the processing cycle (code for functions not performed still takes up main storage space).

When multiple functions are performed, the functions are either *independent* of each other or *dependent* on each other.

Summary of CCP Program Types

Combining these concepts–number of requesters; number and relationship of functions–yields the following breakdown of CCP program types:

- Single function SRT (SF/SRT)
- Multiple independent function SRT (MIF/SRT)
- Multiple dependent function SRT (MDF/SRT)
- Single function MRT (SF/MRT)
- Multiple independent function MRT (MIF/MRT)
- Multiple dependent function MRT (MDF/MRT)

Single Function SRT

The single function SRT program type is the easiest to write. A terminal operator requests a program (for example, an inquiry), the program performs the function and terminates. CCP attaches the terminal to the program at program initiation and automatically releases it at program termination. No code is needed in the application program to perform either of these functions.

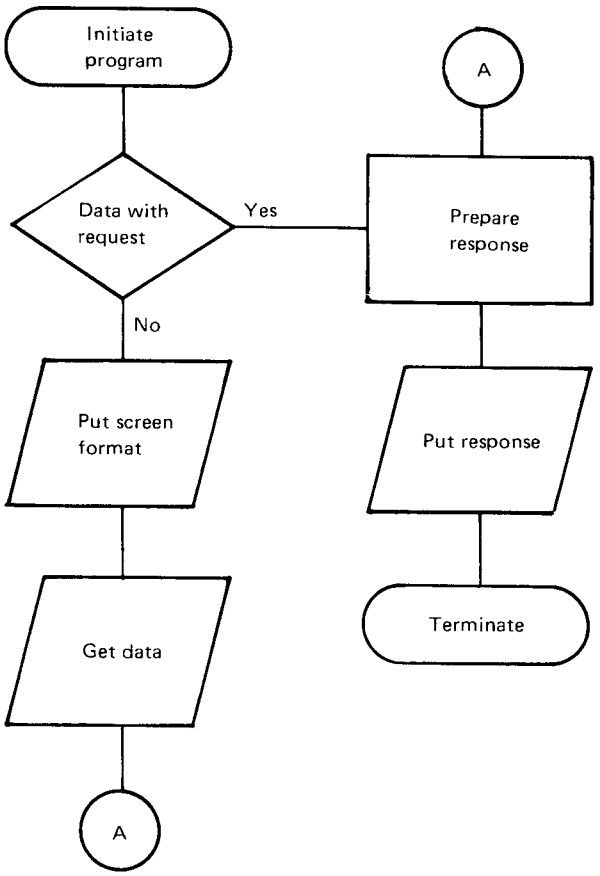


Figure 2. Receiving Program Data in an SRT Program

PRUF (Program Request Under Format)

Figure 2 shows two methods of receiving data from a terminal in an SRT program: with the program request or after the program has been initiated. Where possible, the data should accompany the program request. Putting a format and waiting for the operator to respond ties up parts of main storage during the keying operation. Another operator's response time can be adversely affected if main storage is not available. However, in order to enter data with the program request, the terminal operator must know exactly what data format is acceptable. This contradicts one of the basic CCP system design guidelines: ease of use. But because CCP has the PRUF facility, good utilization of storage and ease of use at the terminal can be achieved. The program now becomes two programs with the flow described in Figure 3.

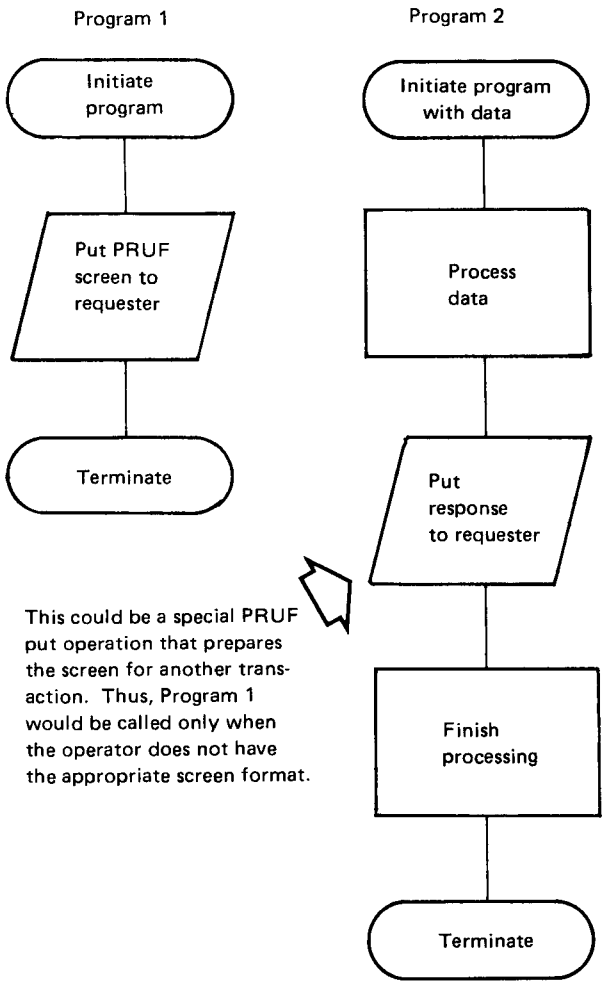


Figure 3. PRUF Concept–Single Function SRT Program

Single Function MRT

A single function MRT requires minor additional code compared to the previous example. The additional code is required because an MRT must not go to end of job unless all users are finished with the program. Therefore, rather than simply terminate, the program must release the requester and ask CCP if there are any other terminals requesting the program. If there are, the cycle must be performed again. Still using the PRUF approach, the program logic shown in Figure 3 would be modified as shown in Figure 4.

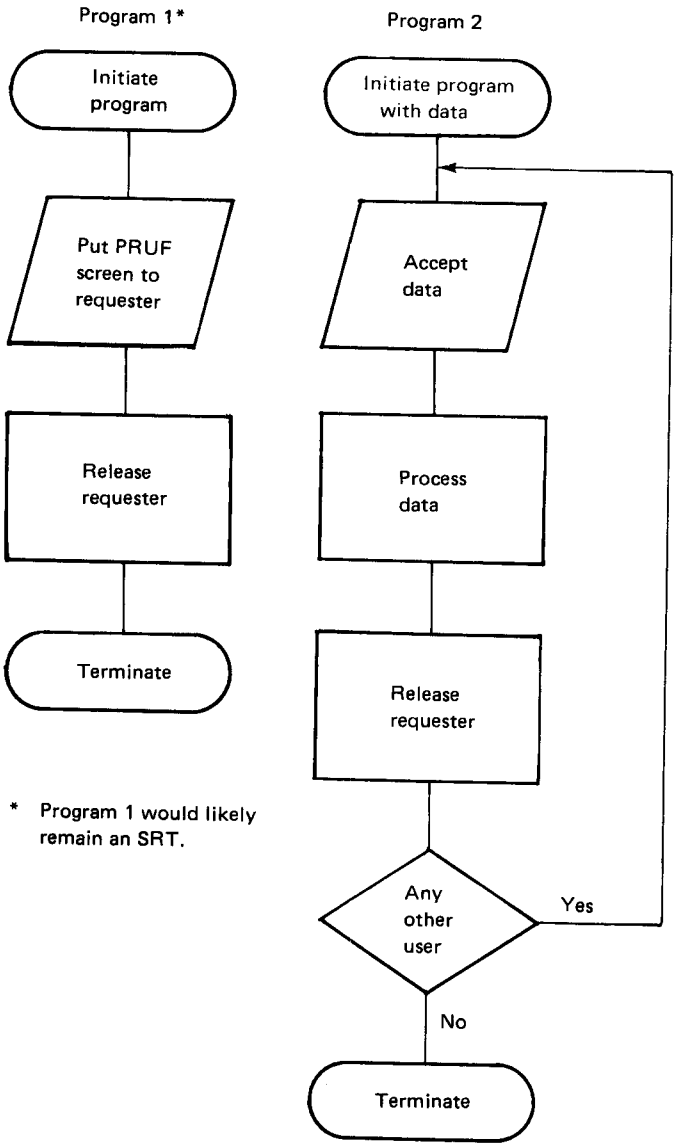


Figure 4. PRUF Concept–Single Function MRT Program

Comparison of Single Function SRT and MRT

The major considerations in choosing between an SRT or an MRT approach are use of main storage and response times. (The terminal operator does not do anything differently between the two approaches.) The advantage of the SRT approach is the ability to have multiple copies in main storage concurrently. The advantage of the MRT approach is the potential for reduced response time, since the program does not have to be loaded for each request. Each approach also has a disadvantage: the SRT must always include a program load in its response time, and the MRT allows only a single terminal to process through its code at one time. (Terminals that are queued to an MRT have a correspondingly greater response time.) The decision for either approach is examined further later in this chapter.

Multiple Independent Function SRT

Combining the programs in Figure 3 results in the logic shown in Figure 5.

This logic represents a multiple, independent function SRT program or MIF/SRT. This complicates the code somewhat, but cuts down by one the number of programs needed. In Figure 5, a small amount of code has been added to one program while doing away with another. (The program must check for a 02 return code, which indicates that initial input was not received with a previously put PRUF format.)

A logical question would be: if it is good in some instances to combine two functions, why not more? Although the program in Figure 5 is performing two functions, the terminal operator works in the same way as with the single function SRT and the increase in response time is minimal. If, on the other hand, more functions are added as shown in Figure 6, the program must monitor input and determine which subroutine to follow. Since CCP is itself a monitoring program, it seems redundant for an application program to do this. If the functions are quite similar and all use the same file, then the effect on performance may not be significant. However, if the program becomes quite large due to code or additional files, then the user is affected by the reduction in available storage. In some cases, file utilization could be affected as well. Finally, the program takes longer to load than the smaller, single function SRT.

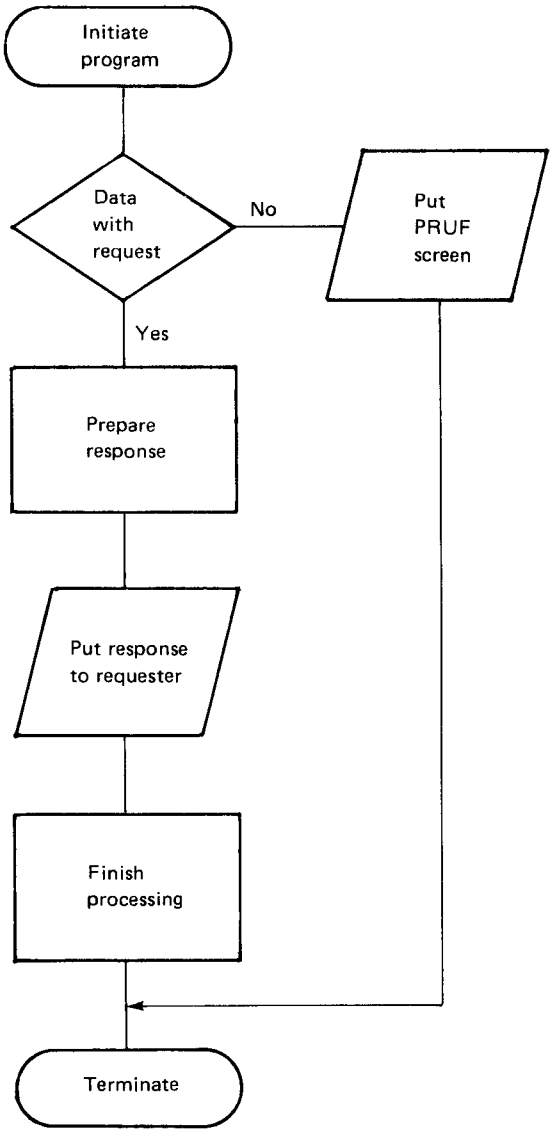


Figure 5. Multiple, Independent Function SRT Program

Multiple Independent Function MRT

MIF/MRT programs differ from MIF/SRT programs only in the method of releasing terminals and terminating the program. However, in some situations, MIF/MRT programs provide advantages. For example, main storage in System/3 Models 8, 10, and 12 is more limited than in Model 15. Also, for any of these models, one program level can be devoted to batch work while the other level is used for online processing with CCP. An MIF/MRT may be useful in this environment because the limited main storage available for CCP tasks may make it impossible to load multiple copies of an SRT program.

Because the program can perform several independent functions, it must contain logic to determine which function to perform.

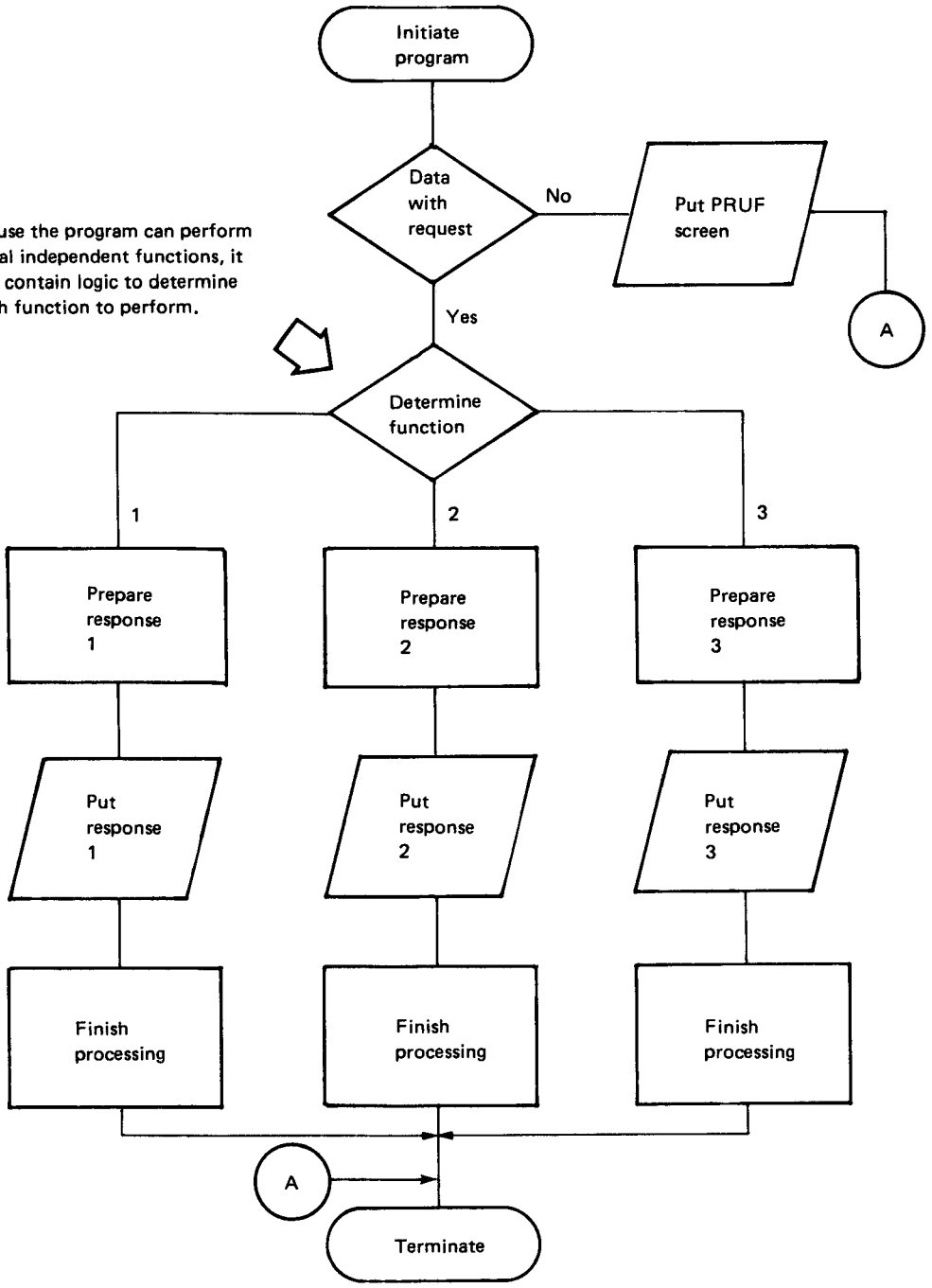


Figure 6. MIF/SRT Program with More than Two Functions

Multiple, Dependent Function SRT

Figure 7 is an example of MDF/SRT program logic. This type of program is straightforward, but results in large programs that remain in main storage until the entire process is completed. Main storage is occupied even though there is no processing being done by the system. Each process could be done in a separate program: by using PRUF PUTs to prepare the terminals for the next step, storage will be used only when there is data to process, and not during the keying time of the terminal operator. The effect on the operator of using many single function SRTs to perform the same functions would be to increase the response time for each step by the program load time. This increase in response time must be weighed against the delay that could occur if MDF/SRT programs are used and main storage is not available when another operator requests a program. Except for response time, the user should not know which type of program (MDF/SRT or single function SRTs) the programmer chose to implement. If the number of users is small, the MDF/SRT would give a better response; however, the programmer should be advised that a growth in number of work stations will probably lead to a redesign.

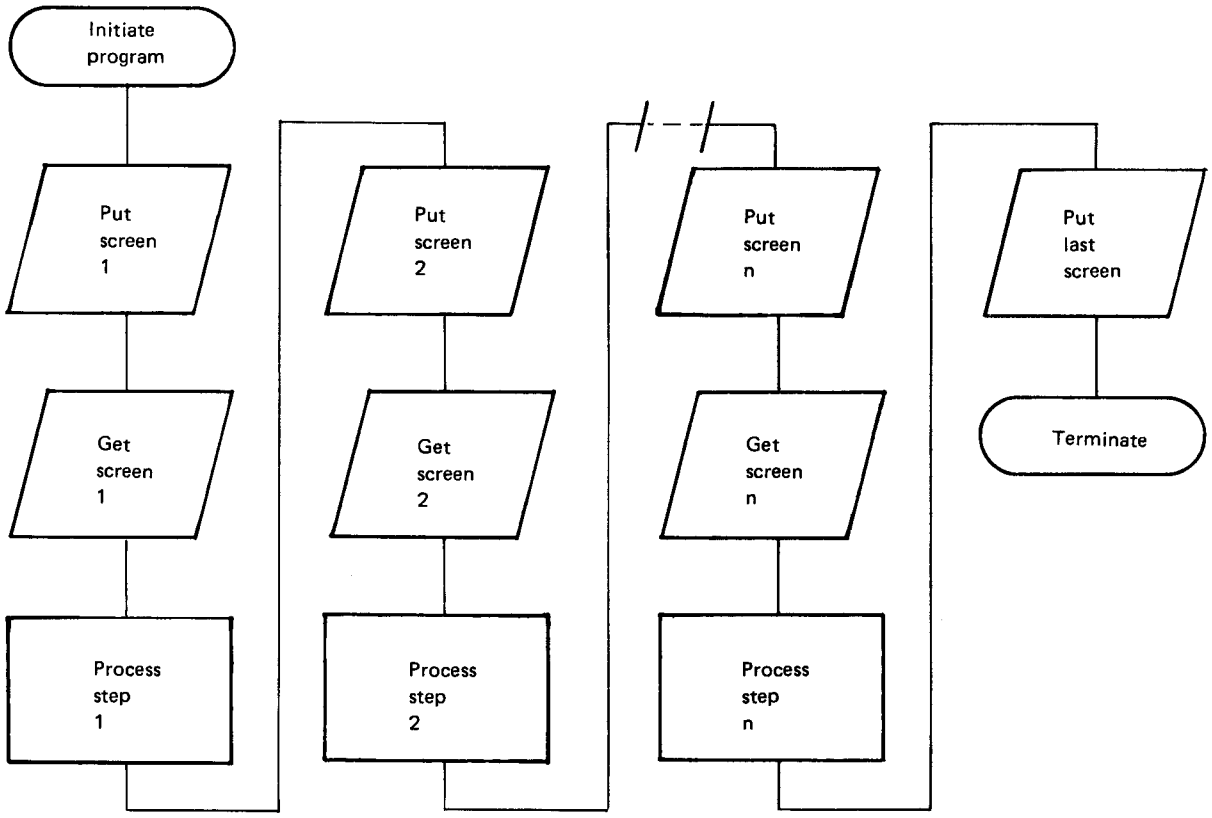


Figure 7. MDF/SRT Program Logic

Multiple Dependent Function MRT

The MDF/MRT is the most complicated structure to code. Not only must the program keep track of which step is next, but it must first identify the terminal and then determine what processing is next for that specific terminal. Whereas type MDF/SRT is more like batch programming, type MDF/MRT adds a level of complexity that may not be desirable in the System/3 environment.

An alternative to the MDF/MRT would be a combination of single function SRT and MRT programs. The availability of PRUF makes these types very useful for solving application design; however, using them involves some special considerations of interprogram communication.

Interprogram Communication

MDF/MRT program information can be passed between various processes within the program. With a combination of single function SRTs and MRTs, on the other hand, interprogram communication must be accomplished by writing and reading the information either to disk or to the terminal. Using the terminal to pass information between programs adds characters to a busy resource, the transmission line. For this reason, it is usually better to use the disk as a holding area for messages between different programs, since this facility is not usually as heavily utilized as the transmission line. However, this choice is not an absolute; if the installation has 3270 screens attached to the system via display adapter only, then the terminals provide an excellent holding area, since the display adapter's transmission rate is very fast.

The Model 15D (5704-SC2 only) portline function can support task-to-task communications. Thus, a program written to communicate with a program in another 15D (via the portline statement) can communicate with a program in the same 15D. Data is passed to the other program entirely within the CCP communications manager.

Also unique to 5704-SC2, task chaining provides an easy method of passing data between programs. Refer to Chapter 7 in this manual for additional information concerning task chaining with data.

Another consideration in choosing a method of passing data between single function programs is the overhead in the user task that is required. DFF program-appended storage and output hold area requirements add to the size of the user task if the terminal is used for interprogram communication. However, if the disk is used there is an overhead of additional disk I/O. The system designer should carefully evaluate the advantages and disadvantages of each technique in his particular circumstances before choosing a method of interprogram communication.

NEP (Never-Ending Program)

A never-ending program is a user application program which, once initiated, normally remains in memory until CCP is shut down. The never-ending attribute can be assigned to both SRT and MRT programs (PROGRAM assignment statement). The same general considerations described previously in this chapter for SRTs and MRTs also apply when these programs are NEPs.

On the System/3 Models 4, 8, 10, and 12, the main storage occupied by an NEP is unavailable to other tasks, even if the NEP terminates abnormally. On the Model 15, the main storage is available again when an NEP terminates.

There are several reasons for using an NEP:

- Program usage should be high enough so there is a real advantage to keeping a program in main storage to save program load time.

- The NEP may be a monitor program performing repetitive functions and may not communicate with a command terminal at all. Two basic examples are:
  - A program that monitors a disk print file, checking for documents to be sent to terminal printers. This use of an NEP is discussed in detail in Chapter 6.
  - A program that forwards messages to another system and may also receive responses, which it posts to files for later inquiry by terminal operators. This is a common approach in distributed systems where, for example, inventory may be allocated or ordered from a central site when the local site cannot fill an order. The advantage in this case is that an operator need not call the NEP but merely needs to post the request to a disk file that the NEP is monitoring. Task chaining (Model 15D) may also be used to communicate with the NEP.

SUMMARY

The primary consideration in choosing between the SRT and MRT approaches is the amount of time that a program will reside in main storage. If a program is used infrequently, the program probably has to be loaded when called and the MRT advantage is lost. At what point is the use of a program frequent enough to gain the MRT advantage? Simplified queuing theory (Chapter 10) can be used as an aid in this evaluation, but the final decision rests with the user.

Why not code every program as an MRT? In many cases, this would present no problem. However, if the program does extensive disk work between operator interactions, a response advantage is gained by having multiple copies of SRT programs running concurrently. This advantage is not possible with the MRT approach unless the user can have several copies of the same MRT program, each with a unique name. If the copies can be assigned by terminal, the effect of multiple copies can be achieved.

As more users are added to the system, the serial use of a program becomes more of a problem. If several users are queued waiting to be serviced by an MRT program, the last user must wait for the preceding terminals to be serviced; this wait time can become excessive.

MIF/MRT Technique

The following technique may be useful in some cases:

1. Write every program as an MIF/MRT. The functions are all similar and the terminal is released upon completion of any function.

2. On the PROGRAM assignment statement for each program, assign an MRTMAX value equal to the number of terminals.

Now, by assignment set manipulation, the program can be made to react in different ways. For example, with low utilization and no change to the assignment set, the program acts like an SRT. Should two requests occur at approximately the same time, the response time of the second is equal to the service time plus the wait time of the first to complete rather than the wait time to load another copy.

As the utilization of the program increases, the chances of finding the program in memory increase, thus improving response time whenever service time is less than program load time. At some level of utilization, the wait time for preceding terminals may exceed the wait time for load. If this level is reached, then the assignment set can be altered to remove the MRTMAX keyword from the PROGRAM statement, and the program reverts to an SRT with no change to the program itself.

There is one disadvantage to this approach. If an SRT releases a terminal rather than just terminating, additional load is placed on the CCP disk drive since transients must be used. Removing the code that releases the terminal improves the response time.

Chapter 3. Direct Files

Disk file organization can affect system throughput and terminal response time in an online environment. Since much of the file processing is random, the choice of file organization is usually between indexed and direct. Indexed organization offers a wide variety of processing methods and ease of programming; however, direct file organization provides some key advantages that can contribute to an efficient CCP system design:

- Fewer accesses to the disk can greatly improve response time.
- File recovery in the event of a system failure is easier than for indexed files.
- Sharing files between programs is simplified.

These advantages can be critical to the performance of the system, especially for files that are heavily used (many accesses to the file). However, this does not mean that sequential and indexed files should never be used in an online system. Sequential organization is useful for files that are not processed randomly, such as some logging files. Indexed organization may be satisfactory for master files if the file is not too active, if terminal response time is not critical, or if high performance disk drives (such as 3344 on the System/3 Model 15D) are available.

DIRECT FILE ADVANTAGES

To appreciate the advantages of direct files over indexed files for random processing in an online environment, it is necessary to understand how indexed and direct files are organized, loaded, and processed as described in the following publications:

- *IBM System/3 RPG II Disk File Processing Programmer's Guide*, GC21-7566
- *IBM System/3 Disk Concepts and Planning Guide*, GC21-7571
- *IBM System/3 Model 12 User's Guide*, GC21-5142

Direct file advantages are especially important for files that are accessed (read, updated, added to) frequently, require random processing, and must be accessed by more than one program concurrently.

Disk Accesses

Direct files may require fewer accesses to the file than indexed files for equivalent types of processing. This is important because disk access arm contention can be a major cause of poor response time.

For example, adding records to indexed files requires (1) scanning the index, including added index entries, to ensure the record does not already exist; (2) reading the data area where the new record will reside; (3) writing the record; (4) writing the new index entry. Adding a record to a direct file requires reading the record to verify one does not exist there already and writing the new record.

Depending on how a direct file is organized for *synonym records*, a direct file can require additional accesses. See *Access Algorithm and Synonyms* and *Examples* in this chapter for techniques of handling synonyms.

File Recovery

No file recovery is needed for direct files. If all files in a system are direct, the system can simply be restarted and processing can continue. Some method of determining the last successful update may be necessary. (For more discussion on file recovery procedures, see Chapter 9.)

File Sharing

File sharing between programs is simplified when direct files are used. Only two direct file access methods, direct input (DG) and direct update (DU), are required for processing direct files. Programs that process a file in these ways are not restricted in sharing the file.

ACCESS ALGORITHM AND SYNONYMS

The key to implementing a direct file is defining an access algorithm and synonym handling technique that satisfies the processing requirements for the file while preserving the advantages of direct files.

Determining an Access Algorithm

An access *algorithm* is whatever fixed (programmed) method is used to dictate the position to be occupied by each record. The algorithm can be simple or complex. In any case, the algorithm must yield a positive, whole number as a relative record number.

In the simplest case, the input to the program (the *control field*) is used directly as the relative record number. For example, loan number 3456 could be used without change as relative record number 3456. Another example of a direct technique is using direct files to store large arrays of data. If element X(10) is desired, then the tenth record in the file X is read. A control field should be used directly as a relative record only if there is not an excessive number of unused values within the range of values for the control field. If there are too many unused values (and, therefore, unused record positions), an algorithm should be defined to reduce the size of the file.

A formula can be used as an algorithm to determine the record number. For example, if loan numbers start with 1001, then loan number 3456 will be relative record number 2456 (3456 minus 1000). The formula can be as complex as you need to make it. See *Examples*, in this chapter, for more information and examples.

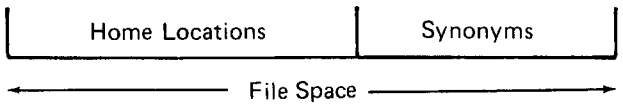
The control field containing alphameric data could also be used as a basis for an access algorithm. The algorithm must convert the alphameric data to a relative record number. See *Handling Synonym Records* for an example of using a customer name as the control field.

The choice of an access algorithm and, ultimately, the decision whether or not to use a direct file is usually based on how well synonym records can be handled. A *synonym record* is a record in a direct file whose control field yields the same relative record number as another record. If the handling of synonyms requires a significant number of additional disk accesses, one of the important direct file advantages is lost. Also, because access algorithm and synonym code must reside in each program that uses a direct file, a risk is involved: if the algorithm and synonym handling are ever revised, it may be necessary to rebuild files and modify all the programs that use those files.

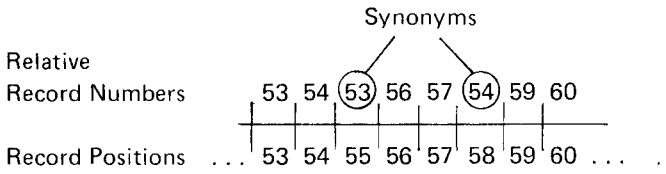
Handling Synonym Records

Synonyms can be handled in many ways. Some of the common ways are:

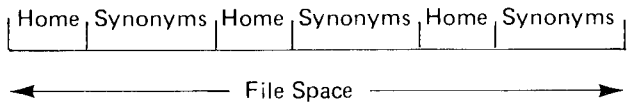
- Place synonyms in a separate part of the file, following the home locations, the area used for home records. A *home record* is a record that is stored in the location indicated by its relative record number.



- Place synonyms in the next available blank location, closest to the home location.

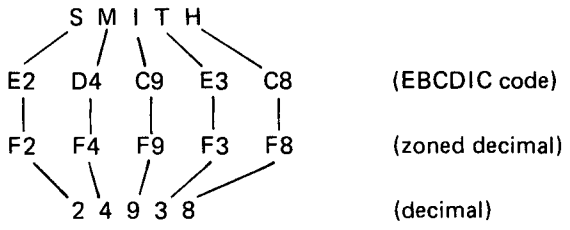


- Place synonyms in an area, next to the home location, that is reserved for synonyms.



In the first two methods, the record in the home location must contain a pointer (record location) to the synonym record. If there are two or more synonyms for a home location, then the first synonym contains a pointer to the second synonym, and so forth.

In the third method, synonyms are close to the home location. For example, assume the control field for a file is the first five characters of the customer's name. The file contains space for 40,000 records and allowance for three synonyms for each home record. The customer name is converted to a decimal value as follows:



The decimal value is then divided by 9999:

24938 ÷ 9999 = 2.4940

Ignoring the whole number of the quotient, the home location is calculated as follows:

(4940 x 4) + 1 = 19761

Since there may be many Smiths in the file, the program may have to read records 19761, 19762, 19763, and 19764 to find the correct Smith. If extra synonyms are required, the third synonym could point to the next available space in the file (possibly the next home location will not require all of its synonym locations). Another possibility, to reduce the number of synonyms, would be to accept six or more characters from the customer name.

EXAMPLES

The general steps in building a direct file are as follows:

1. Define an access algorithm.
2. Decide how to handle synonym records.
3. Evaluate the direct file, perhaps by using a test program.
4. Refine the algorithm and/or synonym handling.

The following examples illustrate direct file approaches to some online file requirements.

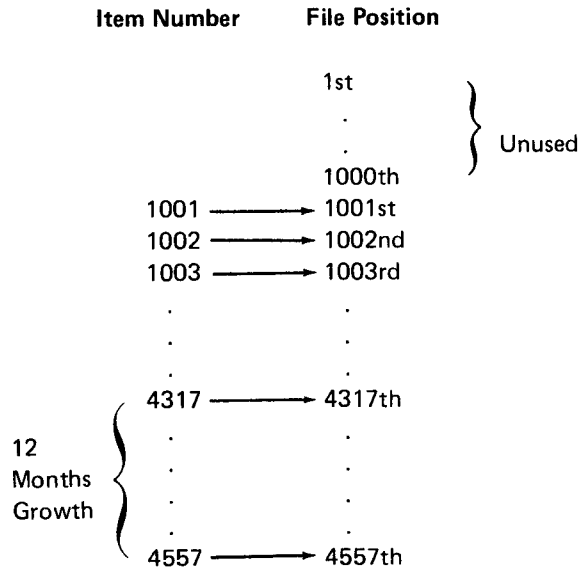
Example 1

In the example used here, the major goals are to build a file in which (1) the records can be accessed with an average of slightly more than one disk access, (2) the amount of disk space used for the file does not contain excessive unused space, and (3) there is room for growth within the file to easily accommodate new records.

Defining the Algorithm

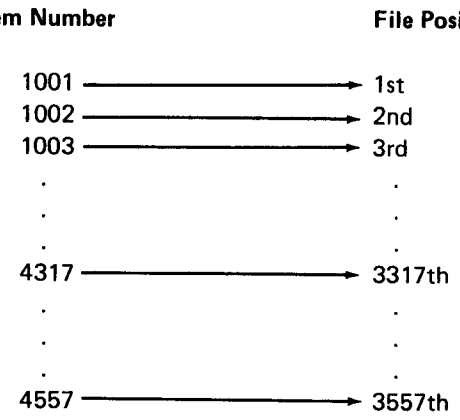
In this example, an item file, currently indexed, is to be converted to a direct file for an online order entry application. The key field is a five-digit item number–four digits assigned by the user and the fifth being a check digit. The four digits employed start with 1001, and the user merely assigns the next sequential number to new items. Deleted item numbers are not reused until number 9999 has been taken. Approximately 20 new items are added per month, while four items are dropped. Highest current number is 4317, but the file contains only 2,812 items. The 110-track allocation assumes 12 months growth.

As a first approach, the algorithm could be stated: The direct file position for each record shall be equal to the four-digit item number. Assume that the new record will be a few bytes larger than the old record, and that the file will also accommodate 12 months of growth before reorganization. The algorithm would require a file of 161 tracks containing 4,557 record positions. The mapping of items to direct file positions would appear as follows:



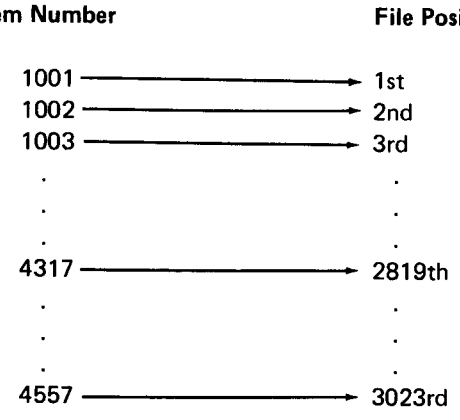
This first approach, while yielding no synonyms, uses only two-thirds of the record positions and most of the unused space is at the beginning of the file.

Assume the algorithm is revised to state: The direct file position for each record shall be equal to the four-digit item number minus 1000. The file requires 126 tracks containing 3,557 positions with the following mapping:



This approach, also yielding no synonyms, uses 85% of the record positions–the unused portion is embedded randomly within the file where items have been dropped. Although each record only requires one disk access, the file size still is 20% larger than the data portion of the former file. The algorithm can be further revised.

Now assume the algorithm states: The direct file position for each record shall be found by subtracting 1000 from the four-digit item number, multiplying the difference by 0.85, and half-adjusting the result. The file will occupy 107 tracks with 3,023 positions under the following mapping:



This approach uses 99% of the record positions and the file size is only 1% larger than the indexed data. It has, however, introduced the possibility of synonym records; item number 1004 (if it exists) will also be assigned to direct file record position number 3 (same as 1003). Similarly, item numbers 4316 and 4317 conflict, as do 4556 and 4557. Thus, the refinement of the algorithm to meet the second major goal (minimum file space) may now have impacted the first goal (minimum disk accesses) since synonym records will take a minimum of two accesses.

Handling Synonyms

Various methods of handling synonyms are described in *IBM System/3 Disk Concepts and Planning Guide*, GC21-7571, or *IBM System/3 Model 12 User's Guide*, GC21-5142. Whatever method is used, it must accomplish two overall goals: minimum accesses and minimum file space. The more immediate goal is to define (program) the manner in which a record will find an alternate position when its first location choice is filled.

Further analysis of the item file example might offer some suggestions for synonym handling. Note that a synonym can only occur about once in seven records.

The previous algorithm causes the following mapping (asterisks identify synonyms):

Item	Position	Item	Position	Item	Position
1001	1	1009	8	1017	14*
1002	2	1010	9*	1018	15
1003	3*	1011	9*	1019	16
1004	3*	1012	10	1020	17
1005	4	1013	11	1021	18
1006	5	1014	12	1022	19
1007	6	1015	13	1023	20*
1008	7	1016	14*	1024	20*

Recall that approximately one in seven item numbers is unused due to deleted items–the file is only 86% full. Thus, you might expect to find an unused position in the direct file with about the same frequency as the synonyms occur.

Assume the method of handling synonyms can be stated: A synonym record will be placed in the next higher numbered position that is unused. Since the file uses only 85% of the range of numbers, 15% of the numbers will not be used because they are deleted. However, the deleted numbers are randomly distributed through the entire range of numbers. Thus, there will be some positions available in the file for synonym records. (About every seventh number will be a synonym or 14%). Let's assume that of the first 40 item numbers, items 1007, 1008, 1015, 1017, 1020 and 1039 are among those deleted numbers.

Item	Position	Item	Position	Item	Position	Item	Position
1001	1	1013	11	1026	22	1036	31
1002	2	1014	12	1027	23	1037	**
1003	3	1016	14	1028	24	1038	32
1004	6	1018	15	1029	25	1040	34
1005	4	1019	16	1030	26		
1006	5	1021	18	1031	**		
1009	8	1022	19	1032	27		
1010	9	1023	20	1033	28		
1011	13	1024	33	1034	29		
1012	10	1025	21	1035	30		

Note the following:

- Item number 1031 will occupy some position numbered greater than 34.
- Item number 1037 will occupy a higher numbered position than will item number 1031.
- Record positions 7 and 17 are unused.
- After accessing a record, the program will have to verify that the record is the one that the program really wants; if it is not, the program must access a synonym.
- There will not be more than two items with the same relative record number; thus, most records require no more than two disk accesses.

  Note: This assumes that records are loaded into home locations before synonym records are loaded in a second run; this also assumes that there will be few added records. If records are added after the home records and synonyms are loaded, the home locations for the added records may be occupied by a synonym. Thus, the added record becomes a *pseudo synonym.* If many records are added, it is likely that most will have to be handled as synonyms. In this situation, the technique described here may be less useful, because performance tends to degrade as records are added.

In this synonym-handling technique the average synonym should be close to the first position searched. Thus, a second access is necessary approximately 15% of the time, and this access, hopefully, finds the record not too distant from the home location.

At this point, the file should be loaded (home positions only) and the synonyms added in a second pass. As the synonyms are added in the next available higher numbered position, a synonym pointer in the home record will have to be updated to point to the synonym record position.

Evaluating the Direct File

At this point, a program should be written to assess the file organization in light of the goals for the file. The program might measure factors such as:

- Number of accesses for the average record
- Which records require two accesses
- Distance between record locations for synonyms
- Where the unused positions are in the file
- Which records require three accesses (if any), four accesses, etc

The data reported by the program should be analyzed in terms of the user's requirements. It may not be sufficient just to report that records require 1.17 accesses on the average. Perhaps most of the synonyms are on one end of the file rather than randomly spaced. It could occur that the second access for synonyms was averaging 41 tracks, and half the synonyms were among the top 10% in file activity. Such a condition might necessitate putting all synonyms together in the middle of the file or in a separate file for faster access. Perhaps the most active records should be loaded first to guarantee that they will not be synonyms.

Refining the Algorithm and/or Synonym Handling

To further meet the needs of your system, refinements to the algorithm and/or synonym handling might be added. For example:

- Load items 1024, 1377, and 1844 in a preliminary pass so they will be found on the first access (because they are the more active items).
- Subtract 3024 from each file position and change the result to a positive number so the newer items will appear at the front of the file.
- Chain to another file on a separate drive for synonyms.

Example 2

Assume a customer master file contains three types of records (A, B, and C) for three types of customers. These records are in an indexed file where type A records have keys (customer numbers) from 10000 to 49999, type B are numbered from 60000 to 79999, and type C from 90000 to 99999. Each type of record is arranged alphabetically by customer name.

The file was first loaded with approximately 500 alphabetized type C records, followed by 1,000 alphabetized type B records, and finally about 3,000 alphabetized type A records.

Additions have been made at the end of the file in the following manner: first, the added record type is determined (A, B, or C); then it is assigned an unused customer number that corresponds to the alpha sequence of the customer name according to a listing of the file. When first loaded, the contents of the file were as follows:

Record #0001	Customer #90000	Type C (alphabetical by customer name)
Record #0002	Customer #90020	
Record #0003	Customer #90040	
.	.	
.	.	
Record #0467	Customer #60020	Type B (alphabetical by customer name)
Record #0468	Customer #60040	
Record #0469	Customer #60060	
.	.	
.	.	
Record #1592	Customer #10000	Type A (alphabetical by customer name)
Record #1593	Customer #10013	
Record #1594	Customer #10026	
Record #1595	Customer #10039	

The file originally contained 4,725 records (space was allowed for 6,000). Eighteen months later, the file contains 5,638 records.

An analysis of the file indicates the following:

- The file is experiencing about 12% annual growth and should probably be planned for about 6,600 records to meet one year's requirements.
- Customer numbers 10000-50000 are 8% used, while the other numbers are 5% used.
- Synonym records should be kept as close as possible to the home location.
- The best file design solution might be more than one file and more than one type of file organization.
- If all the customer numbers will reside in one file, an algorithm must take into account the necessity of loading the type C customers at the front of the file, followed by the types B and A.
- The ratio of A:B:C types is about 6:2:1.

A trial algorithm might try to accomplish the following mapping:

Customer Number	Type	File Record Number
90000-99999	C	0001-0733 (1/9 x 6600 = 733)
60000-79999	B	0734-2200 (2/9 x 6600 = 1467)
10000-49999	A	2201-6600 (6/9 x 6600 = 4400)

In order to accomplish the mapping, the algorithm must:

- Convert the customer numbers 90000 to 99999 into a set of relative record numbers from 1 to 733
- Convert the customer numbers 60000 to 79999 into a set of relative record numbers from 734 to 2200
- Convert the customer numbers 10000 to 49999 into a set of relative record numbers from 2201 to 6600

One method of doing these conversions is as follows:

- If the customer number is greater than 89999, subtract 89999 from it, then multiply the difference by .0733 (the ratio of 733 positions to 10,000 numbers), and use the half-adjusted product for the record position.
- If the customer number is less than 50000, subtract 9999 from it, then multiply the difference by 0.11 (the ratio of 4,400 record positions to 40,000 record numbers), add the half-adjusted product to 2200, and use the sum for the record position.
- For all other customer names (60000 to 79999), subtract 59999 from the number, multiply it by 0.0733 (the ratio of 1,467 record positions to 20,000 numbers), add the half-adjusted product to 733, and use the sum for the record position.

The synonym handling technique might be the same as in *Example 1*.

The test of success, as with *Example 1*, is to implement the algorithm/synonym handling technique by loading the file. Then the success can be measured by another program which attempts to retrieve all records and counts the number of accesses necessary. The results of the second program dictate whether modifications are necessary or desirable. To further test the file, a sample program can be run in an online environment to see whether response times at the terminals are acceptable.

Example 3

Other master files might have altogether different uses and for that reason use different techniques. Consider a rate file in a telephone revenue accounting application wherein one record exists for every *from-to* location in the United States. A call made from number (507) 286-5688 to (518) 392-5536 would require the retrieval of a rate record from the master file that would have a key of 507286518392. How can such a number be equated to a relative record position on a direct file?

One algorithm might be to multiply the numbers 507286 and 518392 together and use the second, fourth, sixth, eighth, and tenth digits of the product as the relative record position. This technique might yield a random distribution across a file for approximately 100,000 records. Another algorithm might be to take the second, fourth, sixth, eighth, and tenth digits from the 12-digit key. Thus, the first algorithm might locate the rate record in relative position 69301 (262973004112), while the second might place the same record in position 02613. Some records, for a given billing location, would be far more active than the majority of the records. These very active records might be placed in a separate file which may or may not be direct.

The techniques described in the previous paragraph are randomizing techniques. Many randomizing techniques are employed by users of direct files. Regardless of which technique is used, the concept and approach should be well documented in each program that uses the technique.

TRANSACTION FILES AS DIRECT FILES

Typically, transaction files (order records, deleted invoices, purchase records, and other transaction records) are the first in an online environment to become direct files. The reason for this is that the operator usually must page back and forth through these files, add new records, delete old records, review all or part of a file due to an inquiry and rely on very fast response time at the terminal.

Another reason for using direct organization for transaction files is that they can be updated easily by multiple programs (a file that is processed as consecutive output cannot be shared). Another program can access the data and do further processing on the data just as soon as the data is put into the file by an update program. Communication between the programs can be accomplished by the update program inserting a special character in each record that contains valid data, and the following program deleting that character as it processes that data. Thus two or more programs can loop through a direct file, filling it with data and processing the data, because the updated information is available immediately to other programs. For more discussion on the use and design of transaction files, see Chapter 9, *System Security/Integrity*.

MASTER FILES AS DIRECT FILES

The master files in online operations are also often becoming direct files. In many instances, the user may be converting from a batch system and already have indexed master files. Changing these files from indexed to direct could cause problems for the existing batch applications that must also continue to use them. The following paragraphs describe some compromise solutions to satisfy the requirements of both the batch and the online applications.

A master file that has served a batch design for a period of time may have more information in it than is needed for an online application. A new file could be created to contain a subset of the information in the batch master file; only those fields that are needed by the online application would be retained. There may be other deficiencies in the batch master file that can also be corrected when a separate file is created for the online applications. In addition to eliminating the unneeded fields, the designer may want to add new fields that are needed. In effect, a specialized master file is created to serve very specific design needs--no extra fields, smaller file, faster retrieval of information, and faster response times at the terminals. There are some potential problems with file integrity, since the same information is now in two or more different files. There may also be some additional file maintenance.

Can the benefits of this approach possibly outweigh the disadvantages? The major consideration is whether or not the online version of the master file contains fields that will be updated and thus differ from the true master file values. With careful planning and design, the technique described in the following paragraph can solve this problem.

Once a day, run a program that passes through the true master file and creates the online version. Process against the online version during the online run as if it is the true master file, updating fields as required, or, perhaps, creating a separate transaction file of updated records. At the end of the online execution, or concurrent with it, run a batch program against the online master file or the separate transaction file to update the indexed master file. At the end of the CCP run there should be agreement between the indexed master and the online master files, or reconciliation between these files and the transaction records created during the online execution. Further attempts to reconcile field values among the files might be done through a transaction log file.

Chapter 4. File Sharing

Programs that are executing concurrently under CCP can share the use of files. With some restrictions, files can even be shared among programs that are running concurrently in different partitions of the System/3 Model 15D. The system designer should give careful attention to file sharing, because it can have a significant effect on system throughput and terminal response time. It is especially important to consider the effects of file sharing in a transaction-oriented processing environment, because multiple copies of the same SRT program may be running concurrently using the same files.

The purpose of this chapter is to describe how file sharing affects CCP performance and to provide some system design recommendations. The reader should refer to the *CCP Programmer's Reference Manual* for specific file sharing guidelines and restrictions on various System/3 models.

FILE UPDATE CONFLICT

When two programs are updating the same file and the first program accesses a particular sector or block of sectors of that file, CCP prevents other programs from updating that data until the first program completes the update. This is known as *sector protect*. The conflict caused by this situation can result in a noticeable delay in the execution of the second program; this delay might show up as a degradation in response time at a terminal. If the application is designed so that multiple terminals call SRT programs that may update the same records of a file, this conflict situation must be considered.

The seriousness of file update conflicts can be greatly reduced through proper design. The likelihood of conflict can be reduced by using a minimum block size. This reduces the probability that two programs will try to access the same block of sectors at the same time.

Another way to reduce the likelihood of conflict is to avoid using control records that are updated frequently by different programs. Contention for access to a control record becomes particularly noticeable if your programs do considerable processing between reading and updating the control record. Programs that update control records should be designed to read the record, do minimum edit-type processing, update the record, and then perform any extensive processing that is required.

CCP/DISK SORT FILES

CCP/Disk Sort files should not be shared. The sort work files and output files cannot be shared, and the input file should not be shared with a program that adds or updates the file while the sort program is in process. In an online order entry transaction processing application, where records entered into a transaction file are to be sorted for each order, the records should be written from the transaction file to a nonshared sort input file (see Chapter 8, *Sort Under CCP*). If the transaction file is used as the sort input file, the terminals should not be allowed to add to or update the transaction file while the sort program is in operation. In most cases, this would be an unacceptably long lockout time for the transaction file.

ANALYZING FILE SHARING CONFLICTS

File sharing considerations are quite complex. Perhaps the best way to start analyzing potential conflicts in applications is to make a chart listing all programs (including maximum possible number of copies of SRTs) across the top of the chart and all files down the side. The files used by each program should be identified on the chart. In this way, the files that are used by many programs are identified and can be concentrated on for file sharing considerations. Possible serious conflicts can be identified by answering questions such as:

- Which programs must be operating at the same time?
- Can multiple copies of the file be used?
- Will different programs access the same records at the same time?
- Do some programs require that the file not be shared?
- Is response time a consideration in programs using this file?
- Does application logic require that the file not be modified while the program is executing?
- Can a program using this file be run in batch mode?

If file sharing is not really needed for certain programs, make sure that NOSHR is specified for those programs in the assignment set. Letting the parameter default to SHR will cause additional processing overhead, because each add or update record must be written out individually instead of being written out when the buffer is full.

By analyzing file sharing requirements carefully, the system designer can reduce or perhaps eliminate file sharing conflicts. With an understanding of what the potential conflicts are, the system designer can design applications to avoid the more likely trouble areas.

Chapter 5. 3270 Screen Design

In online applications using display terminals, the key objective is to service the terminal operator; therefore, the system must be designed to service the terminal. Good screen design can improve operator productivity and improve the performance of the system.

This chapter focuses on the screen design considerations related to using 3270 display terminals with CCP. The reader should consult *IBM 3270 Screen Design Student Text*, SR20-4441, for a comprehensive discussion of the human factors aspects of 3270 screen design, the various types of formats, relationships between application types and screen types, and other screen design considerations for the 3270 that are not specifically related to CCP.

The screen design guidelines presented in this chapter fall into two general categories:

- Human factors considerations: operator ease of use
- Performance considerations: efficient use of main storage, processing time, and transmission facilities for maximum throughput and satisfactory response time

These two categories are interrelated: operator ease of use affects system throughput and performance; a proper interval of response time from the system contributes to operator productivity and satisfaction.

HUMAN FACTORS CONSIDERATIONS AND TECHNIQUES

Before very many screen design decisions can be made, certain questions about the terminal operators must be answered:

- What type of operators will be used? Are they dedicated to this job and trained in the application, or are they occasional operators? What is their aptitude or skill level?
- How do the operators receive their input? By telephone? From a handwritten order form?
- What is the previous experience of the operators? Will they be willing to change their method of doing the job? What will be the turnover rate?
- What is the best response time for the operator? Response time should be neither too fast nor too slow. If the response time of the system is too slow, the program can give a preliminary response or acknowledgment to assure the operator that the system is working. If response time is too fast, an intentional delay can be built into the program to prevent harassing the operator.
- What kind of keyboard are operators experienced with?
- How will the system respond to operator errors?

General Guidelines

In general, input screens should be designed for ease of key entry; output screens should be designed for ease of reading. Keep in mind that the requirements of application-trained operators are different from the requirements of occasional operators.

Application-trained operators should require fewer prompts and headings and can make more use of short-form data and abbreviations. They also usually require shorter response time for maximum productivity. Occasional operators need more prompting and assistance, but can usually tolerate longer response times. Remember that operators who are proficient in entering data do not look at the screen often and may require the audible alarm to alert them to errors.

The following paragraphs should serve as a guide to the screen designer.

Display a Small Amount at One Time

The screen should be kept uncluttered and include only meaningful information. For example, do not display the entire record on an inquiry if the normal use is to look at one or two fields. Instead, display only the necessary fields and, perhaps, provide a function that allows the operator to display the entire record when it is required.

The 1,920-character screen is not intended only for displaying more data; part of the advantage of large screens is that they allow more flexibility for designing readable screens. In some applications, it may be desirable to skip lines between each line of displayed material and to limit the lines to 40 characters. However, in applications where the user has large volumes to display on a screen, these rules would be unacceptable. For example, when providing order entry screens for drug applications, one finds an item such as toothpaste results in a large screen because of the many different brands and sizes. In this kind of application, limited line length and skipping of lines would likely not be used.

Certain applications result in screens where data is accumulated from screen to screen. Up to a point this may be desirable; however, if the designer is not careful, the screen gets too cluttered and eventually contains useless information. The screen designer may wish to use nondisplay fields for information that is not needed by the operator. (The designer should be aware that transmitting several screens can cause excessive transmission time in a remote line environment.)

Clarity of Format

Clarity of format can be partly achieved by displaying a small amount of data at one time to prevent a cluttered screen. Format clarity can be improved by organizing information in columns, avoiding unnecessary indentations (left justifying), or eliminating unnecessary punctuation.

The screen text must be clear to the operator. There should be no questions as to the information desired, yet the text should not be wordy. The designer should avoid difficult words, symbols, abbreviations, and contractions. If special codes or abbreviations are necessary, the designer should include a means of explaining them to the operator; for example, a program function key could be used to request a help screen.

One Idea Per Display

Whenever possible, screen formats should contain information concerning only a single aspect of an application. For example, a format should not be used to query a file and perform an update at the same time. By concentrating on a single idea at a time, there is less chance for error, faster entry of data, and easier screen maintenance.

Shorter Operator Responses

Whenever possible, keep operator responses short. These responses can include codes, mnemonics, or abbreviations as long as the operator is trained to use them. Cursor positioning by the operator should be kept to a minimum. Instead, autoskip should be used, formats should be designed that do not require the operator to space over unused fields, and the cursor should be positioned under program control.

Always Acknowledge Operator Input

Operator interaction with a display screen is always conversational; therefore, the designer must always be concerned about how long an operator waits for a response. If the response will be slow in reaching the operator, the program should immediately acknowledge the receipt of the input. Always try to avoid a situation where the operator is sitting at the terminal wondering whether the input was ever received.

A good technique to use with large-volume output screens to reduce operator wait time is to transmit the screen in small blocks. This does not shorten the overall response time, but it reduces the time the operator has to wait before output begins to appear. In order to transmit a screen in blocks, the application program can issue an acquire terminal operation specifying an attribute set that has a block length of 512 with DFF. A disadvantage of this technique is that the screen blinks as each block of data is received by the terminal. If blinking is excessive, it may become irritating to the operator.

Standards Between Screens

Every application has its own particular requirements, but it is good practice to maintain certain standards across applications, such as similar screen formats and similar dialogue, abbreviations, codes, and mnemonics. This is particularly important when the same operator is performing more than one application. It is a poor procedure to be constantly switching the operator between different types of dialogue and screen formats. The standards established within an application are even more important than those between applications.

Use of PA and PF keys should be standardized between formats and applications. For example, avoid having PF1 perform a specified function for one screen and PF2 perform the same function for another screen. Of course, it may be necessary to use a key in an application-unique function, but a legend should then be used to tell the operator about the nonstandard use of the key.

Ease of Correction

The operator must be instructed in use of the ERASE EOF key and ERASE INPUT key to correct input before entry to the system. A procedure should also be programmed whereby the operator can request a cleared screen (that is, a format with no data in the variable fields) or a menu screen.

Programmed procedures must also be established for ease of correction after the data has been entered, such as:

- Use high intensity for fields that are in error.
- Sound the audible alarm.
- Place the cursor at the field that is in error.
- Allow retrieval of previously entered data for correction–especially important for data collection applications.

Clarity of Instructions to Operator

The messages to the operator must be clearly recognizable on the screen and be short but understandable. The instructions can be displayed in high intensity and set apart from the rest of the information on the screen to attract the operator's attention. The audible alarm is useful to alert the operator to a message.

The location of messages on the screen should be standardized across all applications. Frequently, operator messages are placed at the bottom of the screen. They can be set off from the other information on the screen with special characters such as asterisks or minus signs; however, this adds to the amount of transmission required.

Provide Means for Help

Not only must the operator be told of problems, but he or she should know what actions to take. These actions must be included in the written operator's procedures for the application. It may be desirable to program certain routines that assist the operator when help is required. This help can be obtained by entering special commands in a specified field on the screen or pressing a predefined PF or PA key.

Make the Operator Feel Comfortable

Most of the preceding guidelines can be summed up with this one guideline. The operator is using the terminal as a tool to do a job. As with any tool, the terminal should be easy to use, make the job easier for the user, and enable the user to do a better job. At the same time, application programming and screen design should take advantage of the operator's intelligence and make the operator feel important; they should not bore, scare, or harass the operator. Be sure to employ as many features of the terminal as necessary to assure the best possible job.

Whenever possible, give the well-trained operator the opportunity to initiate shortcuts in going from one screen to another. For example, do not always force the operator back to a menu screen to select the next screen to be displayed. PF keys can be used to implement this facility.

Specific Suggestions by Application Type

Some specific screen design suggestions for inquiry, file update, and data entry types of applications are given here. Relationships between application types and screen types are described in the *IBM 3270 Screen Design Student Text*. The reader is urged to refer to that publication for information on those relationships.

Inquiry Applications: The following suggestions apply to inquiry applications:

- Show only the data that is normally sought; provide the option to the operator to look at more data if necessary.
- Make limited use of headings on output screens. Arrange fields so that the information returned stands out without numerous headings.
- Avoid extraneous information on output screens, such as asterisks that are used to outline information. Avoiding unnecessary information improves the readability of the screen, reduces the size of the format that must be stored on disk, and reduces the amount of data transmitted.

File Update Applications: The following suggestions apply to file update applications:

- If the file is being updated only (no records are being added) the screen should contain only the item being updated.
- If records are being added to the file, it may be useful to retain previous additions on the screen.
- The operator should always enter data on the same line, such as the bottom or a middle line. If error conditions occur, the item in error is redisplayed with high intensity and the audible alarm.

Order Entry and Data Entry Applications: The following suggestions apply to order entry and data entry applications:

- The screen format should be compatible with the input document used by the operator. The sequence of fields on the input document should be the same as the sequence on the screen. Either the screen should be designed to conform to the document, or the document should be designed to conform to the screen format.
- Errors on the screen should be highlighted in some way for the operator, such as by using the audible alarm and high intensity.
- Fields on input screens should be laid out so that the most frequently used fields are entered first, followed by less frequently used fields. In this way, operator productivity can be improved because the operator will not have to tab across infrequently used fields.
- If operators will enter data based on verbal (telephone) conversation, design the screen in the sequence the operator usually directs the conversation.
- In order entry applications, either the whole order is kept on the screen as it is entered, or only the previous entry is displayed to enable the operator to keep his or her place. If only the previous entry is displayed, a separate program (initiated by a function key) should be available to display the entire order for review. In data entry type applications such as entering data into a file, one line of data should be on the screen at a time to reduce the amount of data transmitted and improve operator efficiency.
- Be aware, when transferring an operator keying job from a batch keying device (such as 3741) to an online keying device (such as 3277), that the online operator will *always* take longer to do the job due to response time.

Example: Three Approaches to Screen Design for File Update

This example compares three different methods of screen design for a file update application. Comparisons are made from the following viewpoints:

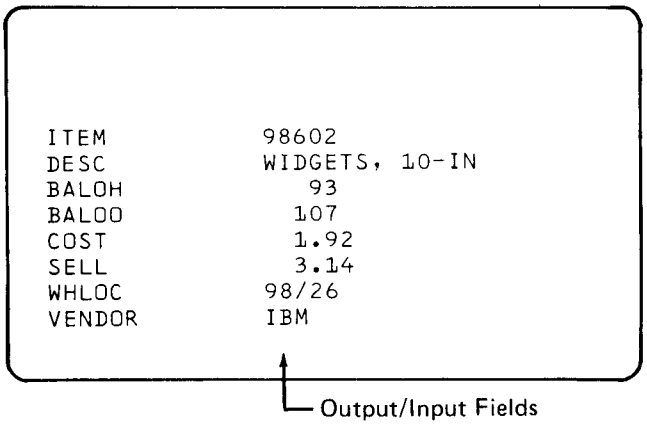
- Operator keystrokes
- Number of characters transmitted
- Amount of editing and updating that the application program must do

Assume the following data fields:

Field	Length
Item number	5
Description	25
Balance on hand	5
Balance on order	5
Cost	6
Selling price	6
Warehouse location	6
Vendor (supplier)	10

First Method

In this method, all data fields are output/input fields. The terminal operator moves the cursor to the beginning of the field that is to be changed and enters the new data over the old. The cursor is initially positioned at the *balance on hand* field, because the item number and description are rarely changed.



Keystrokes: To change a field, the operator must:

- Tab (except to change balance on hand)
- Key data
- Press ERASE EOF
- Press ENTER

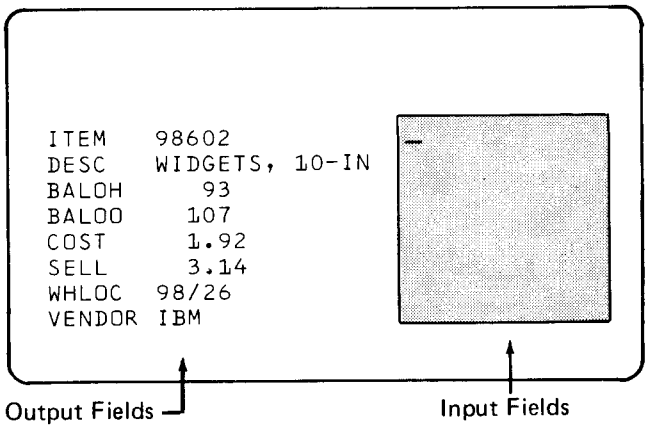
For example, 10 keystrokes are required to change the cost field (two tabs, six characters, ERASE EOF, and ENTER).

Characters Transmitted: All output/input fields are returned to the program. Therefore, 92 characters are transmitted (68 total data characters for the eight fields plus three control characters per field).

Editing and Updating: All fields must be edited and updated.

Second Method

In this method, the data fields are output fields and updates are keyed into input fields. The terminal operator moves the cursor to the beginning of the appropriate input field and enters the change.



Keystrokes: To change a field, the operator must:

- Tab
- Key data
- Press ENTER

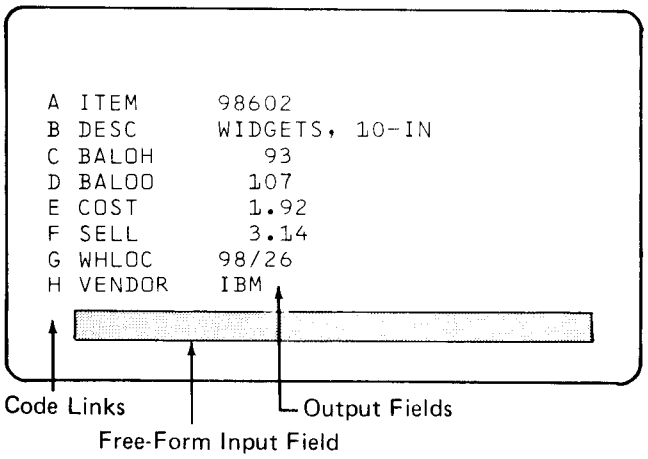
For example, nine keystrokes are required to change the cost field (two tabs, six characters, and ENTER). ERASE EOF is not required because data is being entered into input fields.

Characters Transmitted: Only the changed fields are returned to the program. For example, nine characters are transmitted if the cost field is updated (six data characters and three control characters); 16 characters are transmitted if balance on hand and balance on order fields are updated.

Editing and Updating: Only the fields that are actually changed must be edited and updated.

Third Method

In this method, the data fields are output fields and changes are entered into one free-form input field. The cursor is initially positioned at the input field.



Keystrokes: To change a field, the operator must:

- Key the code link (followed by a slash)
- Key changed data
- Press ENTER

For example, nine keystrokes are required to change the cost field (code link, slash, six characters, and ENTER).

Characters Transmitted: Only one field is returned to the program. For example, 11 characters are transmitted if the cost field is updated (code link, slash, six data characters, and three control characters). If more than one field must be changed, a separate transmission and separate disk access are required for each.

Editing and Updating: Only the field that is changed needs editing and updating.

PERFORMANCE CONSIDERATIONS AND TECHNIQUES

The screen design techniques described in this section can help to improve the performance of the system in one or more of the following ways:

- Reduce the overall main storage requirement of CCP user tasks
- Reduce the amount of data transmitted
- Reduce operator time, errors, and keystrokes
- Make full use of available CCP facilities

Display Format Facility (DFF)

Despite the additional main storage requirement in CCP for DFF, DFF should be used to design and generate display formats. The benefits of DFF, ease of format description and ease of programming, usually outweigh the costs in terms of additional CCP overhead. DFF offers the additional benefit of allowing formats to be tested using the display format test routine prior to using them in an application.

Program Request Under Format (PRUF)

One of the primary advantages of PRUF is that a program does not have to be in main storage during a lengthy operator keying operation. The following are some screen design considerations when using PRUF:

- If the terminal receiving the PRUF format is also the terminal that requested the program issuing the PRUF format, ENDMSG-NO should be specified in the PROGRAM assignment statement for the program. This prevents positions 82 through 160 of the screen from being cleared by an ending message when the program terminates.

- Always keep positions 82 through 160 of the screen open to receive system messages. This is required because messages from the system, such as a system operator shutdown request, can be sent to the terminal while the terminal is not under user program control (while the operator is keying into the screen). In order to keep positions 82 through 160 open, do not define any fields for those positions at display format generation time.

- In some cases, the screen provides a convenient buffer for passing control information and other data from program to program in a PRUF string. This technique can eliminate some disk file accesses for temporary transitional data storage. There is a hazard in passing information in nondisplay form, however, since the operator can unknowingly destroy the data by pressing the CLEAR key. If the operator should not see the information, it is better to keep the data in a disk file. In no case should the operator have to repeat previous steps in the application and reenter the input because of pressing the CLEAR key.

Headings and Prompts

Keep the wording in headings and prompts to a minimum. Use common abbreviations; for example, use QTY instead of QUANTITY. Also, use abbreviations unique to the applications; for example, use ORD for ORDERED QUANTITY and SHPD for SHIPPED QUANTITY. In some instances, headings can be omitted; for example, SOLD TO NAME & ADDRESS. Standardize headings and prompts among formats.

If a series of screens with common headings is used, consider transmitting the headings as a separate screen format. This reduces the total amount of transmission, since headings need only be transmitted once. Reducing transmission time is especially important for remote terminals, where transmission time is more critical than for local terminals that are attached via the display adapter. Headings can be transmitted separately by means of the overlay screen technique. Subsequent screens must not erase the initial heading screen.

Attribute Characters

Minimize the number of attribute characters on a screen. Each attribute written to the screen requires five transmission characters (two characters if the attribute is being written to the present screen position). Attribute characters are generated by the display format generator routine, DFGR, based on the display format specification.

To reduce the number of attributes on a screen, heading lines should be used with the data entered below the headings. In this way, only one attribute is used for heading line. Avoid the sequence: heading/data/heading/data on the same row, since each heading requires its own attribute.

Field Descriptor Table (FDT)

The FDT, as created by the format generator, must be available in main storage during the execution of a user task. The size of the largest FDT associated with a program is specified in the PROGRAM statement during the assignment run. The size specified is rounded up to the next multiple of 256 bytes and that amount of storage is allocated at program load time (plus an additional amount, based on the number of terminals and number of formats). The smaller the FDT, the smaller is the main storage requirement for the task. (See Chapter 11, *Performance Tips* for further information about the FDT.)

The screen designer can reduce the size of the FDT in the following two ways:

- The number of fields defined in the screen format can be reduced by eliminating unnecessary fields, such as headings and prompts that are not essential, or by combining two or more fields into a single field. In order to know when it is advantageous to reduce the number of fields, the designer should be aware that 17 field description entries can fit into the first 256 bytes of the FDT and 18 entries can fit into each succeeding 256 bytes. Therefore, it is advantageous to reduce the number of fields defined if, by so doing, an entire 256 bytes of FDT can be eliminated. For example, if 37 fields are defined, three 256-byte increments of storage are required for the FDT. Eliminating two field definitions reduces the size required for the FDT by 256 bytes. Thirty-five fields (17 plus 18) will fit into two 256-byte increments of storage.

- Output fields can be defined with an F specified in column 16 (data source) of the DFF field definition form. If F is specified in this column, CCP does not build an FDT entry for a field. The F designation should be used when output data is defined at generation time and the field will not be used in put override operations.

Put Override

Appropriate use of the put override operation can improve system performance by reducing the amount of data transmitted and by reducing the number of disk accesses for formats. In many instances, especially in error situations, use of a put override operation can make retransmitting a full screen unnecessary. Some suggested uses for the put override operation are:

- Put error messages to the screen.

- Highlight fields in error.

- Position the cursor at the next field after an error.

- Change the status of an existing format, such as restoring the display to its original condition.

- Highlight error messages on the screen. In some applications, error messages can be kept on the screen in nondisplay status. In case of error, a put override operation can be used to change the attributes of the message to display it, perhaps using high intensity.

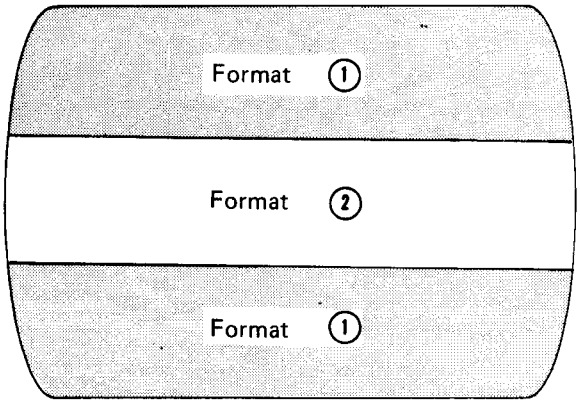
Overlay and Segmented Screens

Overlay screens can be used to reduce the amount of data transmitted. Overlay screens can be used when some data is to remain on the screen and a part of the data is to be replaced. Screens can also be segmented, with different parts used for the different steps in a process. Figure 8 illustrates the use of overlay and segmented screens.

If overlay or segmented screens are used in a PRUF environment, Program A might terminate leaving a screen that is composed of several different segments from separate formats. If Program B, the PRUF read program, is to receive the entire screen as input, Program B must use a single format that defines all the fields it needs, using identical field locations. The format used by Program B can have a different name from the formats used by Program A.

Overlay Screen

Program A transmits format (1) to the terminal and then repeatedly transmits and receives information using format (2).



Segmented Screen

Program A transmits three formats (segments), each from a different step in its processing.

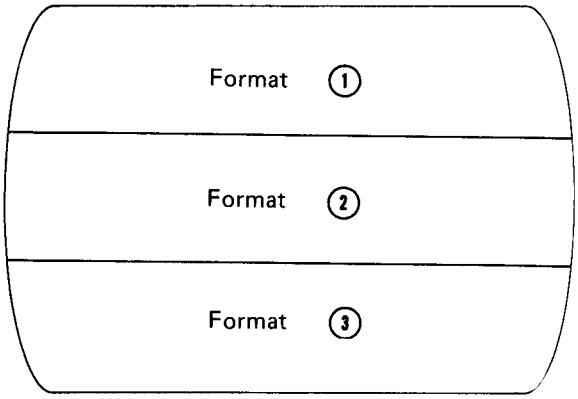


Figure 8. Overlay and Segmented Screens

Chapter 6. Use of Printers Under CCP

Using printers in programs that run under CCP requires special program design considerations. Two types of printers can be used in CCP programs: the system printer and terminal printers. The system printer is a line printer on System/3 Models 8, 10, 12, and 15; and a serial (matrix) printer (115 characters per second) on the System/3 Model 4. Terminal printers are from the 3270 family (3284, 3286, 3288) with maximum rated speeds of 40 cps, 66 cps, and 120 lines per minute, respectively.

SYSTEM PRINTER

Coding for a system printer under CCP is identical to coding for the printer in a batch program. However, the following should be noted:

- On System/3 Models 4, 8, 10, or 12, special print modules are available with the compilers. These are used when creating an object program to be run under CCP so that other tasks running under CCP are not degraded because a print program is being executed. These print modules also prevent an I/O attention from occurring if the printer is not ready. An I/O attention would stop the system. In order to have only one copy of the compiler, $MAINT can be used, as shown in Figure 9, to rename the print modules (and other unit record modules) before compiling a program. It may be convenient to call an appropriate procedure before compiling a program, as shown in Figure 9. (See *Compilation and Link Editing* in the *CCP Programmer's Reference Manual*, GC21-7579, for details.)
- Only one program may use the system printer at a time, with the following exceptions:
  - If spooling is used by the CCP program level on Model 12
  - If spooling is used by the CCP partition on Model 15
  - If PRINTER-SHR is specified in the PROGRAM assignment statement for Model 15 only, even if there is no spooling
- The system printer may be made available to the other programming level (Models 8, 10, and 12) through a system operator command.

Run $MAINT as follows after CCP generation (assume F1 is the RPG II compiler pack):

```
// LOAD $MAINT,F1
// RUN
// COPY FROM-F1,LIBRARY-R,NAME-$$ARFF,NEWNAME-$$XRFF,TO-F1,RETAIN-P (1442)
// COPY FROM-F1,LIBRARY-R,NAME-$$LPRT,NEWNAME-$$XPRT,TO-F1,RETAIN-P (Printer)
// COPY FROM-F1,LIBRARY-R,NAME-$$MF.ALL,NEWNAME-$$XF,TO-F1,RETAIN-P (MFCU)
// COPY FROM-F1,LIBRARY-R,NAME-$$CP.ALL,NEWNAME-$$XP,TO-F1,RETAIN-P (3741)
// END
```

Thereafter, to compile RPG programs for batch use:

```
// CALL RPGBCH,F1
// RUN
```

Procedure RPGBCH

```
// LOAD $MAINT,F1
// RUN
// COPY FROM-F1,LIBRARY-R,NAME-$$URFF,NEWNAME-$$ARFF,TO-F1,RETAIN-P
// COPY FROM-F1,LIBRARY-R,NAME-$$UPRT,NEWNAME-$$LPRT,TO-F1,RETAIN-P
// COPY FROM-F1,LIBRARY-R,NAME-$$UF.ALL,NEWNAME-$$MF,TO-F1,RETAIN-P
// COPY FROM-F1,LIBRARY-R,NAME-$$UPIP,NEWNAME-$$CPIP,TO-F1,RETAIN-P
// COPY FROM-F1,LIBRARY-R,NAME-$$UPOP,NEWNAME-$$CPOP,TO-F1,RETAIN-P
// END
```

To compile RPG programs for CCP use:

```
// CALL RPGCCP,F1
// RUN
```

Procedure RPGCCP

```
// LOAD $MAINT,F1
// RUN
// COPY FROM-F1,LIBRARY-R,NAME-$$XRFF,NEWNAME-$$ARFF,TO-F1,RETAIN-P
// COPY FROM-F1,LIBRARY-R,NAME-$$XPRT,NEWNAME-$$LPRT,TO-F1,RETAIN-P
// COPY FROM-F1,LIBRARY-R,NAME-$$XF.ALL,NEWNAME-$$MF,TO-F1,RETAIN-P
// COPY FROM-F1,LIBRARY-R,NAME-$$XPIP,NEWNAME-$$CPIP,TO-F1,RETAIN-P
// COPY FROM-F1,LIBRARY-R,NAME-$$XPOP,NEWNAME-$$CPOP,TO-F1,RETAIN-P
// END
```

Figure 9. Renaming Unit Record Data Management Modules for Compiling CCP or Batch Programs

Spooling Printed Output Under CCP

Output from print programs directed to the system printer may be spooled on System/3 Models 12 and 15. The advantage of this is that the system printer can be concurrently used by programs in the batch partition. There are some considerations:

- If only one task may use the printer at a time (PRINTER-YES), and the DEFER-NO parameter was specified in the OCL at startup, the spooling routines close the spool print file when the CCP termination routine is called for the task. This means that another program requesting the printer will be refused until the first task ends. This prevents mixing report lines from multiple tasks.

- If the printer is to be used as a logging device or for debugging purposes during testing, concurrent tasks are allowed to print if DEFER-NO is specified on the // PRINTER OCL statement and PRINTER-SHR is specified on the PROGRAM assignment statements. Print lines will appear in the order they were put to the spool file; the output will be mixed.

TERMINAL PRINTERS

Coding for terminal printers is more like coding for a 3277 display terminal than coding for a system printer. The terminal printers work as follows: A data stream is sent to the buffer of the printer. The data stream includes such control characters as NL (new line) and EM (end of message). When the user program gives the order to start printing by means of WCC (write control character), the buffer contents are printed. The program that sends the output to the printer can either remain in memory for further processing or terminate. The printing itself is an offline function.

The programmer can use PFGR (printer format generator routine) and DFF to format printed output, catalog the format in an object library, and access the format from an application program. (See *CCP Programmers Reference Manual* for details.)

The programmer can also use the terminal printers in a program without using DFF. However, all the control characters required by the 3270 system must be included in the data stream the application program builds.

Forms Design for Terminal Printers

Forms design for terminal printers is critical for good performance. Since printing is done one character at a time on serial (matrix) printers, the print heads must move to the appropriate position on a print line. This head movement takes time. Printing as far to the left as possible reduces head movement. In addition, because skipping to a new line is done with carriage returns, a large form with little printing is inefficient. For this reason, forms should be as short as possible.

Figure 10 is an example of an inefficient form design for a terminal printer. Figure 11 shows how the form could be designed for better performance.

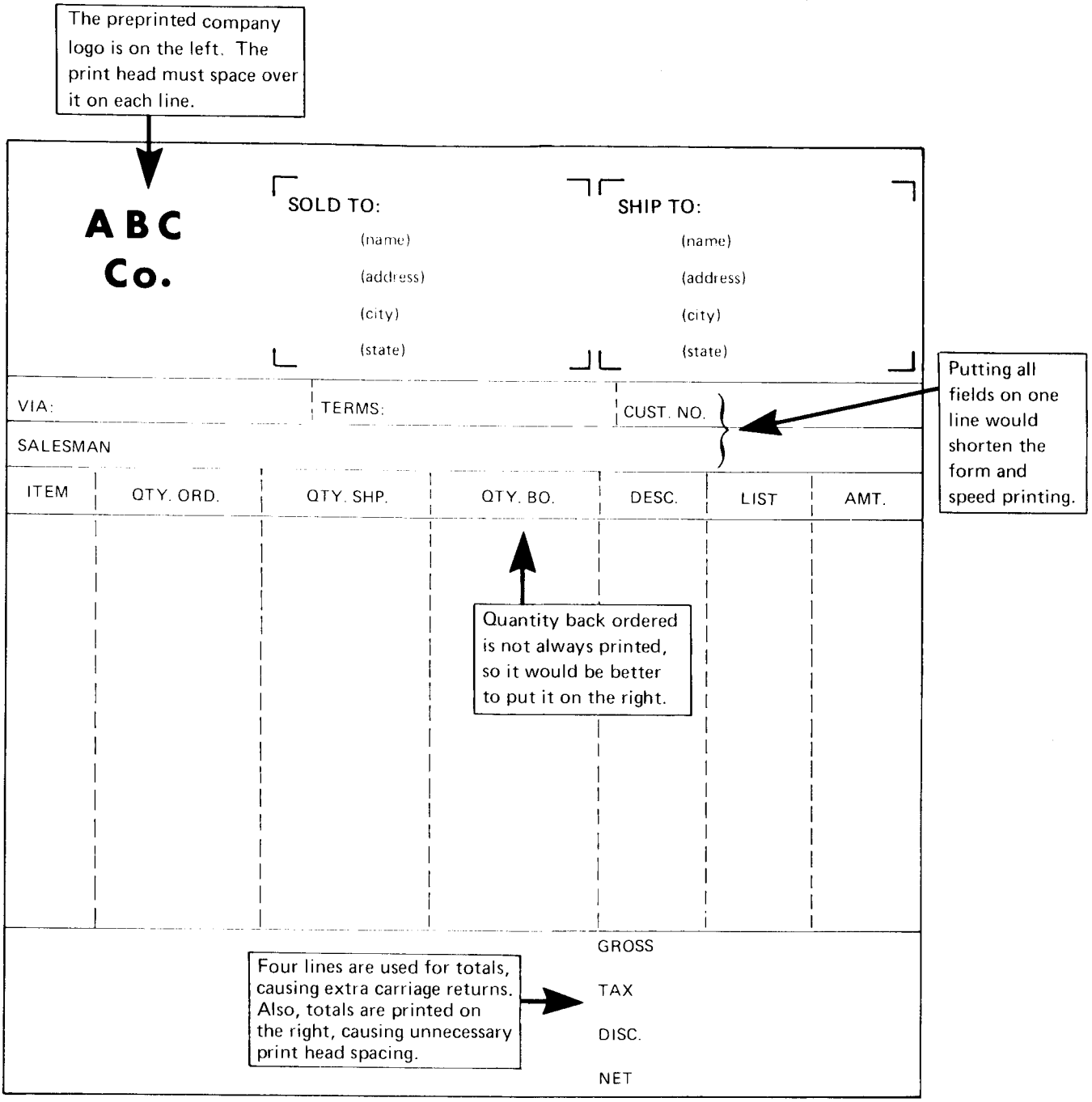


Figure 10. Example of Inefficient Form Design for a Serial Printer

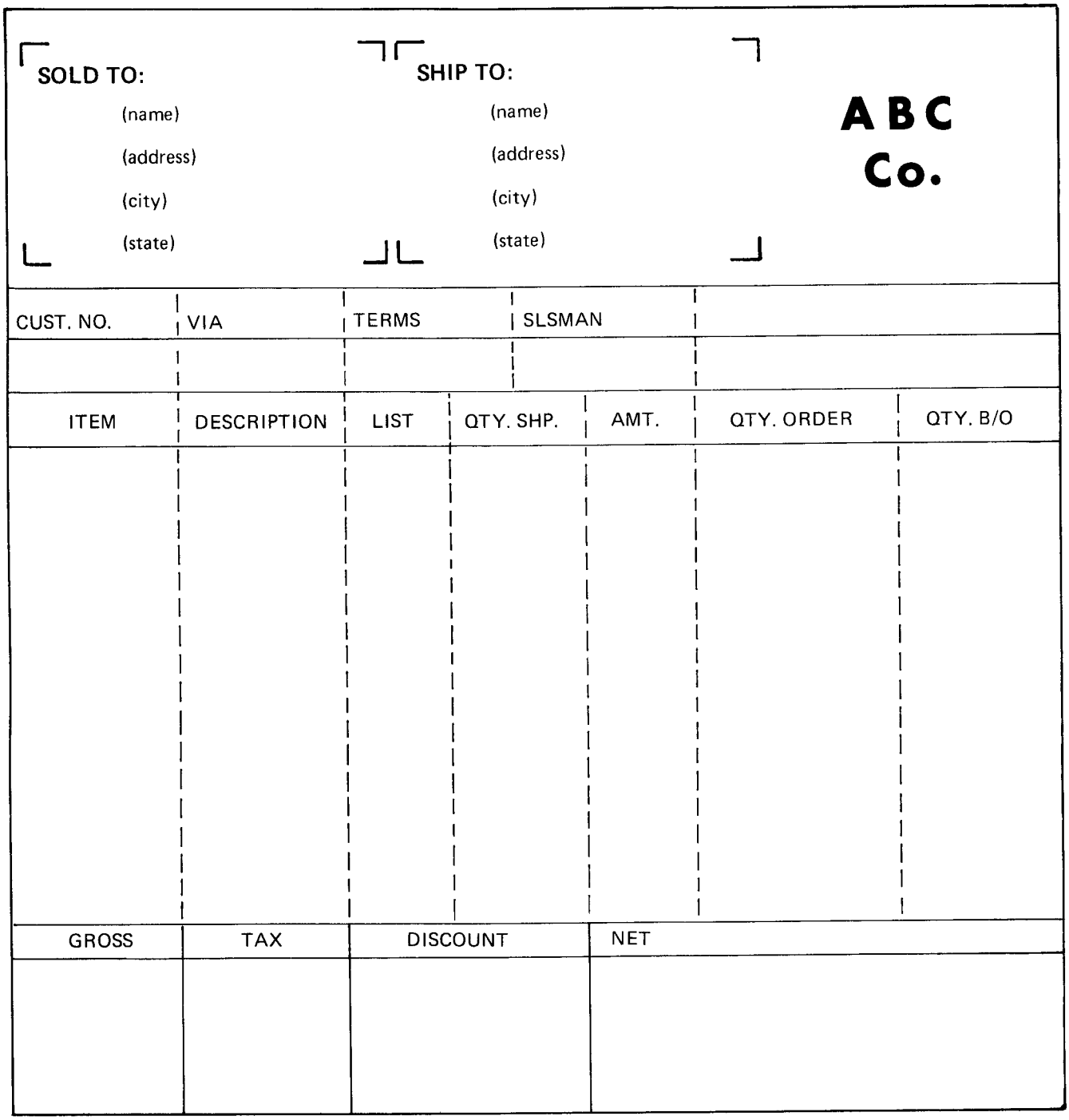


Figure 11. Example of Efficient Form Design for a Serial Printer

Program Design Techniques for Terminal Printers

Assume an application requires entering and editing data, validating by the operator, and printing a single document. Figure 12 shows the flow of the application.

Two key points are illustrated in this example. First, the terminal operator is free to do other work as soon as the print program (program C) has issued a release terminal operation. This means the operator can return to program A and overlap the next operation with the printing of the previous. Second, the number of buffer loads to be put to the printer determines how long the print program will be in main storage. If more than one buffer load will be sent to the printer, the following coding/design tip should be considered.

Assume that a form to be printed includes 3,000 characters of data, including control characters. Assume the buffer size is 1,920 positions. Obviously the entire form cannot be sent in one put operation. The second put must occur after the first buffer load is printed.

If 1,900 characters were sent first, then 1,100, the second put would be done after waiting for all the data within the 1,900 to be printed. This would not be efficient. The first put should be as short as possible, perhaps 1,100 characters. After this is printed, the second put of 1,900 can be done and the program can immediately terminate. The 1,900 characters are printed offline without using processing unit time.

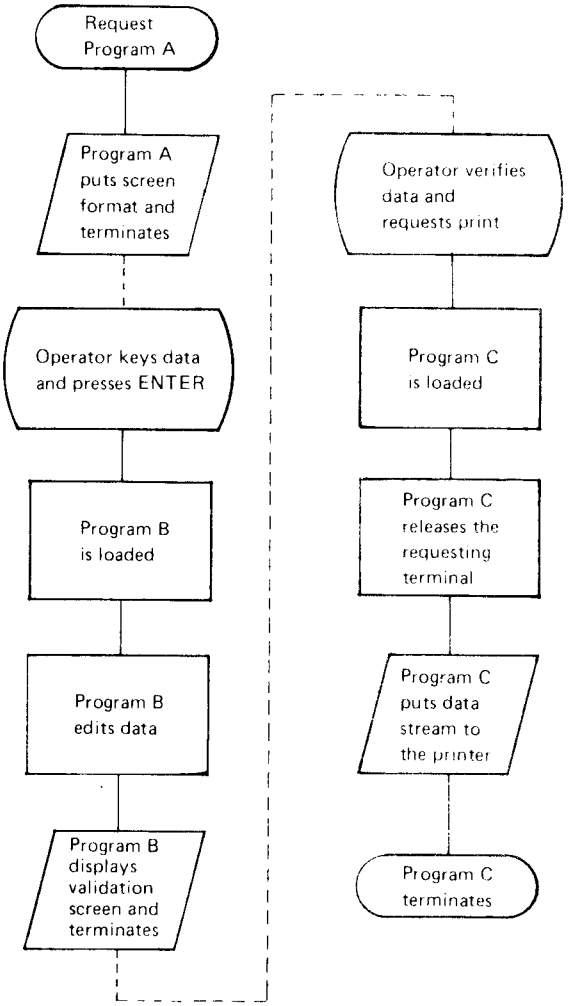


Figure 12. Printing a Single Form

Printer Busy Condition

Figure 13 shows the logic of a program that is using a terminal printer.

If the program requests a print operation, but a previous print operation is not completed, CCP informs the program that the printer is busy via a return code. It is the user's responsibility to try the operation again. If the data sent to the buffer on the first put is being printed and the program has a second buffer to send, the program loops between trying to put the second buffer, getting a printer busy return code, and trying to put the second buffer again. This prevents CCP from regaining control and allowing another concurrent task to do work.

There are several methods of preventing CCP from being shut out:

- Design the application so that forms are printed one at a time and are short enough to fit into the buffer. This is ideal but obviously for some applications cannot be done.
- Use terminal printers for short reports and print long reports or documents on the system printer.
- On Models 4, 8, 10, and 12, when generating a CCP system, specify BSYPRT-YES in the $EBSC macro. This enables CCP to detect the busy printer condition. After the printer completes the print operation, your program regains control. This option enables your program to execute properly without testing for the busy printer return code, and it allows other tasks within the system to use system resources while the matrix printer is busy.
- On Model 15D CCP (5704-SC2), when generating a CCP system, specify BSYPRT in the $EFAC statement. This includes the same support as explained above for Models 4, 8, 10, and 12.
- On Model 15 CCP, use the WAIT operation code. This facility, together with the interval timer (hardware feature), allows an application program to issue a wait of some seconds after determining the printer is busy. This causes CCP to regain control, service other tasks, and return to the print program after the specified time interval.
- On Models 8, 10, or 12, when BSYPRT-YES has not been specified in the $EBSC macro, the user program should contain code that forces a CCP transient load so that CCP regains control. One way to do this is to assign a dummy terminal name to a terminal in the assignment set. After obtaining a return code indicating the printer is busy, an acquire terminal operation can be issued to the dummy terminal with new attributes. When operation fails, CCP takes control and allows another task to obtain service. When CCP returns control, the print program can retry the printer operation.

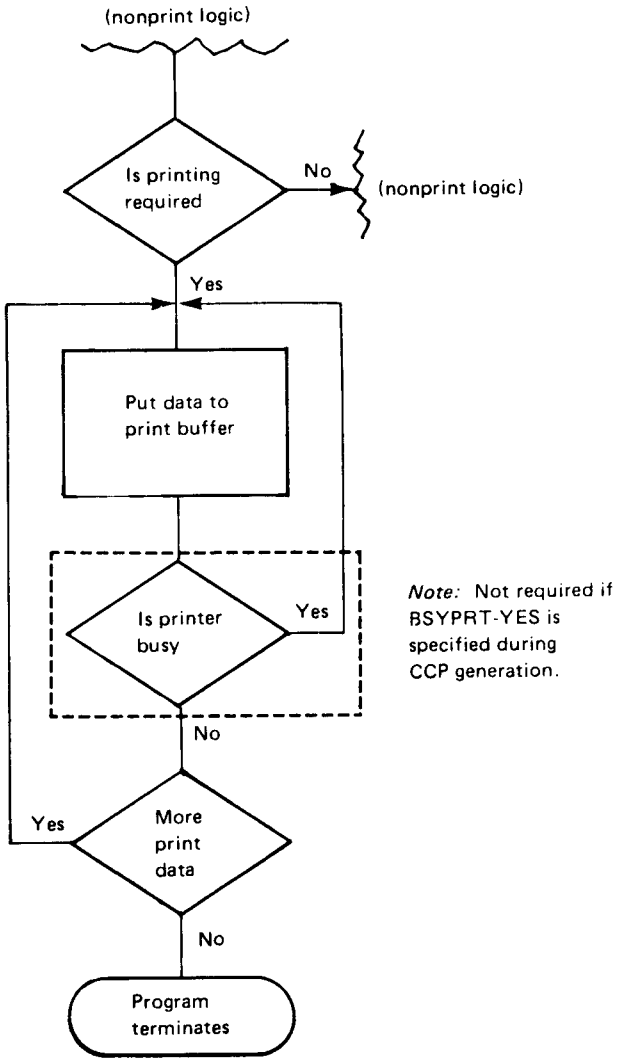


Figure 13. Logic of a Program that Uses a Terminal Printer

Using an NEP for Terminal Printing

On a system with enough memory, a print program could be implemented as an NEP. The program would have the following characteristics:

- It would interrogate a direct disk file for documents to be printed.
- A header record in the direct file for each set of transactions would designate the terminal printer to which that file should be sent.
- The program would use the WAIT operation code or one of the other techniques described previously to allow concurrent tasks to operate without degradation.

The advantage of this approach is that a terminal operator need not request a print program. The queue for documents becomes a function of this program rather than having multiple SRT print programs waiting to be loaded. In effect, this simulates a print spool function.

Chapter 7. Task Chaining

The task chaining facility of System/3 Model 15D CCP allows user tasks to initiate other user tasks by means of a task chain operation, without system operator or terminal operator action. CCP handles a task chain operation in a manner similar to a program request. A program that is loaded by a task chain operation passes through program load, resource allocation, open, close, and terminate functions, like any other user task.

The requirements for using the task chaining facility are as follows:

- The requesting program must issue a task chain request (an output type operation) and identify an output record area for the operation.
- The requesting program must place the requested program's name in the output record area. Data may accompany the program name.
- The requested program must issue an accept input operation to receive data if data is passed from the requesting program.

The task chaining facility is similar to the PRUF (program request under format) facility: data may be passed between programs through a record area, and the requested program must issue an accept input operation to receive the data. With PRUF, however, the screen buffer is used as the record area for passing data, and therefore, all programs must communicate with a terminal. The requirement for operator intervention is a disadvantage of using the PRUF facility to request one program from another.

The task chain operation (with data) works as follows: CCP moves the data from the output record area of the requesting program to the *get (invite) input area* (see index entry) of the TP buffer, and loads the program named in the terminal name field of that output record area. When the requested program issues an accept input operation, the data is moved from the TP buffer to the input record area of the requested program. This passed data could be the name of the *next* program in the chain and would be moved by the requested program to its output record area and used to chain to the next task. In this way, many programs could be chained together, and the sequence of execution can be determined in advance by the first task, or even determined by program logic within each task in the chain.

A chained task can communicate with the command terminal that requested the first task in the chain by issuing an acquire command mode terminal operation for that terminal. However, the terminal must be either in initial mode, or in command mode and not formatted by a PRUF display.

Task chaining lends itself to online transaction-oriented processing applications in the following ways:

- Task chaining is useful for breaking an application into small, single function programs.
- Batch programs can be run under CCP without operator intervention, because a chained task need not communicate with a terminal.
- MRT/NEP (never-ending MRT) programs can be chained and used as resource handlers.

BREAKING APPLICATIONS INTO SMALL PROGRAMS

Task chaining is a useful technique for dividing the functions of an application among small, efficient programs. For example, a small, simple program can gather order entry information from terminals; and, when an order is completed, that program can chain to another program that will process the information. This processing program may chain to a sort program which in turn chains to a final program that prints out the order. This allows an online application to be truly oriented toward transaction processing, completely handling one order at a time with small, efficient programs that are loaded only when required. An example of this use of the task chain operation is given in Chapter 8, *Sort Under CCP.*

Before using task chaining in this way, the system designer should assess the critical needs of the application and the effect of task chaining on system resources. Additional program loads with task chaining may increase response time at the terminal. If fast response time is the critical need of the application, then task chaining probably should not be used. If, on the other hand, the critical application requirement is that several applications must operate concurrently in an online transaction processing environment, then task chaining may be a good technique to use. The concepts of simplified queuing theory (Chapter 10) can be useful in analyzing the effect of task chaining on system resources.

RUNNING BATCH PROGRAMS UNDER CCP

Another use of the task chain facility is to run batch programs under CCP. There are advantages to doing this: for example, a batch job stream (cataloged procedure) has five programs to be executed in sequence. Four of these programs are 8K programs while the fifth program is 22K and the largest program in any other job stream is 12K. The batch partition size must be large enough for the largest program (22K); therefore, this main storage space must be reserved for long periods of time even though the large program executes for only a short time.

This same application could be rewritten to run under CCP, with each program chaining to the next logical step. The programs would then be in main storage only while they are executing, and, because 10K of the 22K partition can be assigned to the CCP partition, more program tasks can execute concurrently under CCP.

CHAINING TO RESOURCE HANDLERS

Another use of task chaining is to code MRT/NEP programs as resource handlers with other programs chaining to these programs, passing data along with the request. Two instances of this use are:

- A program that writes records to the transaction file
- A program that performs write operations to the terminal printers

Transaction File Writer Program

The first example of a resource handler program is a program that writes records to a transaction file (assumed to be a direct file). Since the program is an NEP, it is always in main storage when CCP is running, and therefore it can be coded to maintain record address pointers within the program logic. Because there are no forward and backward pointer records in the transaction file to be accessed or updated, only one disk access is required for each transaction record added to the file. By reducing the number of disk seeks, this way of using task chaining can significantly reduce disk activity, thereby improving terminal response time and system throughput.

Example

An MRT/NEP program that is used to control the writing of records to a direct transaction file is serving several terminals that are entering transaction records into the same transaction file. It is desirable to tie together all entries for a particular terminal. This is accomplished by having each transaction record from a terminal linked to the other transaction records from that terminal.

There are various ways of linking records together. In this example, reserved fields in each record are used as linkage pointers. These pointers are the relative record numbers of other records in a chain or queue that came from the same terminal.

The first record in the transaction file is a master pointer record that retains the status of all the terminal queues in the transaction file. When the MRT/NEP program is loaded, it first loads this pointer record into a control array so that the program can determine the status of the file. When the MRT/NEP goes to end of job (at the end of the CCP run), the final action it performs is to update this pointer record from the control array.

The master pointer record contains the following entries for each terminal:

- Symbolic terminal name
- Relative record number of the first record of an order
- Relative record number of the next available record to be written into
- A flag byte that indicates whether this is the first, middle, or last record of an order

Figure 14 shows how the master pointer record would appear when the transaction file is created.

The first record from each of the terminals will be written into record 2 through n + 1, where *n* is the number of terminals to be served by the program. The next available record (n + 2) is reserved for the first terminal to enter data. The relative record number n + 2 will be written into the forward pointer field of the first record from the terminal and the next available record counter will be increased to n + 3. By this method, record locations are reserved in advance, because the forward pointers are established when the current record is written.

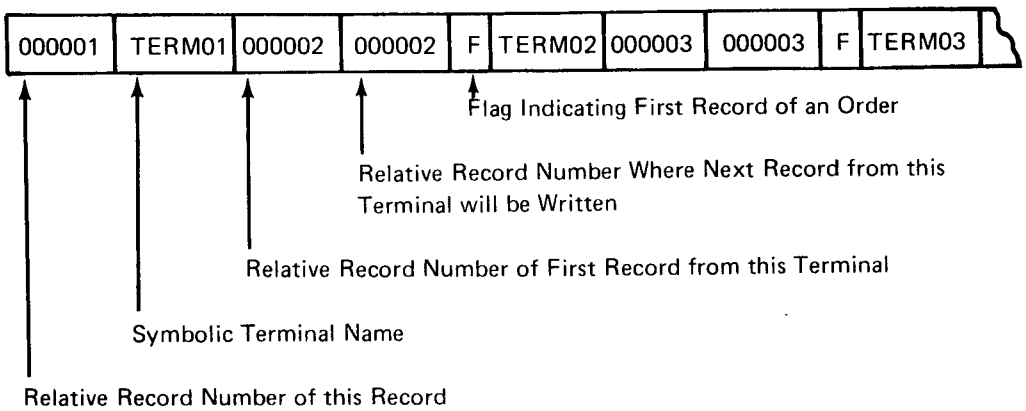


Figure 14. Content of the Master Pointer Record when the Transaction File is Created

The pointer values are contained in an array that has an element for each terminal, as shown in the following diagram:

Array

TERM01	First Record	Next Record	F/M/L
TERM02	First Record	Next Record	F/M/L
TERM03	First Record	Next Record	F/M/L
TERM04	First Record	Next Record	F/M/L
TERM05	First Record	Next Record	F/M/L
TERM06	First Record	Next Record	F/M/L
Symbolic Terminal Name	Relative Record Number of First Record of this Order	Relative Record Number Where Next Record will be Written	Flag Byte

The program maintains a counter that always points to the next available record location. This counter is used to update the *next record* field in the array. The array element associated with a terminal is then written into the transaction record when that record is written to the file. In effect, the relative record number for the next record from this terminal is reserved at this time. Records are reserved in sequence, but are not necessarily written in sequence.

For records other than first records, the *first record* field points to the first record of the current order. For the first records of an order, this field points to the first record address of the previous order, so that records are linked forward by record within an order and backward by first records of all orders.

The logic of the program is illustrated in Figure 15.

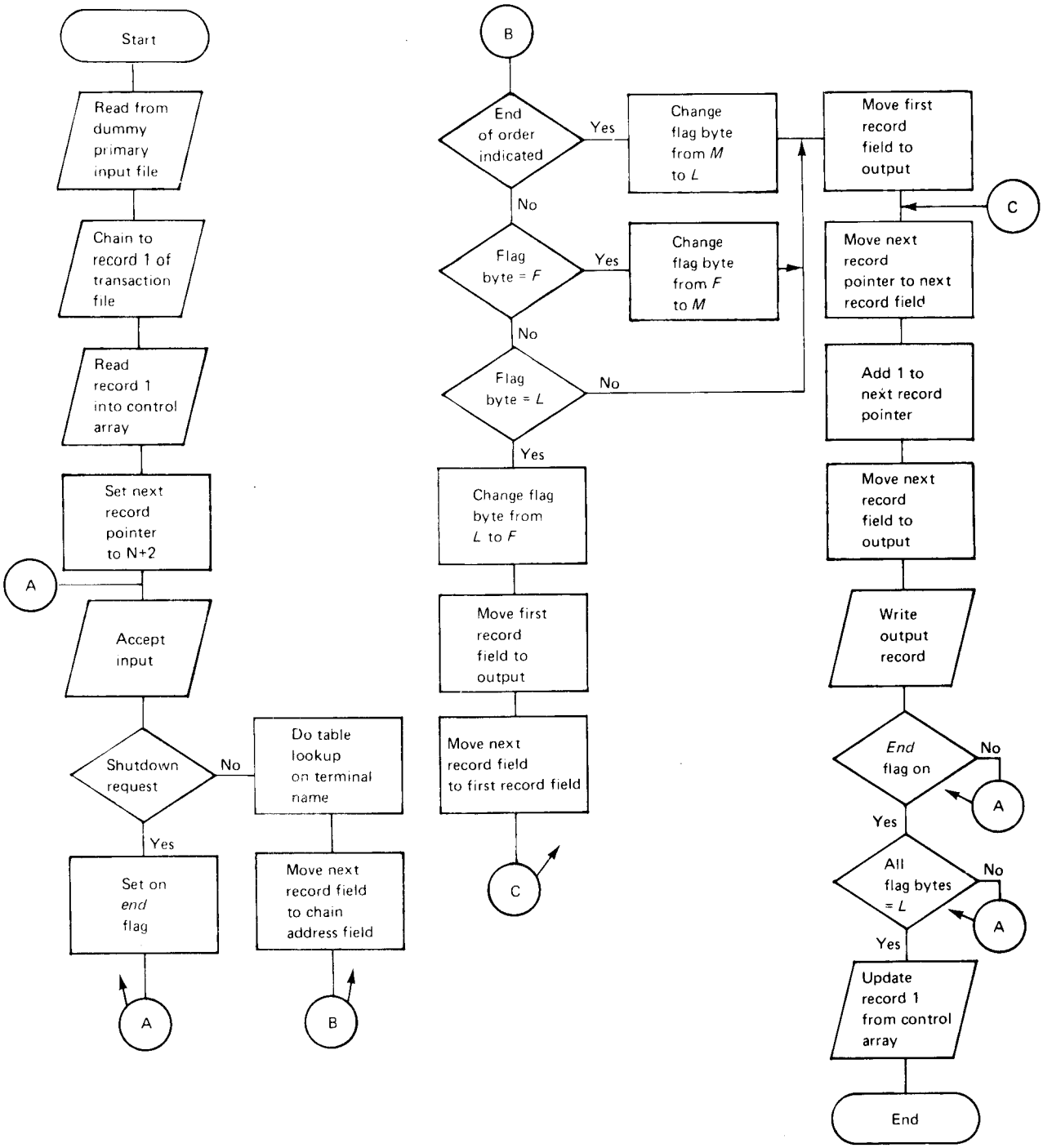


Figure 15. Logic of a Transaction File Writer Program

Terminal Printer Program

A second resource handler technique is using an MRT/NEP program to perform the write operations to the terminal printers. A single program of this kind can take care of all the programming required to make efficient use of these devices. One factor that has made efficient use of these printers somewhat difficult is that a program is notified when the printer receives a message, but is not notified when the printer has completed printing the message. A negative return code is returned from an output operation to the printer if it is still busy, but line activity is increased if a program repeatedly retries the operation without an intervening wait. By chaining to a single MRT/NEP for all terminal printing, the user tasks can keep printers busy, reduce pauses between messages, and yet not burden the communication line by testing for busy conditions. The following example describes a way to implement a resource handler program for terminal printers.

Example

A program needing to make use of a printer chains to the MRT/NEP printer task, passing the record to be printed along with the request, including, if needed, an identification of the terminal printer. The printer program is residing in main storage with an accept input operation pending.

The program uses the TP buffer as the storage medium for the print queue by processing one message completely before accepting input on the next message waiting in the TP buffer.

CAUTION
If there is a backlog of messages queued in the TP buffer using up all available space, subsequent task chain requests are rejected and the request is not queued by CCP. Terminal requests to the TP buffer are queued and honored immediately when space becomes available. Therefore, even when the requesting program retries the request, space still may not be available and the system may be limited to the speed of the terminal printers.

The program accomplishes the task of keeping the printer busy by performing variable waits. With the timer support and the use of the wait op code, the printer program puts a message to a printer and issues a variable wait, dependent upon the number of characters to be printed. One way to determine message length in RPG II programs when SUBR92 is not used is to load the record from the requesting program into an array with enough one-position elements defined to hold the longest record and then to scan backward through the array to find the first nonblank element, thereby determining the number of characters to be sent to the printer. This value determines the output length value for the put operation to the printer and is used as an algorithm to set the value for the wait operation code. Number of characters divided by characters per second plus time for printer carriage return and line spacing equals seconds of wait.

Coding must be in the program to handle the -14 return code, but the occurrence of this should be infrequent with proper design.

In many cases, more than one printer is in the system, and messages from a terminal are sent to a specific terminal printer. Either the *to* printer name will have to be passed with the task chain data or the *from* terminal name will have to be passed with the program using an array and a lookup operation to match a *from* terminal name to the proper output printer.

Chapter 8. Sort Under CCP

The CCP/Disk Sort program is a program product that is used to sort disk files according to user specifications. The functions of CCP/Disk Sort are similar to the functions of the System/3 Disk Sort Program, except that a user of CCP/Disk Sort can generate a sort object program that can be executed as a user task under CCP control. Multiple sorts, each having a unique program name, can be run concurrently under CCP. CCP/Disk Sort object programs must be generated offline from CCP.

Sort programs can be requested by the system operator, a terminal operator, or by another program through a chain task command (System/3 Model 15D only–see Chapter 7, *Task Chaining*). The requested sort program issues an accept input operation (PGMDATA-YES must be specified on the PROGRAM assignment statement) and then releases the requesting terminal, if the requester was a terminal. Thus, a requesting terminal is free for other work while the sort is running. The sort program cannot communicate with a terminal while the program is running. If the sort program is running when a shutdown command is entered, the program is allowed to complete before the shutdown process begins.

For each sort program, the input file(s) must have consecutive input (CG) access type, the work file must have consecutive add (CA) access type, and the output file must have consecutive output (CO) access type (PROGRAM assignment statement).

The sort input file(s) may be shared, but the sort work file and output file cannot be shared (NOSHR on the PROGRAM assignment statement).

If task chaining is specified in the CCP/Disk Sort header specification, a chain task request is issued by the generated sort module for a task to follow the sort task. The name of the task to be chained to by the sort program must be passed to the sort program as program data with the sort program request.

CONSIDERATIONS FOR USING CCP/DISK SORT

With CCP/Disk Sort, as with all other system facilities, tradeoffs must be made to make the operating performance of the system fit your requirements. For example, using CCP/Disk Sort may increase processing unit utilization; nonshareable files may conflict with other program requests; response time may increase; and main storage availability may cause program requests to be rejected. These are performance factors that must be considered when using CCP/Disk Sort. Some of the benefits you may realize are: workloads in using departments can be spread throughout the day; master file status can be current to the latest transaction; batch run time requirements can be reduced; and the need to shut down CCP may be reduced.

TRANSACTION-ORIENTED PROCESSING WITH CCP/DISK SORT

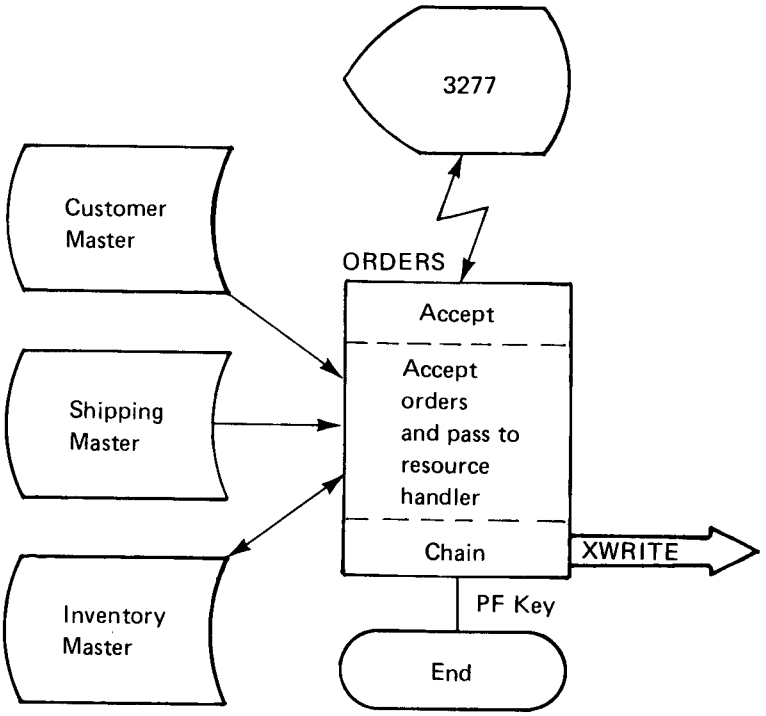
The sort program can be used in online transaction-oriented processing applications. For example, the input items making up one order can be sorted for that order. Sorting for each order will, of course, increase the system time required for each order and may well take more total time for one day's processing than if one sort were run to process all orders at the end of the day.

Therefore, the decision whether or not to incorporate sort programs into a transaction-oriented processing application is not always an easy one to make. Order entry applications are an obvious place to start using CCP/Disk Sort, but the number of transactions (orders) per day must be considered. If the system is near its limit of transaction capabilities using batch sort procedures, then CCP/Disk Sort programs probably should not be used in an application. If, however, the application bottleneck is that picking ticket slips are sent to the warehouse twice a day with a resultant rush in the warehouse workload, then perhaps full transaction-oriented processing may be a wise course to follow.

Figure 16 describes a transaction-oriented order entry application using the facilities of task chaining and CCP/Disk Sort.

ORDERS Program

ORDERS is an SRT program that collects orders entered by an operator at a 3277 terminal. The ORDERS program accesses the customer master and shipping master files and updates the inventory master file. For each item, the ORDERS program chains to the XWRITE program, passing the transaction information with the chain task request. The terminal operator presses a PF key when the last item for an order has been entered; this causes ORDERS to pass a last record flag to the XWRITE program.



Data with XWRITE chain task request

XWRITE	terminal name	last record flag	invoice number	customer number	warehouse location	item information

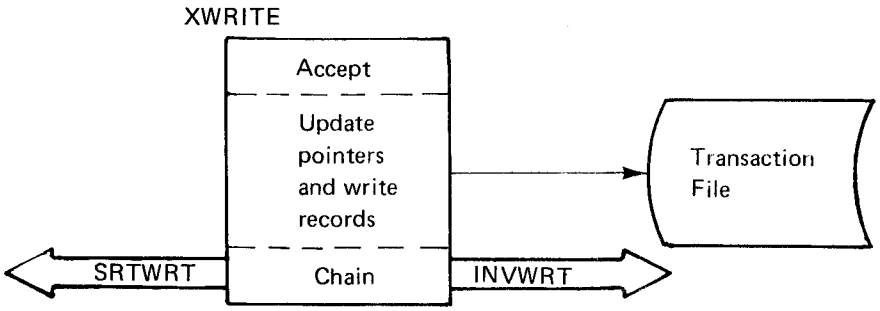
Assignment Statement

```
// PROGRAM NAME-ORDERS,PGMDATA-YES,
           FILES-'CUSTMAST/IR/SHR,SHIPMAST/IR/SHR,
           INVENTRY/IRU/SHR',DFFMTERM-1,DFFNDF-n,
           DFFSFDT-n
```

Figure 16 (Part 1 of 6). Example of Using CCP/Disk Sort and Task Chaining (Model 15D Only)

XWRITE Program

XWRITE is a never-ending program that accepts transaction information from the ORDERS program and writes transaction records to the transaction file (XACTION). When XWRITE receives a last record flag from the ORDERS program, XWRITE issues two chain task requests. The first request is for the INVWRT program, which prints the billing invoice on the system printer. The second request is for the SRTWRT program, which writes order items from the transaction file to a sort input file.



Transaction Record

relative record	next record or last	invoice number	customer number	warehouse location	item information	terminal name

Data with SRTWRT chain task request

INVWRT	first record	invoice number	customer number

Data with SRTWRT chain task request

SRTWRT	first record	invoice number

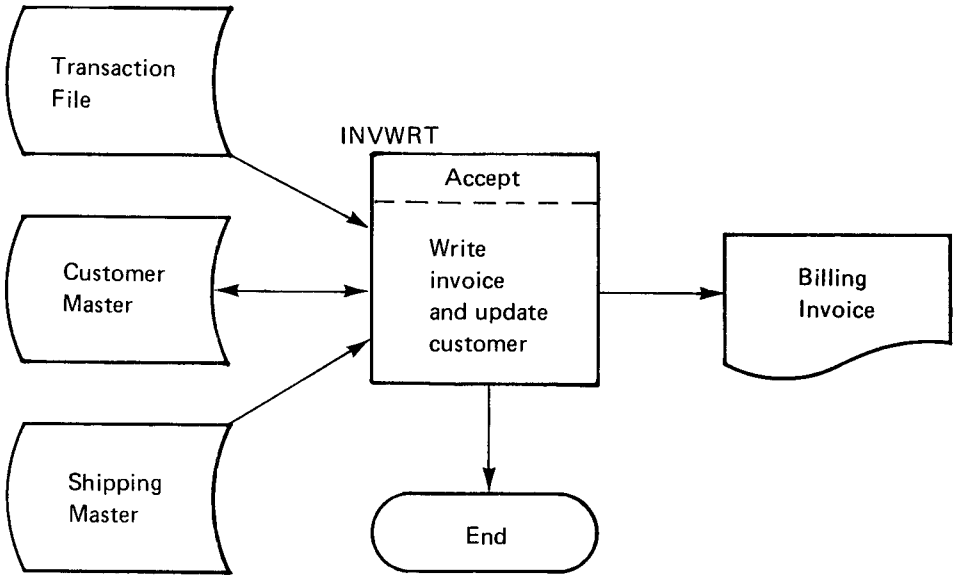
Assignment Statement

```
// PROGRAM NAME-XWRITE,NEVEREND-YES,PGMDATA-YES,
          FILES-'XACTION/DO/SHR'
```

Figure 16 (Part 2 of 6). Example of Using CCP/Disk Sort and Task Chaining (Model 15D Only)

INVWRT Program

INVWRT is an SRT program that reads order items from the transaction file, reads shipping information from the shipping master file, and prints billing invoices on the system printer. INVWRT also updates the customer master file.



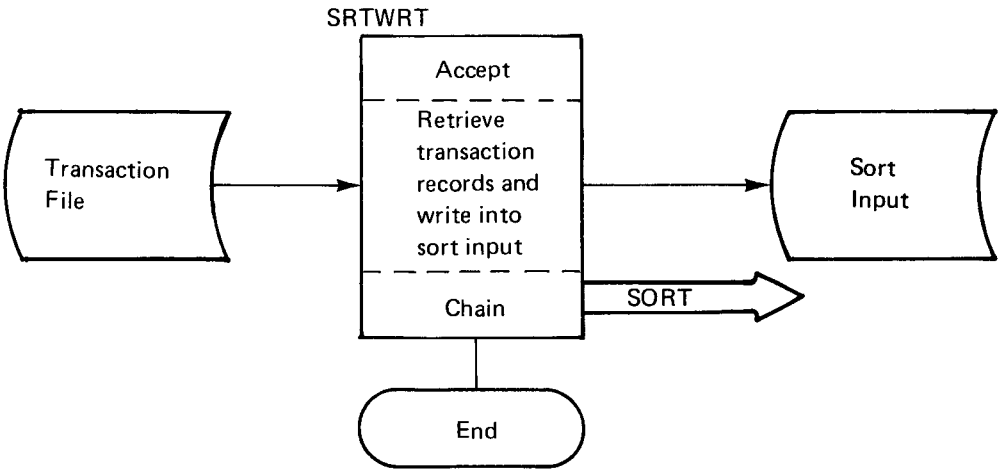
Assignment Statement

```
// PROGRAM NAME-INVWRT,PRINTER-YES,PGMDATA-YES,
           ENDMSG-NO,FILES-'CUSTMAST/IRU/SHR,
           SHIPMAST/IR/SHR,XACTION/DG/SHR'
```

Figure 16 (Part 3 of 6). Example of Using CCP/Disk Sort and Task Chaining (Model 15D Only)

SRTWRT Program

SRTWRT is an SRT program that writes records from the transaction file to the sort program input file, chains to the sort program, then goes to end of job. Because SRTWRT uses the sort input file as CO (consecutive output) access type, old data in the file is written over, and the file can be reused.



Sort Input Record

invoice number	customer number	warehouse location	item information

Data with SORT chain task request

SORTbb	PIKWRT	invoice number	customer number

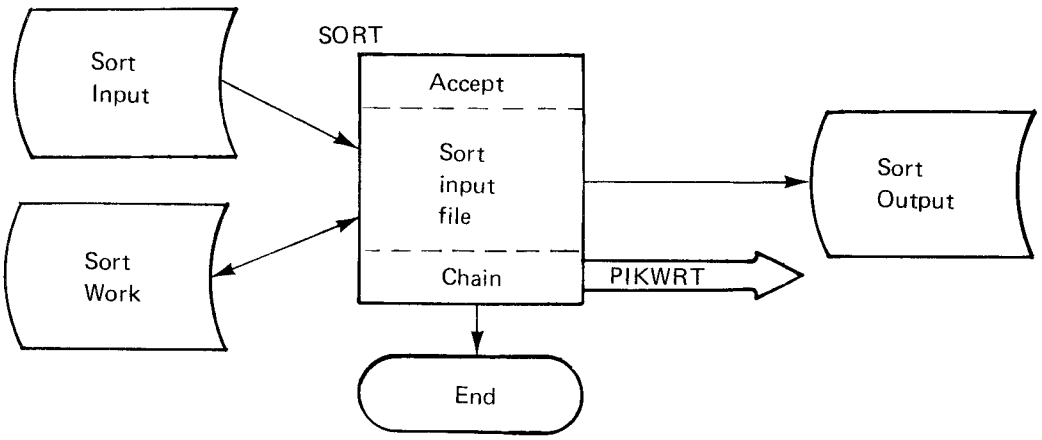
Assignment Statement

```
// PROGRAM NAME-SRTWRT,PGMDATA-YES,ENDMSG-NO,
          FILES-'XACTION/DG/SHR,SORTIN/CO/NOSHR'
```

Figure 16 (Part 4 of 6). Example of Using CCP/Disk Sort and Task Chaining (Model 15D Only)

SORT Program

SORT is an SRT program that sorts transaction records into warehouse location sequence and then chains to a terminal print program (PIKWRT). Warehouse location information is inserted into the transaction records by the ORDERS program.



Data with PIKWRT chain task request

PIKWRT	invoice number	customer number

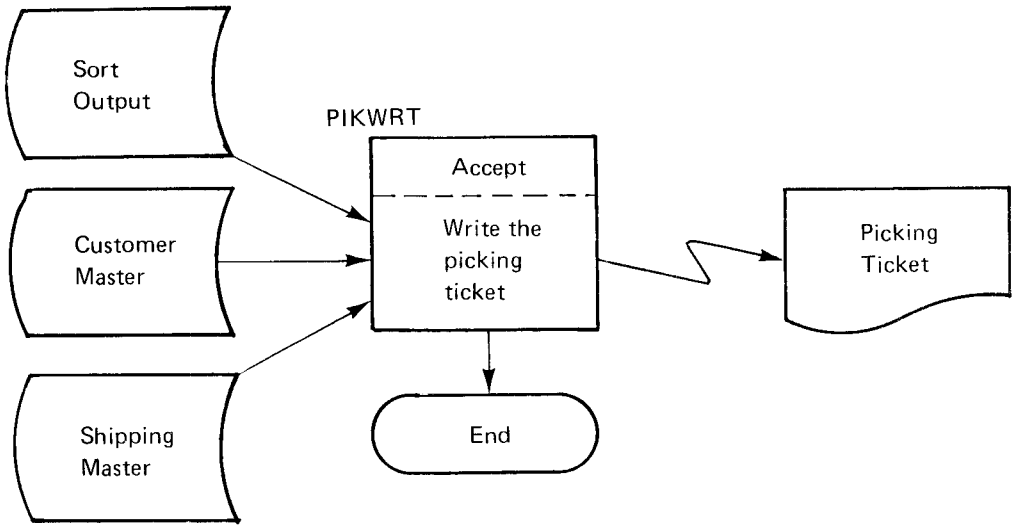
Assignment Statement

```
// PROGRAM NAME-SORT,PGMDATA-YES,ENDMSG-NO,SORT-YES,
          FILES-'SORTIN/CG/NOSHR,SORTOUT/CO/NOSHR,
          SORTWORK/CA/NOSHR'
```

Figure 16 (Part 5 of 6). Example of Using CCP/Disk Sort and Task Chaining (Model 15D Only)

PIKWRT Program

PIKWRT is an SRT program that prints a picking ticket with a mailing label header on a remote terminal printer at the warehouse location. Items are printed in warehouse location sequence, as they appear in the sort output file.



Assignment Statement

```
// PROGRAM NAME-PIKWRT,PGMDATA-YES,ENDMSG-NO,
           FILES-'SORTOUT/CG/NOSHR,CUSTMAST/IR/SHR,
           SHIPMAST/IR/SHR',DFFMTERM-1,DFFNDF-n,
           DFFSFDT-n,TERMS-'PRINT'
```

Figure 16 (Part 6 of 6). Example of Using CCP/Disk Sort and Task Chaining (Model 15D Only)

Chapter 9. System Security/Integrity

This chapter presents system security/integrity considerations for an online system using local or remote terminals. Some of the information applies specifically to operating 3270 terminals under CCP, but most of the information is applicable regardless of the terminal type used.

System security is defined as protection of computer data, programs, and devices against damage, loss, unauthorized access, or unauthorized use. The scope of system security also includes protection of other assets against damage or loss through misuse of computer facilities.

System integrity is defined as preservation of the accuracy and completeness of data and programs. The scope of system integrity includes the capability to prove the accuracy and completeness of data and the capability to restore the system and files after an unscheduled interruption in processing.

In a batch environment, system security/integrity is usually maintained by keeping copies of master files, batch balancing and editing input transactions before update, and keeping a control book. Source documents are accompanied by transmittal slips and, since the whole data processing operation takes place in a central location, procedures for maintaining system security/integrity can be relatively uncomplicated.

In an online environment with local or remote terminals, the requirements are somewhat more complex. This chapter covers the following five aspects of maintaining system security/integrity in an online environment:

Transaction logging
Audit trail
Control procedures
Data security
Backup and recovery

In each of these areas, plans should be developed concurrently with the application and should be an integral part of the overall operating plan.

TRANSACTION LOGGING

In a batch system, input data that is used to update files is normally recorded in cards or on diskettes. The system reads the data in these records (transactions), edits it, and writes it to disk. The resulting disk file might then be sorted and used to update master files. If a system failure occurs during the master file update, the procedure is to copy master files back from backup files, and then rerun the job. Effectively, input transactions are being *logged* to disk before using the data for update purposes.

In an online system, a somewhat similar approach can be used; that is, transactions received from remote terminals can be written to a *transaction file* (or *transaction log file*) on disk. However, master files would not usually be updated with this data in batch mode, because an inquiry to a master file would provide only information as of the most recent master file update, and transaction-oriented processing could not be done. Logging transactions to disk is a good procedure because it allows rerunning a job, if necessary, and thus restoring system integrity after a failure. This is one of the reasons for logging all transactions in an online system. Another reason is that the disk file can be used to provide an audit trail through the system (see *Audit Trail and Control*).

The $TRLOG program on the Model 15D can be used to assist the user in applications requiring transaction logging. This program provides both batch and CCP programs with the ability to log data to tape. The data logged can be used to create tape audit trails, transaction logs, program use statistics, terminal use statistics, and debug information.

Transaction Data

What data should be logged in the transaction file? This question can only be answered by the requirements of the system, and for this reason, recovery, restart, and audit requirements should be kept in mind when designing a system or new application. Not only do master files have to be restored after a failure but terminal operators must be able to start again from the point at which the system failed; that is, they need to know what transactions if any have to be reentered.

A checklist of data fields that can be contained in each transaction record is given below. This list is a suggested minimum for restart and audit purposes; the actual fields depend on the needs of the application:

- Record code
- Date and/or time of transaction (if timer support is available)
- Reference numbers, for example, account numbers and order numbers
- Amount or value of transaction
- Content of master file fields before update
- Terminal name or identification code
- Operator name or identification
- Program name
- Master file name

The record code (if used) should be unique because it can be used as an indication of what program created it, and which master file was updated by that transaction. To ensure that the record codes are unique, a register should be kept of all the codes allocated; this register should be cross-referenced to program and master files.

The transaction file could be shared by all tasks running in the system, and records could be added to it whenever changes or updates are made to master files. A benefit of logging all transactions to one file as they occur is that it is possible to reproduce the sequence of all transactions, whereas if separate files were kept for each application, the entries would have to be time stamped in order to establish this relationship. If timer support is available, better performance may be possible if each application has its own transaction file, because the potential for disk access conflicts is reduced.

When you establish a record length for your transaction file, make sure that you allow for new applications that may require a longer record.

AUDIT TRAIL

An audit trail is a generalized recording of who did what to whom, when, and in what sequence. For example, it may be necessary to trace a receipt in an accounts receivable application backwards from the master record it updated to its point of origination.

An audit trail should provide the information that would be used by someone outside of a data processing department to prove that the system is doing the job correctly. The audit trail additionally should provide this person with sufficient information as to *who, what, when, and why* so that errors can be identified and corrected. If errors are discovered, it must be possible to isolate the conditions that gave rise to the error; for example, it may have been a program failure, a combination of events, or even fraudulent action by someone in the organization. After the reason for the error is identified, the next question that must be answered is: Has this occurred before and who or what was affected by the error?

Implementing an Audit Trail

A good way to start planning the implementation of an audit trail system is to discuss the system with the financial controller, accountant, and perhaps a representative of the company's external auditor. Find out what information they need to perform their function and how they want the information presented. Perhaps they would like to be able to put test transactions through the system to verify the audit trail. There are many different ways to handle an audit and each company has its own specific needs.

The transaction logging file discussed earlier in this chapter is a good place to start, since some of the information stored there for restart purposes is also required for the system audit function. Based on the uses recommended for it so far, the transaction logging file may be very large, but a transaction logging file is the type of file that can be purged each day, and the data contained in this file can be split into historical files by application. For example, all transactions related to accounts receivable could be written into a history file for this application. Only the data required to reconstruct the sequence of events by account need be copied into this file. These application history files need not be online at all times, but only when required for audit purposes.

An audit trail can be a method of controlling or ensuring system integrity *after the event* and, as such, can make it possible to prove or check that transactions generated by the day-to-day operations of your company have been processed correctly. However, an audit trail alone does not provide adequate control in a company's daily operations. Control procedures are also required that will prevent errors.

CONTROL PROCEDURES

The objectives of control procedures are to ensure accuracy and to prevent accidental or intentional modification or loss of data. Control must be exercised in two areas: manual procedures and programmed procedures.

Manual Control Procedures

Manual control procedures apply in all areas of a company operation. Discussion in this manual is limited to control within the data processing department in an online environment.

Manual control procedures are best achieved by the division of duties among employees. One employee should not be solely responsible for, or have full control over, critical aspects of the business without some counter check over which the employee does not have control.

Programmed Control Procedures

In a batch environment, the conventional control function of punch/verify, batch balance/edit of input works well because of the centralized operation. The accuracy of input is proved before being used to update files and prepare reports. In an online system, the source input is randomly entered at the terminals, and data is not necessarily accumulated into batches before submission to the computer system, so the control function has to be slightly different.

This is not to say that online processing precludes the use of batch control methods. On the contrary, if local or remote 3741 data entry terminals were used for remote data capture, the data submitted for processing to the central system would still, in fact, be batched, and the conventional control and error correction procedures could be used to advantage.

Interactive Control

However, if the terminal is operating in an interactive manner in which the program must accept input, perform an edit, and update (if accepted), the control is then more difficult to impose. For example, what assurance do we have that the amount entered by the terminal operator was the amount paid by the customer in an accounts receivable application? There is little point in asking the terminal operator to enter the amount twice, since errors may be repeated. The program could compare the amount entered with the customer balance from the master record and, if equal, accept the amount as correct. If the balances are unequal, the program could issue a query to the operator and allow the operator to either correct or confirm the balance. As a final check, perhaps the overall receipting operation could accumulate totals by terminal and, in this manner, make any discrepancy traceable to any one terminal operator.

The reconciliation of control totals with the actual amounts taken should not be done by the terminal operator who entered the amounts. Part of the reconciliation statement should reflect all receipts where the amount entered is confirmed, but less than the outstanding balance. The supervisor then is able to establish whether or not a pattern exists in the nonbalance situations. The pattern would be that check payments that can be verified were recorded accurately but cash payments are less than the balance owing on a fairly regular basis. Do not display the balance outstanding to the receipting operator; simply display an indication that the amounts differ, and display the amount entered by the operator. If a customer requests his outstanding balance, he could be referred to another operator who can make the inquiry but is not able to update the record.

Methods of control in an online terminal environment depend upon circumstances. When determining methods of control, the system designer must anticipate the error situations that the terminal operator may encounter and take appropriate precautions. An operator working under pressure cannot be relied upon to do a visual verification of every entry made. Check-digit verification can be used for account numbers, and related names and addresses can be displayed to ensure posting to the correct account. Use of modulus 10 and modulus 11 to calculate self-check digits is explained in the following text.

Using Modulus 10 to Calculate Self-Check Digit: The subroutine:

1. Multiplies each digit of the account number by the corresponding digit of a weighting factor. A five-digit weighting factor is used in this example–the *X* could be shifted to any digit of the account number.

Account number	5	2	0	5	6	3
Weighting factor	2	1	2	X	1	2
	10	2	0		6	6

2. Adds each digit of the products:

   1 + 0 + 2 + 0 + 6 + 6 = 15

3. Determines the next number divisible by 10 that is higher than the sum computed in step 2.

   From the sum 15, 20 is the next higher multiple of 10.

4. Subtracts the sum computed in step 2 from the number determined by step 3. The difference is the self-check digit.

   20 - 15 = 5

   The number 5 is the self-check digit.

Using Modulus 11 to Calculate Self-Check Digit: The subroutine:

1. Multiplies each digit of the account number by its corresponding digit of the weighting factor.

Account number	5	2	0	6	3	2
Weighting factor	6	5	4	3	2	X
	30	10	0	18	6	

2. Adds the products:

   30 + 10 + 0 + 18 + 6 = 64

3. Determines the next number divisible by 11 that is higher than the sum computed in step 2.

   From the number 64, 66 is the next higher multiple of 11.

4. Subtracts the sum computed in step 2 from the number determined by step 3. The difference is the self-check digit.

   66 - 64 = 2

   The number 2 is the self-check digit.

Online Batch Control

If a system requires that a terminal operator enter data in a batch mode using source documents for reference, the method of control could be very similar to the method of control for a batch operation. The control could take the form of requiring the operator to enter first a batch header record that contains a count of the number of records to be entered and a total of the value of these records. Each data record entered would be edited against master file records, and error messages would be displayed at the terminal, allowing corrective action to be taken. No master file update is done at this time. As each data record is accepted by the system, it is written to the transaction file.

The count and value fields in the header record are reduced for each transaction. When the operator indicates the end of the batch by pressing a PF key on the 3270 terminal, rather than the ENTER key, the program updates the master files if these control fields are zero. If, however, the header control fields are not zero, the program displays the record entered and the balance in the control fields. The operator can then make the necessary corrections and again indicate the end of the batch. If the net effect of the changes now satisfies the imbalance value, then the program performs the update task.

In effect, the balance and edit function has been removed from the data processing department, and the responsibility for accurate data entry has been placed in the user departments. The terminal operator in the user department must be provided with sufficient data to enable the operator to make the necessary corrections.

Data Processing Department Controls

In addition to the control within user departments, procedures must be adhered to within the data processing department. The data processing department should not generate any input transactions to the system. All transactions should originate from outside the department. The responsibility of the computer department should be limited to ensuring that information entrusted to its care is not lost, destroyed or distorted.

When CCP is shut down, the normal batch-type controls should be applied for processing files updated or created by the online system. For example, before purging the transaction log file, extract from it totals by record type and application. These totals should be reconciled with master file opening and closing balances before making backup copies of master files for security. The reason for doing this is to make sure that all updates have been made to master files; there may have been a system failure during the day and, if the recovery procedures were not done correctly, there may be some records in the transaction file that have not updated their corresponding master records. As far as user departments are concerned, they have entered their data correctly, so it is up to the data processing department to process it correctly.

DATA SECURITY

Data security can be defined as the protection of data aginst damage, loss, unauthorized access, or unauthorized use. Two basic aspects of data security are considered here; they are (1) physical security measures and (2) programmed security measures.

Physical Security Measures

Physical security measures traditionally include locks on doors, alarms, guards, fireproof safes, and off-site storage. These measures are concerned with protecting tapes, disk files, printed reports, and programs against destruction, such as by fire, and against access by unauthorized people. Most organizations recognize that fire can destroy data files, and they protect against this by storing copies of master files in fireproof safes or in vaults that are remote from the data processing center. It must also be recognized that it can be just as damaging to an organization's operation if confidential information falls into the wrong hands as if this information is physically destroyed.

All of these security considerations are as important in an online environment as they are in a batch environment. However, in an online environment, there are some additional areas of concern. The distributed nature of a terminal network makes the control of access more difficult. The person using a terminal may not be visible to the system operator, and therefore it is more difficult to verify that the person is authorized to use the terminal. Data security in this kind of environment requires a combination of physical and programmed security measures.

Risk Versus Cost

When evaluating physical security measures, two questions must be asked: What is the risk of a particular event occurring? What would the cost be to the company if the event did occur? The answers to these questions should be assessed by management outside the data processing department. Absolute security is impractical, if not impossible, to achieve, therefore, the security budget should be allocated to cover the risks where the threat of loss is the greatest.

For example, the loss of certain data due to fire might cost the company $40,000. The probability that this fire might occur is once in 20 years. On the other hand, an event that causes a $20 loss and is likely to occur every working day is a greater risk. Calculating the cost per year of a particular loss can be a useful way to establish the relative importance of the loss.

Types of security threats are:

- Physical hazard: fire, water damage, power loss, wind, explosions, civil disorder
- Hardware/program failure
- Carelessness: terminal operator, system operator
- Malicious damage: programmer, operator, terminal user
- Crime: fraud, embezzlement

The above list can be expanded into a table that gives yearly risk probability and estimated dollar loss to your operation. The probability times the dollar loss provides the yearly weighted risk values. The sum of the weighted risk values is the estimated loss if no security measures are taken. It is then reasonable to assume that annual operating expenses (including depreciation of capital equipment) equal to the aggregate estimated losses could be justified for security measures.

Fraud Protection

A final aspect of physical security that should receive close attention is that of fraud protection. The following are suggested measures to protect the system against fraud:

- Balance cash and accountable items frequently.
- Apply strict validity checks.
- Control access to tape and disk libraries.
- Batch balance and control online files.
- Log and review computer operator actions.
- Thoroughly test and review new programs or changes to existing programs before including them in the system program.
- Allow only official operators into the computer area.
- Control access to data and terminals.
- Divide user and staff responsibilities.
- Restrict the knowledge of how the total system works to the fewest possible people.
- Set up an internal audit group whose function is to continuously review the system for security breaches.
- Maintain an orderly operation. Do not leave tapes, disks, and listings lying around the computer area.
- Keep a high standard of documentation but make sure that this documentation is kept in a secure location.

Programmed Security Measures

In a batch environment, control of access to information in files is a function of the data processing department staff. If a printout of sensitive information is required by management, a request is submitted, usually to the data processing manager, who perhaps retains control of the only copy of the program to do the job. The resulting printout and any carbon paper used are then handed to the person requesting the printout. In a remote terminal operation, where all files are online, access controls must be built into a system, otherwise sensitive information could be available to any person who had access to a terminal.

When designing a system or application in a remote terminal environment, the designer should look at the data to be stored in the files and decide:

- What level of security must be applied to this data?
- Which group of people needs access to this data?
- To what extent is modification allowed and by whom?

The information stored in computer files can be classified according to the degree of security required to prevent unauthorized access. For example, the following classifications might be used:

Top secret
Secret
Company confidential
Unclassified

The type of data that might be allocated to these categories depends upon an organization's security policy. Some information may be of such a sensitive nature that management would never allow the data to be stored in computer files.

Besides being classified by degree of security, data can be classified departmentally. For example, production planning information should be of little interest to anyone in the payroll and personnel departments. The reverse situation, however, is not necessarily true. Payroll and personnel information is generally of interest to anyone in the company. It is necessary, therefore, to inhibit access to data on the basis of a need to know. Data access can also be inhibited on the basis of what the terminal operator should be allowed to do with the data. Even within one department employees have different responsibilities: some may be allowed to access only parts of a data file and not be permitted to make any changes, while others in the same department are allowed to access greater portions of the data and in some cases make changes to, add to, or delete records from the file.

Sign-On Security

The password used for CCP sign-on must be considered the lowest level of security; it provides, at best, protection against the casually interested outsider. It would not be difficult for a more determined individual to discover the password and sign on to the system.

The user security interface to CCP ($CCPAU) allows sophisticated techniques to be used and allows the passwords to be changed as frequently as required. Changing a password provides better security and the more frequently it is changed, the greater the protection. However, frequent changes give rise to other problems, such as the need to advise authorized users more frequently of the new password and the possibility of confusing the passwords. If an operator writes down the password instead of memorizing it, security is immediately compromised.

Sign-on security, whatever the method used, has a limitation in that too many people need to know the current password(s) and the more people who know it, the lower is the level of security.

Access to Data Files

Three general methods of limiting access to data files are described here:

- Using another password, in addition to the sign-on password
- Using CCP symbolic file support
- Manipulating terminal names

Using Additional Password(s): An easy way to limit access to data files is to use another password that is checked by the program as the first data field entered from the terminal with the program request. If this password is coded in the program, then it is necessary to recompile the program whenever the password needs to be changed. A better way to check passwords would be to have the passwords recorded in an execution-time table of valid passwords that are related to employee numbers within a department. Each employee would then be given a unique password that he should memorize and not divulge to anyone.

The operator should not enter password data with the program request, since this data would be displayed on the screen. Rather, a first time routine should be included in the program to prompt the operator with a brief message to ENTER PASSWORD. The input fields for this screen format would be coded as nondisplay field types (type 7 or 8). The operator, after being prompted, would enter department number, employee number, and unique password. The program would then use the combined department number/employee number as an argument to look up the table of passwords for a password that corresponds to the operator's entry. If the operator's entry was not the password assigned to the operator in the table, the program could put an appropriate message to the screen and allow the operator to try again. If, on the second attempt, the passwords do not match, the program should release the terminal, put a message to the system operator, and record the event on a sign-on type log record.

The log records should contain as much information as possible within the constraints of log record size. These records should be analyzed regularly for frequency of occurrence by type, location, terminal, employee number, and time, in order to pick up patterns as soon as possible. If a pattern does emerge, it could indicate that someone is attempting to enter the system without authorization.

In order to establish departmental security where information can only be accessed by authorized personnel in that department, individual password table files have to be set up for each data file. As a further precaution, the terminal names or IDs could be included in these table files.

Some individuals within the company may need authorized access to files and programs that cross departmental boundaries. These people would have their employee numbers and passwords included in more than one table file.

In order to protect against unauthorized modification of data or deletion of records, another byte could be added to the table entry and coded according to type of access allowed. This byte value could then be used to set an indicator that conditions execution of portions of the program code. An update type program could then be selectively restricted to allow inquiry only.

A limitation in these security procedures is the fact that if someone were to gain access to the disk containing the password table files, then this individual would have complete access to all data in the system until the passwords were changed. For this reason, the disk containing the password tables should be protected as well as possible.

Symbolic File Technique: An easy way to secure files accessed by SRT programs is to write the programs to use SYMFILE support, so that the operator must use the /FILE command to reference the file. To change the *password* (actual file name), the user needs only to change the assignment set and the OCL.

Manipulating Terminal Names: The following technique can be used to allow a certain kind of transaction to be done by only one operator using one of a selected group of terminals: Alternate TERMNAMES can be supplied to the selected terminals and the TERMS parameter of the PROGRAM assignment statement can be used to require that the requesting terminal's symbolic name be its alternate name. This requires that (1) the operator know the alternate name and enter a /NAME command to change the *doing business as* name of terminal, and (2) that only one terminal at a time can be used to run the transaction.

BACKUP AND RECOVERY

The goal of system backup plans and procedures should be to minimize the impact of any breakdown in the system and to recover from this breakdown as quickly and economically as possible.

Most backup procedures used for batch systems also apply to online systems. However, there are additional backup considerations in online systems. In online applications, the entire business can become dependent upon the system; therefore, the system must have reliable backup and recovery procedures.

Hardware Backup

Mutual hardware backup plans between users with similar equipment have long been used for batch oriented systems. However, this type of arrangement is generally not practical in an online environment. Even if an identically configured system could be located, the delays and difficulty of establishing the necessary data links would preclude relying on this approach to hardware backup.

Online system hardware backup plans can vary according to the business requirements and the cost of implementing those plans. Hardware backup can range from having a spare terminal or a standby modem to completely duplicating the central system. The actual level of hardware backup provided must be determined by weighing the cost against the possible business loss if the backup is not there when it is needed.

Data Backup and Recovery

Damaged or destroyed hardware can be replaced, sometimes quickly, but lost data may be impossible to replace or prohibitively expensive to replace. Installations have had the central data processing facility completely destroyed by fire and, because there was good data backup, have recovered so quickly that their customers were not aware of any disruption in service. Other installations without data backup have had minor programming failures that caused them to lose the pointers to their transaction files, resulting in a great financial loss and immeasurable loss in customer satisfaction. The data in this case was not lost, but it could not be read or recreated because of inadequate data backup and recovery procedures.

The level of data protection can vary greatly without a significant difference in cost of implementation. The development of a backup plan and the testing of that plan can provide a potential return far greater than the time and effort involved. Developing a backup plan includes analyzing the effect of a system failure on each step of an application, defining appropriate recovery procedures, and testing those procedures. It is important that the procedures be tested. A crisis situation is not the time to find the shortcomings of a backup plan.

For backup and recovery purposes, data can be divided into different categories. These are:

- Historical data
- Master files
- Data processed but not distributed
- Data logged but not processed
- Data received but not logged

Historical Data

Historical or archives data is not used in day-to-day processing and is retained on some media (such as cards, magnetic media, or microfilm) in some secure location, preferably off site. Any audit of the backup plans should include trying to reconstruct current files from these historical files. The audit should verify such considerations as:

- Are the files *always* available?
- Is the storage media compatible with present hardware?
- Is the media protected against modification or deterioration?
- Are copies of the programs that process this data protected in the off-site location also? Is program documentation available?
- How current are master files that are reconstructed directly from these archives? What would be the cost of making the reconstructed files more current?
- Are operating procedures and run books available?
- Is the data stored the right data?

Master Files

Master files are those files used in day-to-day operations. Many users keep backup copies of all master files on site and update them on a daily basis. In this way, no more than one day's processing can be lost and the transactions for that day are logged in the transaction log file. The transaction log file is retained until the daily update run has been completed successfully.

Data Processed but not Distributed

If a system failure occurs during the day's processing, then data that has been processed but not distributed must be considered. Master files have been updated, but the output of the application is stored within the system and is not recoverable. The transactions cannot simply be rerun or the master files would reflect double activity. A method that can be used is to have the application programs that process the transaction file flag the header record of all orders processed. The recovery program would be designed to begin processing at the last order printed or distributed to terminals and to process all the following transaction records, but not actually update master files on flagged orders.

Data Logged but not Processed

When the recovery program has processed all the flagged orders, another category of data must be recovered: the records in the transaction file that have been received from the terminals but have not been processed. If a system failure occurs, these records must not be reentered by the terminal operators. The operators must be notified as quickly as possible not to enter more data because a recovery is in process. The recovery program must scan the transaction file and identify for the terminal operator the last record correctly entered in the transaction file. If transactions are linked together by terminal within the transaction file, the logic of the recovery program can be much more straightforward. The example in Chapter 7, *Task Chaining* illustrates this type of linkage.

Data Received but not Logged

Data that has been received but not logged must be reentered by the terminal operators. This data was in main storage at the time of the failure and is now lost.

Loss of Transaction File Data

In many instances, the transaction file is the key factor in a successful recovery from a system failure. Transaction file data that is lost or unusable will have to be reentered from terminals. If tape support is available, the effect of losing transaction file data can be reduced by running a program in the batch partition to copy the transaction file to tape on a regular basis. If tape is not available, transaction file data can be made less critical if master files are backed up on a daily basis. If a failure occurs in this case, a maximum of one day's input has to be reentered. If the master files are backed up twice a day, then a maximum of one-half day's input has to be reentered. Even orders that have been processed and printed have to be reentered so that master files can be correctly updated. The printing of the output can be bypassed for the duplicates to prevent wasting forms.

Whichever method is used, the time required to reenter records should be balanced against the time required to back up files.

File Recovery Procedures

In the event that the CCP partition is abnormally terminated, the system operator should have predefined procedures available to assist in recovering files. If none of the files used with CCP are add or output (load) files, no special procedures are needed except to start the CCP partition again. However, if add or output files are used, the file recovery programs provided with the system ($CCPRB or $RINDX) and $COPY can be used to recover the file.

$CCPRB

The $CCPRB program works only on files defined for the last CCP execution. It does not require any file OCL statements. $CCPRB designates a new file (output or load) as an existing file, and indicates that a consecutive file is full. For each consecutive file, a user-written program should be run after $CCPRB is run to search the file for the last add or output record and copy all the valid records to a temporary file. If the consecutive file is defined to CCP as an add file, the temporary file should then be copied back on top of the file CCP will use, so that the file pointers reflect the last valid added record. For an output file, the existing file should be deleted after it has been copied, so that the CCP start-up OCL will create another output file. At the end of the day the last built file should be merged with the temporary file from before the abnormal termination. Add files or direct files should be used instead of consecutive output files because less system operator intervention is required to recover files.

For indexed files, $CCPRB sorts the keys and indicates the location of the last data record entered into the file. For indexed add files, $CCPRB sorts the added keys into the existing keys.

For further information concerning $CCPRB, see *IBM System/3 Models 8, 10, and 12 CCP System Operator's Guide*, GC21-7581.

$RINDX and $COPY

For systems that are supported by $RINDX and the ACCESS and SELECT statements of $COPY, different procedures should be defined. $RINDX can be used to recover batch files and CCP files. File OCL is needed for this program. $RINDX ignores file OCL statements that do not refer to an indexed file; therefore, a copy of CCP start-up OCL could be used as input to this program. Because this program only works on existing indexed files (load to an old *empty* file also), it is suggested that an empty indexed file be defined before CCP start-up (by using $COPY or by writing a program that creates a file but does not output records to the file). If an indexed file must be created under CCP, the file should be defined as an add file in the CCP program instead of an output file. Then, when an abnormal termination does occur, $RINDX should be run (using file OCL statements) to recover the added records. CCP can then be started again and the application programs can continue to add to the existing file.

To restore consecutive add files, $COPY should be used and the location and number of records parameters in the CCP start-up OCL must be used. After an abnormal termination of the partition, this location and number of records must be used in a $COPY run to copy the file to a temporary file. A user-written program should then be run to search for valid records and copy those records back over the original file used with CCP. Then CCP can be started up again. It is suggested that the add access be used instead of the output or load access, since system design is easier and less operator intervention is required for recovery. In all cases, the files should be predefined (create a dummy or empty file) before CCP start-up so the location and size of the file is known if $COPY is needed to restore the file or parts of it.

For further information concerning $RINDX, see one of the following publications:

- *IBM System/3 Model 12 System Control Programming Reference Manual*, GC21-5130
- *IBM System/3 Model 15 System Control Programming Reference Manual*, GC21-5077
- *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual*, GC21-5162

Chapter 10. Simplified Queuing Theory

A *queue* is a waiting line or list formed by items in a system that are waiting for service. Queuing theory equations describe what happens in a system when queues develop and the resulting effects on system performance; these equations are quite complex. However, a basic knowledge of queuing theory helps the designer of a CCP system to determine system requirements and to understand the effects of queues on the performance of the system. For this reason, this chapter describes a simplified method of using queuing theory.

SIMPLIFIED QUEUING THEORY EQUATIONS

The response time for one transaction is a function of the arrival rate (A) of all like transactions and the service time (S) required to process these transactions. Response time to deposit money in a bank is a function of how many deposits must be made and how long a deposit takes. Both A and S must be in the same time units. For tele-processing messages, this is usually given in seconds. The utilization (U) of a facility by a transaction is the product of the arrival rate and service times:

$$U\ (\%) = \text{arrival rate} \times \text{service time}$$

or

$$U\ (\%) = A \times S$$

Utilization may also be defined as the ratio of the actual transactions serviced to the total transactions that could be serviced:

$$U\ (\%) = \frac{\text{actual transactions serviced}}{\text{total transactions possible}}$$

The utilization of a facility is limited to 100%. If the product of the arrival rate and the service time is greater than 100%, more than one facility is required to handle the arriving transactions.

The next step contains the simplifying assumptions which make this an estimate rather than an equation. If a facility is used, transactions wanting to use that facility will have to wait. The number of transactions waiting, or queued (Q), can be estimated to be the utilization divided by 1 minus the utilization:

$$Q = \frac{\text{utilization}}{1 - \text{utilization}}$$

or

$$Q = \frac{u}{1 - u}$$

The utilization of a facility will be between 0% and 100% or between 0 and 1. If a facility had a utilization of 75%, the Q = .75/(1 - .75), which means the queue contains on the average three transactions.

Before a transaction can use a facility, it must wait for the facility to service all transactions in the queue ahead of itself. This wait time (W) is the product of the number on the queue (Q) and the service time (S):

$$\text{Wait} = \text{queue} \times \text{service}$$

or

$$W = Q \times S$$

Finally, the response time for a transaction is equal to the wait time (W) plus its service time (S):

$$\text{Response} = \text{wait} + \text{service time}$$

or

$$R = W + S$$

If a transaction must use several facilities serially, then the total response time for that transaction is the sum of the response times for each facility.

SIMPLIFIED QUEUING THEORY EXAMPLE

The remainder of this chapter outlines how the information above can be used to study a system that includes the following applications: order entry, inventory, cash application and production inquiry.

The system requirements of an online system (for example, how much processing unit main storage is required) are a function of the number of transactions that must be handled during a user's peak hour workload. Simplified queuing theory is a method of arriving at these requirements. Doing a volume study to determine system size involves the following steps:

1. Define and flowchart the application to be done. Use transaction-oriented program steps.

2. For each program step or transaction, determine:
   a. How much key entry time.
   b. How many disk accesses.
   c. How many characters will be passed between the terminal and the processing unit.

3. Calculate the number of transactions per hour (at peak load hour, if applicable) for each application.

4. Determine the total number of transactions and the total number of characters involved in the online applications. Calculate the average number of characters that make up a transaction.

5. For each line speed, calculate the line time required to transfer an average transaction between a 3270 terminal and the processing unit.

6. Based on the total number of transactions to be handled in an hour, determine the line utilization for different line speeds.

7. Calculate line response time.

8. Based on the total number of transactions to be handled in an hour, the average number of disk seeks per transaction, and the disk seek times for different disk drives, determine disk utilization.

9. Calculate disk response time.

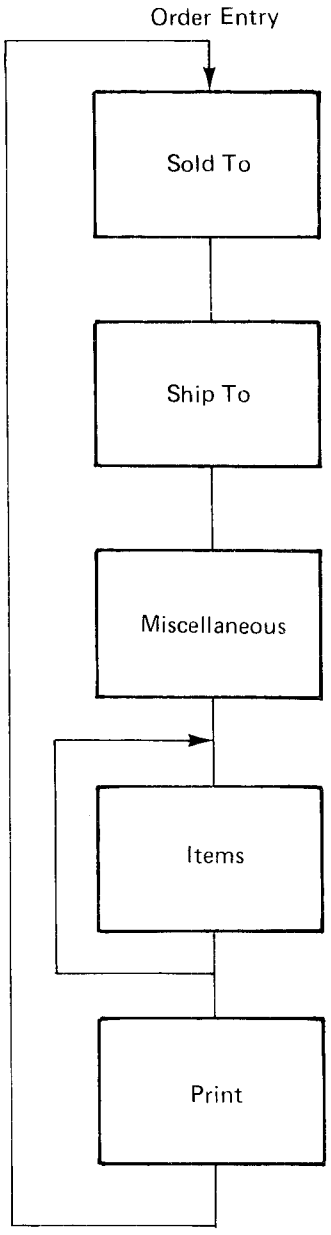
10. Based on the total number of transactions to be handled in an hour and the average processing unit service time for each transaction, determine the processing unit utilization.

11. Calculate processing unit response time.

12. Based on the total number of transactions in an hour by program and the processing unit time required to process that load by program, determine the number of user task areas needed and therefore, the required processing unit size.

Step 1. Define and Flowchart the Application

Define the applications in terms of transaction-oriented program steps. The following is a sample order entry application. The flowcharts for the other applications are shown in the descriptions for step 2.



Step 2. Determine Activity for Each Program Step

For each program step, determine:

- Key entry time (estimate, using three keystrokes per second).
- Number of characters to be sent between the terminal and the processing unit. This number should represent the total characters in and out, including both data and control characters. The number of control characters can be estimated at .20 times the number of data characters.
- Number of disk accesses.

Order Entry

Program Step 1 (Sold To): Enter and validate customer information.

- Key time = 12 seconds
- Line = 400 characters
- One disk read indexed and one disk write direct = 4 seeks

Program Step 2 (Ship To): Same as sold to program.

Program Step 3 (Miscellaneous): Enter and validate miscellaneous information.

- Key time = 20 seconds
- Line = 400 characters
- Disk (write) = 2 seeks

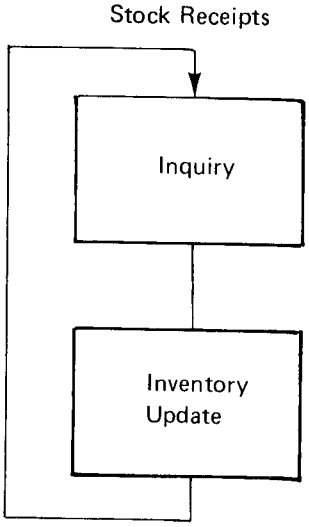
Program Step 4 (Items): Enter and validate items, and allocate inventory.

- Key time = 6 seconds
- Line = 100 characters
- Disk = 8 seeks

Program Step 5 (Print): Print invoices; sort invoices into warehouse sequence; update accounts receivable.

- Key time = 0 seconds
- Line = 1,900 characters
- Disk = 26 seeks

Inventory



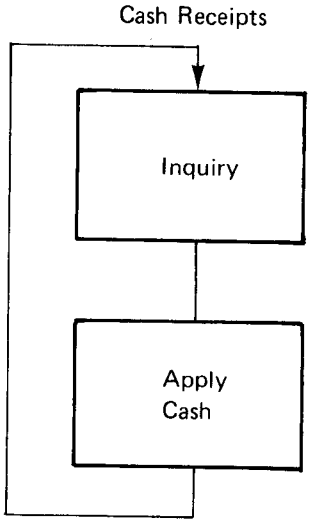
Program Step 1 (Inquiry): Retrieve vendor orders.

- Key time = 10 seconds
- Line = 1,200 characters
- Disk = 12 seeks

Program Step 2 (Inventory Update): Validate and update inventory; close orders; print warehouse routing.

- Key time = 40 seconds
- Line = 200 characters
- Disk = 30 seeks

Cash Application



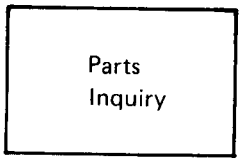
Program Step 1 (Inquiry): Retrieve open items.

- Key time = 3 seconds
- Line = 1,500 characters
- Disk = 8 seeks

Program Step 2 (Apply Cash): Edit and update open items.

- Key time = 20 seconds
- Line = 100 characters
- Disk = 12 seeks

Production Inquiry



Program Step 1 (Inquiry): Retrieve parts information.

- Key time = 3 seconds
- Line = 200 characters
- Disk = 2 seeks

Step 3. Determine Transactions per Hour for Each Online Application

Using (1) the total number of transactions required to accomplish an application, (2) the application volumes, and (3) the total hours available to handle the load, determine how many transactions must be processed in an hour to complete the job. The total for all applications represents the workload the system must be able to handle.

Example: Order Entry

- Transactions per order:

Sold to	1	
Ship to	1	
Miscellaneous	1	
Items	12	
Print	1	
	16	Transactions required to do one average order

- Volumes:

Local	1,500 orders
Remote	750 orders
Total	2,250 orders

- Orders per hour (convert to peak hourly load):

  60% of daily volume done in first 4 hours

  .60 x 2,250 orders/day ÷ 4 hours = 337.5 orders/hour

Summary Chart

The following chart summarizes the total transactions per hour for the system.

Application	Volume per Day	Transactions per Application	Total per Day	Hours to Complete	Transactions per Hour
Order entry	2,250	16	36,000	8	60% processed in 4 hours[1]
			21,600	4	5,400[1]
Stock receipts	450	2	900	8	112
Cash receipts	600	2	1,200	8	225
Production inquiry	2,250	1	2,250	8	60% processed in 4 hours[1]
			1,350	4	338
			System transaction total		6,075

[1] 60% of these transactions must be processed in a 4-hour period. Use these numbers to compute the transactions per hour.

Step 4. Calculate the Average Number of Characters per Transaction

The following chart summarizes the number of characters per transaction and the total characters per hour for each application program step.

Transactions (Peak Load) per Hour	Number of Characters on TP Line per Transaction	Total Characters per Hour
337.5 sold to	400	135,000
337.5 ship to	400	135,000
337.5 miscellaneous	400	135,000
4,050.0 items (12 x 337.5)	100	405,000
337.5 print	1,900	647,250
56.0 inventory inquiry	1,200	67,200
56.0 inventory update	200	11,200
112.5 cash inquiry	1,500	168,750
112.5 cash application	100	11,250
338.0 production inquiry	200	67,600
6,075.0		1,777,250

$$\frac{1{,}777{,}250 \text{ characters per hour}}{6{,}075 \text{ transactions per hour}} = 293 \text{ characters per transaction average (rounded)}$$

Step 5. Calculate Line Time to Transmit an Average Transaction

For each line speed, calculate the line time required to transfer an average transaction to (or from) the processing unit. This example does not consider line turnaround times. Turnaround times should be considered in actual use of queuing theory.

Line Speeds in Bits per Second (bps)	Equivalent Characters per Second (cps)	Time in Seconds to Transfer 293 Characters/Transaction
2,400	300	.98
4,800	600	.49
7,200	900	.33
9,600	1,200	.25
Direct attach[1]	5,000	.06

[1]The direct attach line speeds vary from 1,000 to 5,000 characters per second depending upon the length of the data. The shorter the message being sent to the terminal the lower the effective data rate will be.

Step 6. Calculate Line Utilization

If the transaction volume per hour of the system is known, the line speed that provides optimal utilization can be chosen.

- Transactions/hour x average/characters/transaction = characters/hour

  6,075 x 293 = 1,779,975

- Characters/hour ÷ 3,600 = characters/second

  1,779,975 ÷ 3,600 = 494

- Utilization (U) = characters/second ÷ line speed in characters per second

  or = actual used ÷ total available

- Queue = U/1 - u

Line Speeds

bps	cps	Utilization	Queue
4,800	600	.823	4.65
7,200	900	.549	1.22
9,600	1,200	.412	.70
Display adapter	5,000	.099	.11
Display adapter	1,000	.494	.98

Step 7. Calculate Line Response Time

Line response times can be calculated as follows:

Service Time

Service time is defined as the line time required to send or receive a transaction. Service time for an average transaction (293 characters—see step 4) can be calculated as follows, assuming a 4,800 bps line (600 characters/second):

Service time = 293 characters average/transaction ÷ 600 characters/second

S = .49 second line time required to send/receive 293 characters

Utilization

Assuming two balanced 4,800 bps lines (each line handling one-half of the transaction volume), line utilization can be calculated as follows:

1/2 x 494 characters/second (see step 6) = 247 characters/second

U = actual used ÷ total available

= characters/second ÷ 600 characters/second

= .41 or 41% utilization

Wait Time

Wait time is defined as the time a transaction must spend in the queue waiting to be serviced. This time is based on (1) the length of the queue; that is, how many transactions are pending, and (2) the service time; that is, the line time to send or receive a transaction.

Wait = queue x service

Q = U ÷ (1 - U)

= .41 ÷ (1 - .41)

= .70 transactions pending on the average

W = Q x S

= .70 x (293 character average ÷ 600 characters/second)

= .34 second average wait time

Line Response Time

Line response time for online application transactions (using two 4,800 bps lines) is shown in the following chart:

Program Steps	Characters/ Transaction	Actual Service	Average Wait	Average Response (Seconds)
Sold to	400	.66	.34	1.00
Ship to	400	.66	.34	1.00
Miscellaneous	400	.66	.34	1.00
Items	100	.17	.34	.51
Print	1,900	3.16	.34	3.50
Inventory inquiry	1,200	2.00	.34	2.34
Inventory update	200	.33	.34	.67
Cash inquiry	1,500	2.50	.34	2.84
Cash application	100	.17	.34	.51
Production inquiry	200	.33	.34	.67

Step 8. Calculate Disk Utilization

Disk utilization can be determined based on the number of transactions per hour, the number of disk seeks per transaction, and the disk seek time for the disk drive, as follows:

- Calculate transactions per hour and seeks per hour:

Transactions per Hour	Disk Seeks/ Transaction	Total Seeks per Hour
337.5 sold to	4	1,350
337.5 ship to	4	1,350
337.5 miscellaneous	2	1,350
4,050.0 items	8	32,400
337.5 print	26	8,775
56.0 inventory inquiry	12	672
56.0 inventory update	30	1,680
112.5 cash inquiry	8	900
112.5 cash application	12	1,350
338.0 production inquiry	2	676
6,075.0 transactions per hour system totals		50,503 seeks per hour

- Calculate seeks per transaction:

  Total seeks/hour ÷ total transactions/hour = average number of seeks/transaction

  50,503 ÷ 6,075 = 8.3

- Determine average disk seek times:

3340	Seek	25 ms
	Rotational delay	10.8 ms
		35.8 ms
5445	Seek	60 ms
	Rotational delay	12.5 ms
		72.5 ms
5444	Seek	126 ms (high speed)
	Rotational delay	20 ms
		146 ms

- Disk utilization is a function of how many seeks and how much time is required for each seek:

  Total transactions/hour x average seeks/transactions = total seeks/hour

  6,075 x 8.3 = 50,503

  Total seeks/hour ÷ 3,600 seconds/hour = total seeks/second

  50,503 ÷ 3,600 = 14.0

  Disk utilization = total seeks/second x disk time in seconds/seek (for 3340)

  U = A x S

  = 14.0 x .0358

  = .501 or 50.1% utilization

  Queue = U ÷ (1 - U)

  = .501 ÷ (1 - .501)

  = 1.0 seeks pending on the average

Step 9. Calculate Disk Response Time

- Wait time = queue x service

  W = Q x S

  = 1.0 seeks x .0358 seconds/seek

  = 0.036 seconds

- Response time = wait time + service

  R = W + S

  = .036 seconds + .0358 seconds

  = .072 seconds

- The following chart summarizes disk response times for transactions in each program step:

Programs	Seeks/ Transaction	Average Response/Seek (Seconds)	Average Total Disk Response/Transaction (Seconds)
Sold to	4	.072	.288
Ship to	4	.072	.288
Miscellaneous	2	.072	.144
Items	8	.072	.576
Print	26	.072	1.872
Inventory inquiry	12	.072	.864
Inventory update	30	.072	2.160
Cash inquiry	8	.072	.576
Cash application	12	.072	.864
Production inquiry	2	.072	.144

Step 10. Calculate Processing Unit Utilization

Processing unit utilization is a function of the average processing unit service time per transaction and the peak hour transaction volume.

- Processing unit service time:

Transactions per Peak Hour	Processing Unit Time[1] to Process per Transaction (in Seconds)	Processing Unit Time in Seconds per Hour
337.5 ship to	.1	33.75
337.5 sold to	.1	33.75
337.5 miscellaneous	.1	33.75
4,050.0 items (12 x 337.5)	.5	2,025.00
337.5 print	.8	270.00
56.0 inventory inquiry	.1	5.60
56.0 inventory update	.8	44.80
112.5 cash inquiry	.1	11.25
112.5 cash application	.8	90.00
338.0 production inquiry	.1	33.80
6,075.0 total transactions		2,581.70 seconds of processing unit time

$$\text{Average time to process a transaction} = \frac{\text{total processing unit time}}{\text{total transactions}}$$

$$= 2{,}581.7 \text{ seconds} \div 6{,}075 \text{ transactions}$$

$$= .4 \text{ seconds per transaction}$$

[1] The processing unit service time is an estimate. Programs using multiply and/or divide tend to run longer because these calculations may take up to .15 seconds to execute. Use an average of 25 microseconds to execute one line of RPG II code.

- Processing unit utilization:

  Total transactions/hour x average time/transaction = total processing unit time in seconds required per hour

  6,075 x .4 = 2,430

  Processing unit utilization = total time in seconds divided by 3,600

  = 2,430 ÷ 3,600

  = .67 or 67% utilization

  Queue = U ÷ (1 - U)

  = .67 ÷ (1 - .67)

  = 2.0 transactions pending processing

  Wait time = queue x service

  = 2.0 x .4

  = .8 seconds average wait

Step 11. Determine Response Time for Processing Unit and Total System

Average processing unit response time (service time plus wait time) for each program step is shown in the following chart:

Programs	Service Time (Seconds)	Average Wait Time (Seconds)	Average Total Processing Unit Response Time (Seconds)
Ship to	.1	.8	.9
Sold to	.1	.8	.9
Miscellaneous	.1	.8	.9
Items	.5	.8	1.3
Print	.8	.8	1.6
Inventory inquiry	.1	.8	.9
Inventory update	.8	.8	1.6
Cash inquiry	.1	.8	.9
Cash application	.8	.8	1.6
Production inquiry	.1	.8	.9

The average system response time per transaction is shown in the following chart (load and termination times are added to those response times determined previously):

	Response Time in Seconds									
Program	Line	+	Disk	+	Processing Unit	+	Load[1]	+	Termination	= Total System Response Time
Sold to	1.00		.288		.9		1.0		.5	3.688
Ship to	1.00		.288		.9		1.0		.5	3.688
Miscellaneous	1.00		.144		.9		1.0		.5	3.544
Items (MRT)	.51		.576		1.3		.1[2]		.05[2]	2.536
Print	3.50		1.872		1.6		1.0		.5	8.472
Inventory inquiry	2.34		.864		.9		1.0		.5	5.604
Inventory update	.67		2.160		1.6		1.0		.5	5.930
Cash inquiry	2.84		.576		.9		1.0		.5	5.816
Cash application	.51		.864		1.6		1.0		.5	4.474
Production inquiry	.67		.144		.9		1.0		.5	3.214

[1] Load time for SRT programs on 3340 disk systems is estimated as 1.0 second (for 5444s it would be 2.0 and 1.0 respectively); termination time is estimated as .5 seconds.

[2] Attachment time to a resident MRT program is estimated as .1 seconds; release time is estimated as .05 seconds.

Step 12. Determine System Size

The size of the processing unit required is based on how many user task areas are needed to handle the programs. When a user program is resident in main storage, it occupies a tasking area. Each tasking area can be utilized 0% to 100%. To compute the utilization of a tasking area for a specific program, use the formula:

$$\frac{\text{transactions per hour x system response}}{\text{3,600 seconds per hour}} = \text{utilization}$$

Computing the utilization for each program using this formula yields the following results:

Programs	Transactions per Hour	Average System Response (Seconds)	Program Residency per Hour (in Seconds)	Utilization of a Tasking Area	
Sold to	337.5	3.688	1,245	.35	
Ship to	337.5	3.688	1,245	.35	
Miscellaneous	337.5	3.544	1,196	.33	
Items (MRT)	4,050.0	2.536	10,271	2.85	
Print	337.5	8.472	2,859	.79	
Inventory inquiry	56.0	5.604	314	.09	
Inventory update	56.0	5.930	332	.09	
Cash inquiry	112.5	5.816	654	.18	
Cash application	112.5	4.474	503	.14	
Production inquiry	338.0	3.214	1,086	.30	
				5.47	Total

A simple way to determine the minimum number of areas required is to add the utilization for all programs and round up to the nearest whole number:

Number of user task areas required = 5.47

(or rounded up) = 6

Programs utilizing a task area less than 100% can share a task area with other programs that do not require a dedicated task area (less than 100% utilization). When the task area utilization computation results in greater than 100% utilization (*Items* in this analysis has a utilization of 285%), multiple task areas with duplicate copies of the program are needed. In this case, three copies of *Items* are required to handle the transaction volumes.

The user task areas could be utilized as follows:

	MRT	MRT	MRT	SRT	SRT	SRT
	Items 1	Items 2	Items 3	• Sold To • Miscellaneous • Inventory Inquiry • Cash Application	• Ship To • Production Inquiry • Inventory Update • Cash Inquiry	Print
Utilization:	95%	95%	95%	91%	92%	79%

If a task size of 14K bytes for each program is assumed, this analysis would indicate a system that would have a minimum of 84K bytes available for user tasks. The total utilizations for each task area are high and would not allow for growth in transaction volumes or for additional applications. A larger system should be considered in this case.

Your IBM representative has additional design and performance analysis facilities. Contact your representative for a more complete and detailed analysis.

Chapter 11. Performance Tips

This chapter contains specific tips and techniques that can be used to improve the performance of a CCP system.

CCP-ASSOCIATED BUFFERS

All CCP input and output data passes through several buffer areas. There are four buffer areas associated with a user task:

- The user record area described with the file
- The output hold area used by the display format facility (DFF)[1]
- The TP buffer (TPBUF) specified at startup[1]
- The line buffers

The lengths of these areas can affect the performance of a CCP system and, in some cases, changing the length of an area can improve performance. Each of these areas is described in the following paragraphs to aid the user in determining the best length to use for these areas to achieve maximum performance.

User Record Area

An input/output area is reserved for each file in a program. The length of this area is determined by the record length of a file and is specified in the user program. This area is part of the user program after compilation. Output data is placed in this area by the user program; input data is placed in this area by an input operation.

Output Hold Area

The output hold area is allocated by CCP only if the terminals in the assignment set use DFF (DFF3270-YES specified on the TERMATTR assignment statement). CCP uses this area to merge output text and user data for all DFF output operations.

On System/3 Models 4, 8, 10, or 12, CCP allocates a hold area for each BSCA line (maximum of two areas). The hold areas are appended to the DFF module that executes in the user program area.

On a Model 15, the hold area is dynamically allocated by CCP. Program 5704-SC1 uses a location in the TP buffer for the hold area. Program 5704-SC2 uses an area in either the TP buffer or an optional 2K DFF buffer (one for each BSC line), as specified by the DFFBUF parameter of the BSCALINE assignment statement. The hold area for a Model 15 is used only for user DFF put operations (see the paragraphs that describe the TP buffer use for the Model 15 and 15D in this section for additional information).

The length of the output hold area is specified on the BLKL (block length) parameter of the TERMATTR assignment statement. For best performance, the size of the area should be large enough to hold the largest output display format in this assignment set. However, on the Model 15, if the total buffer area is too small to handle the traffic, decreasing the size of the put area should help improve total throughput. To determine the BLKL value, find the length of the largest output display format (the sizes are printed by the display format generator routine) and round that size up to the next .25K. If multiple TERMATTR statements are specified, the largest BLKL value is used for the output hold area size.

Specifying a value (512 minimum) smaller than the length of the largest output display format causes the format to be sent to the terminal in blocks. Blocking usually causes the screen to blink as each block of text is displayed on the terminal. The advantage to blocking is that part of the format can be displayed on a terminal in about half the time required to display an entire (unblocked) message. The total time to display the entire format is about the same for both blocked and unblocked formats.

If the majority of the formats require about 500 bytes and only a few require 1,000 to 1,500 bytes, then a BLKL value of 512 should be specified to more efficiently use the TP buffer. As an alternative, the large formats could be divided into smaller formats. On the Model 15D (5704-SC2) with DFF buffer support specified, the hold area is 2,048 bytes (2K) for each BSC line that supports DFF buffers.

[1]On the Model 15, the TP buffer functions as the output hold area.

TP (Teleprocessing) Buffer

The TP buffer is an area of main storage used by CCP tasks as a temporary buffer to hold the parameter list and input or output data. The minimum size of the TP buffer is specified on the MINTPBUF parameter of the SYSTEM assignment statement.

TP Buffer for Models 4, 8, 10, and 12

The TP buffer for Models 4, 8, 10, and 12 is logically one area as shown in Figures 17 and 18.

Output Operations: For output operations (Figure 17), this area is used only for put-no-wait operations (for DFF put operations, the output hold area is used; for non-DFF put wait operations, the data is moved directly from the user record area to the line buffer). When a put-no-wait operation is specified, the parameter list and output data stream are moved from the user area (I/O area in the user program) to the TP buffer until the line buffer for that operation is freed. If the TP buffer does not have sufficient space to contain the parameter list and data, the put-no-wait operation is handled as a put wait operation.

Input Operations: For input operations (Figure 18), the TP buffer receives terminal and console data from the line buffer(s). For invite input and DFF get operations, CCP first examines the input parameter lists to determine the largest required buffer space. If sufficient space is available in the buffer, the space is allocated and CCP proceeds to poll the terminals for data. If data is received from one terminal and data is to be invited from other terminals, another area in the TP buffer is allocated before polling the other terminals for input.

If CCP determines that sufficient TP buffer space is not available, CCP waits until space is released before polling the terminals. Buffer space is released after an accept input operation moves data from the TP buffer to the user I/O area.

The TP buffer is also used to store commands and program requests received from a terminal. The space allocated from the TP buffer before polling a command terminal includes space for the largest program characteristics table (PCT) entry and the amount of data expected from a terminal. If the space allocated for the input data is more than that required for the data received, the excess space is freed. Polling is not initiated on a line until sufficient TP buffer space is available.

The calculations for determining the minimum value for the TP buffer are described in *IBM System/3 Models 8, 10, and 12 CCP System Reference Manual*, GC21-7588. An estimated operating size for the TP buffer can be calculated using the formula given below. The optimal value can then be determined by varying the size of the TP buffer until optimal performance is achieved. The operating value for MINTPBUF is:

MINTPBUF value = 1.2 x ([T + 1] x L)

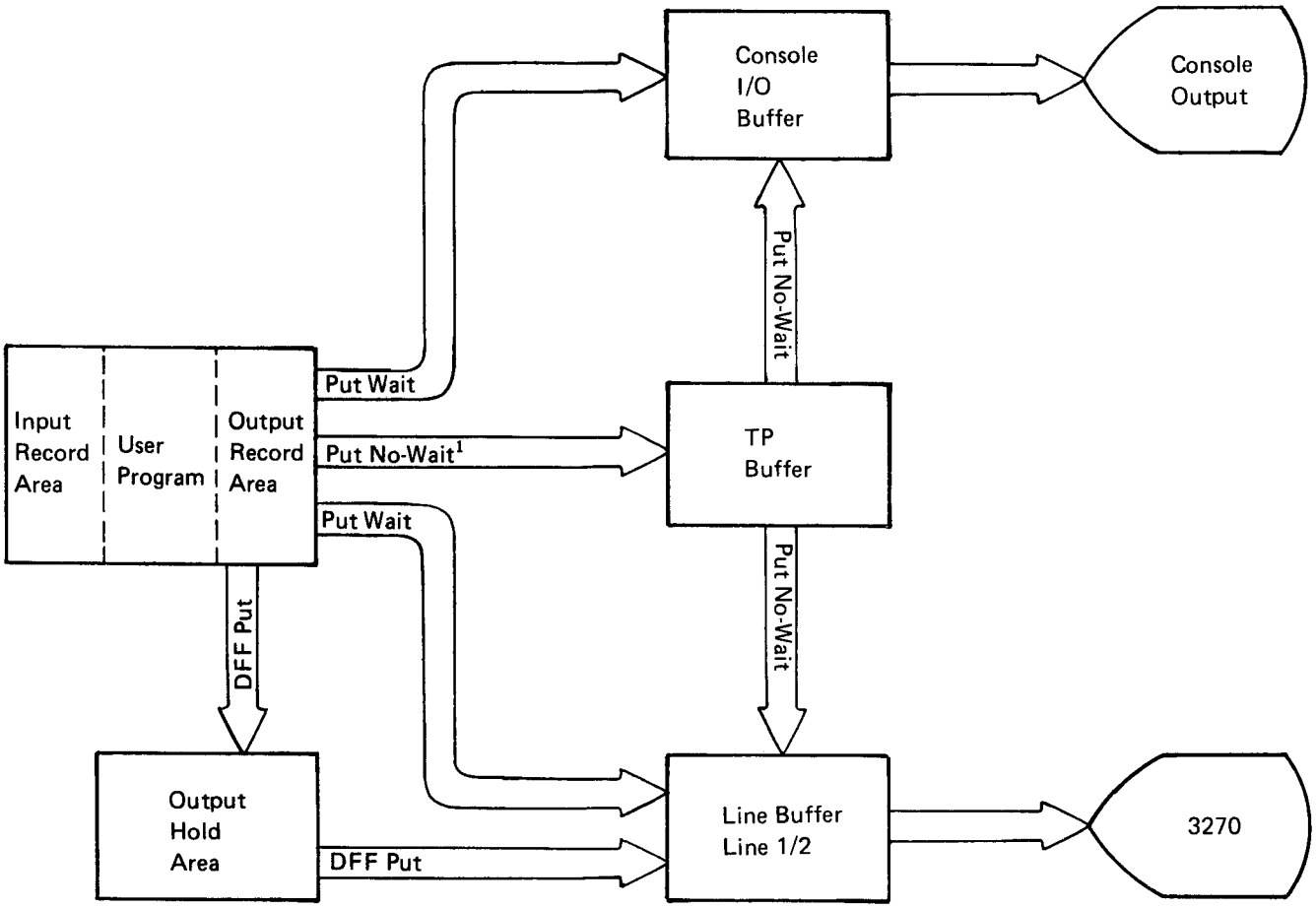
where:

T is the number of tasks generated in your CCP system.

L is the average length of the following two values:

- The maximum length of text to be put by the user programs
- The maximum length of text to be received by the user programs

The value for MINTPBUF is always rounded up to the next .25K boundary at startup; therefore, the value for the buffer area calculated at startup is always equal to or larger than the value specified at assignment or startup.



[1] If sufficient room does not exist in the TP buffer, put no-wait operations are changed to put wait operations.

Figure 17. TP Buffer Usage for Output Operations on the System/3 Models 4, 8, 10, and 12

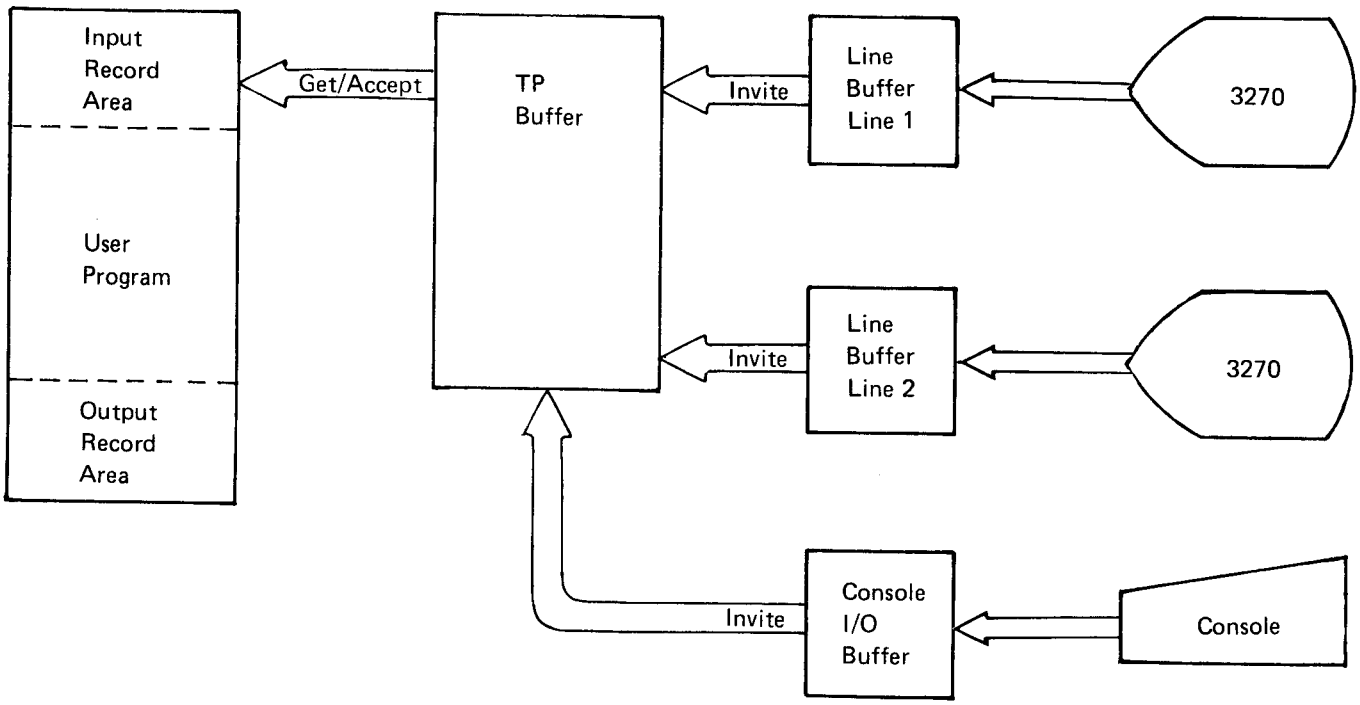


Figure 18. TP Buffer Usage for Input Operations on the System/3 Models 4, 8, 10, and 12

TP Buffer for the Model 15

The following section describes the TP buffer for Models 15 (5704-SC1) and 15D (5704-SC2) not using DFF buffer support.

The TP buffer for a Model 15 is logically separated into three areas as shown in Figure 19:

- Put data area: Used for put parameter lists and for output data.
- Put/get area: Used for put parameter lists and get or put data.
- Invite parameter list area: Used for invite parameter lists.

At CCP startup, the TP buffer is automatically allocated into the three areas. The following steps show the calculation of the lengths of these areas:

- Put data area: The length of this area is determined as follows:
  - If all terminal attributes in this assignment set are DFF, then the area is 516 bytes long.
  - If all terminal attributes in this assignment set are non-DFF, then the length is the greater of MAXRECL + 4 or BLKL + 23.
  - If the terminal attributes in this assignment set are both DFF and non-DFF, then the length is the greater of the two preceding values.
- Put/get area: CCP requires as a *minimum* that this area be large enough to handle the largest system-initiated invite input or program request from the system operator console.

  The length of the largest system-initiated invite input is the largest of the following values:
  - PGMREQL + PCT + 4

    PGMREQL is the length of the longest possible program request as specified on the SYSTEM assignment statement. The size of the largest PCT is calculated using the following formula:

    PCT = 34 + (4 x NF) + (2 x NT)

    NF is the number of disk files used by the program.

    NT is the number of terminals specified on the TERMS parameter of the PROGRAM assignment statement.
  - PRUFLNG + PCT + 11

    PRUFLNG is the largest PRUFLNG value specified on a PROGRAM assignment statement. PCT is calculated the same as in the preceding formula.
  - COMMANDL + 4

    COMMANDL is the value specified on the SYSTEM assignment statement.

  The area required for program requests from the system operator console is calculated as follows:

  104 + PCT

  PCT is calculated the same as in the preceding formula.

  The above calculation is the *minimum* value accepted by CCP for the put/get area. The estimated operating size can be calculated using the formula given below. The optimal value can then be determined by varying the size of the put/get area until optimal performance is achieved. The estimated operating size is:

  Size = 1.2 x ((N + 1) x L)

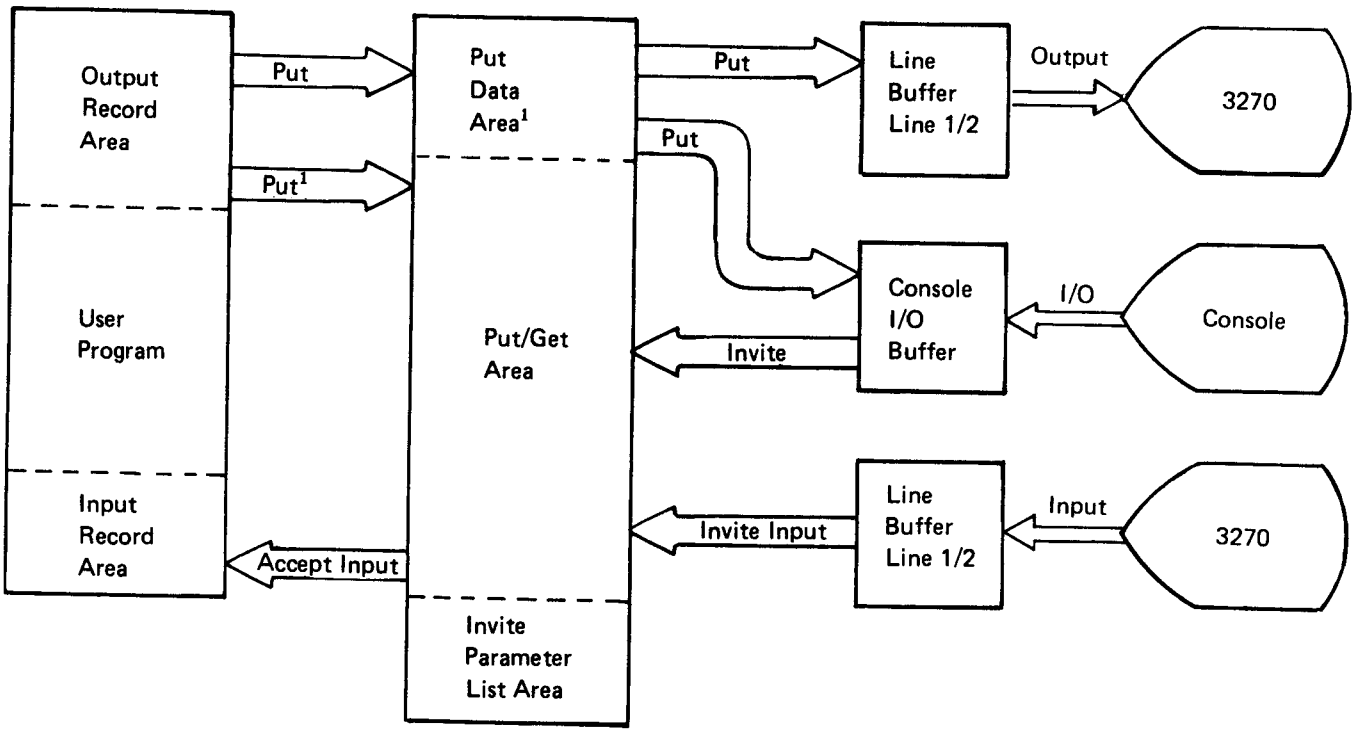
  where:

  N is the number of CCP tasks generated into the DSM system. If there is the probability that many MRT programs will be executing concurrently, increasing this number by one for each MRT may yield improved throughput.

  L is the *average* of (1) the minimum put/get area size calculated above and (2) the average length of text expected to be sent to or returned from a terminal.
- Invite parameter list area: The length of this area is calculated as follows:

  (N x 23) + 4 for 5704-SC1
  (N x 20) + 4 for 5704-SC2

  N is the number of input capable terminals.



[1] If on put operations the put-data area is not available, the output data is placed in the put/get area.

Figure 19. TP Buffer Usage for Input/Output Operations on the System/3 Model 15 or Model 15D without DFF Buffer Support

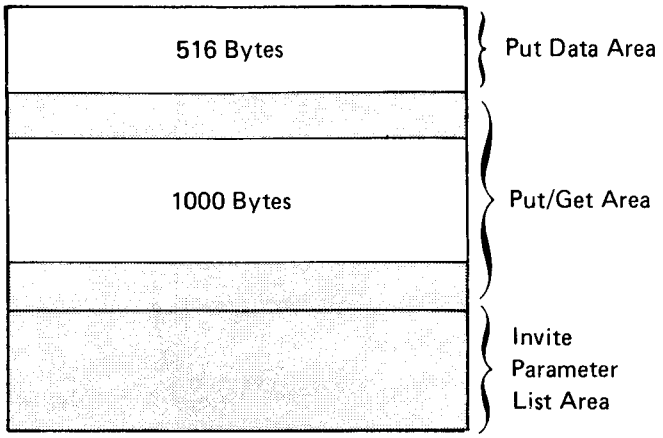
Remember that the *total* TP buffer size is obtained by CCP from the MINTPBUF keyword on the SYSTEM assignment statement. Thus, the values calculated for put area, put/get area, and the invite parameter list area should be added together and the result should be used for this keyword.

All input data must pass through the TP buffer; that is, the data is moved from the line buffer into the put/get area of the TP buffer. The input data remains in the TP buffer until the task for which it is destined indicates that the area is free. User tasks free this area by issuing an accept input operation. The amount of TP buffer needed by the input data depends upon the status of the terminal. For example, a terminal in command, non-PRUF mode requires less TP buffer (PCT + program data) than a terminal in command, PRUF mode (PCT + format data).

The Model 15 does not have separate output hold areas; the put and put/get areas function as the output hold area for all lines. The put data area of the TP buffer insures only that eventually an area will be free such that a put operation can be performed. The BLKL value given should be large enough to handle the largest put operation to assure the immediate handling of an output operation. If the TP buffer is only large enough for one put operation at a time, and two lines are being used, system performance is degraded.

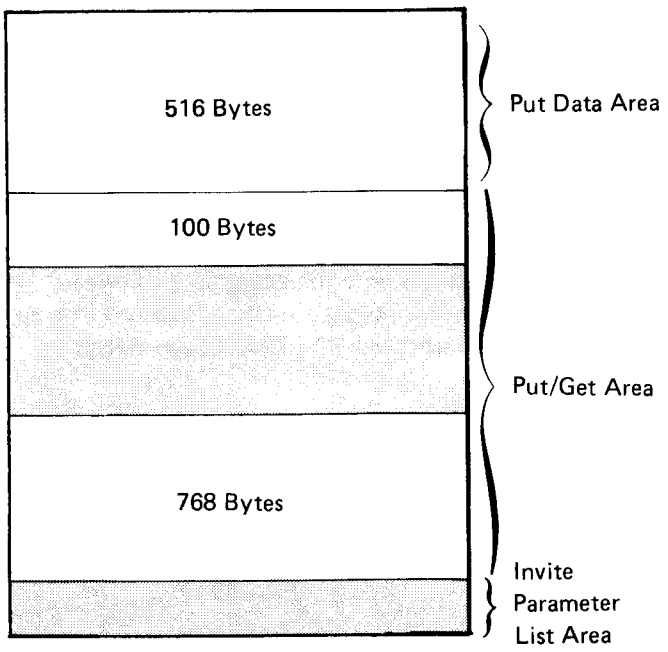
For *DFF put* operations, CCP attempts to get main storage equal to the output text length of the DFF format. Assuming the output text is larger than the area available, CCP again requests main storage using the previous length (rounded up to a multiple of 256 bytes) minus 256 bytes. If the request is successful on the second or succeeding try, the output text is blocked. If the length is reduced to 512 bytes and the request still fails, the program requesting the put operation is placed in a wait state until main storage becomes available. (This usually means that the put area is currently being used and will be available soon.)

For example, a user program issues a DFF put with a length of 950 bytes. The buffer is allocated as follows (areas not shaded are available):



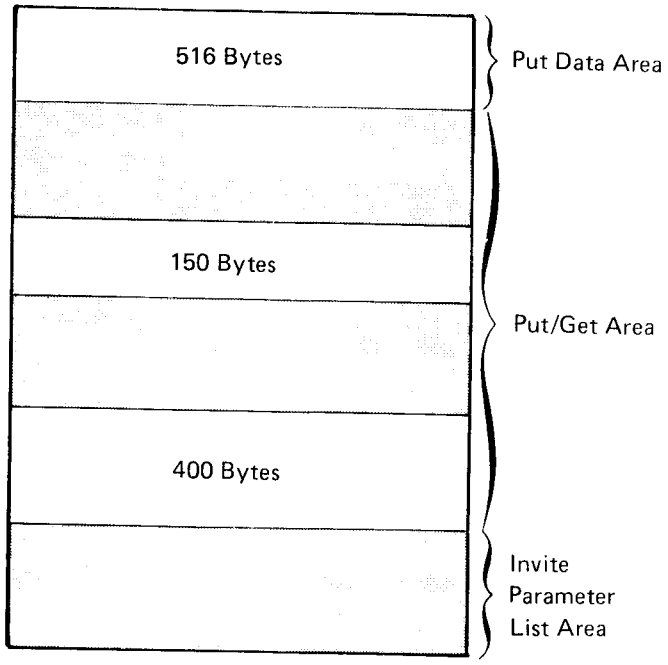
The total output text (950 bytes) does not fit into the put data area. The put/get area does contain sufficient space for the output text and the put operation is performed.

Suppose the buffer is allocated as follows:



In this case, the buffer is fragmented in such a way that there is not sufficient buffer space to contain the entire text, and the text is blocked using the 768 bytes in the put/get area.

Suppose the buffer is allocated as follows:



In this case, the largest area available for output text is in the put data area. The output text is sent in blocks using the put data area.

For *non-DFF put* operations, a get storage request is performed using the length of the output text. If the output text does not fit in the available area, the program requesting the put operation is placed in a wait state until sufficient TP buffer becomes available.

The number of terminals is also a factor in determining the size of the TP buffer: the size of the invite parameter list area is calculated at startup time using the number of input capable terminals. The more terminals specified, the larger the invite parameter list area will be. Besides this area, the number of terminals affects the overall size of the TP buffer. For example, assume the put/get area is large enough to hold only two input messages, there are more than two input-capable terminals on the system, and the probabilities are such (due to transaction arrival rate) that two transactions will always be pending (which is to say the put/get area of the TP buffer will always be in use); in this case, all output will be funneled through the put area of the buffer. System performance will be degraded because output requests are processed serially rather than concurrently.

If, in this example, two BSC lines are used, both may not be active at the same time because of the funneling effect. Better performance may be achieved in this case by specifying a size for the TP buffer large enough to handle both the maximum number of concurrent input messages and two maximum output data streams. This would allow put requests for each BSC line to be operated on concurrently.

As a minimum value, if two BSC lines are used, the TP buffer should be large enough to hold twice the maximum input data stream. If the TP buffer does not have enough available space for two input data streams, then only one line is polled.

On the Model 15D (5704-SC2), when the system operator issues a display terminals command or a secondary display user's command, the word WAIT will appear on the terminal name line if at that point in time a program operation is waiting for the TP buffer in order to complete. The wait indication on the display does not necessarily mean that a performance problem exists regarding the TP buffer. However, if the frequency of wait indications increases as terminal response times increase, the TP buffer size could be the limiting resource causing the longer response times.

TP Buffer for the Model 15D (5704-SC2) with DFF Buffer Support

The TP buffer for a Model 15D with DFF buffer support is logically separated into the areas shown in Figure 19.1:

- Put data area: Used for put parameter lists, non-DFF output data, and DFF output data for BSC lines without a DFF buffer.
- Put/get area: Used for put parameter lists, get data, non-DFF output data, and DFF output data for BSC lines without a DFF buffer.
- Invite parameter list area.
- DFF buffers (one for each BSC line with DFF buffer support): Used for DFF output data.

  Note: The DFF buffers are not included in the TP buffer size.

At CCP startup, the TP buffer and the DFF buffers are automatically allocated. The following steps show the calculation of the lengths of these areas:

- Put data area: The length of this area is determined as follows:
  - If all terminal attributes in this assignment set are DFF, then the area is 516 bytes long.
  - If all terminal attributes in this assignment set are non-DFF, then the length is the greater of MAXRECL + 4 or BLKL + 23.
  - If the terminal attributes in this assignment set are both DFF and non-DFF, then the length is the greater of the two preceding values.
- Put/get area: CCP requires as a *minimum* that this area be large enough to handle the largest system-initiated invite input or program request from the system operator console.

  The length of the largest system-initiated invite input is the largest of the following values:

  - PGMREQL + PCT + 4

    PGMREQL is the length of the longest possible program request as specified on the SYSTEM assignment statement. The size of the largest PCT is calculated using the following formula:

    PCT = 34 + (4 x NF) + (2 x NT)

    NF is the number of disk files used by the program.

    NT is the number of terminals specified on the TERMS parameter of the PROGRAM assignment statement.

  - PRUFLNG + PCT + 11

    PRUFLNG is the largest PRUFLNG value specified on a PROGRAM assignment statement. PCT is calculated the same as in the preceding formula.

  - COMMANDL + 4

    COMMANDL is the value specified on the SYSTEM assignment statement.

The area required for program requests from the system operator console is calculated as follows:

104 + PCT

PCT is calculated the same as in the preceding formula.

The above calculation is the *minimum* value accepted by CCP for the put/get area. The estimated operating size can be calculated using the formula given below. The optimal value can then be determined by varying the size of the put/get area until optimal performance is achieved. The estimated operating size is:

Size = 1.2 x ((N + 1) x L)

where:

N is the number of CCP tasks generated into the DSM system. If there is the probability that many MRT programs will be executing concurrently, increasing this number by one for each MRT may yield improved throughput.

L is the *average* of (1) the minimum put/get area size calculated above and (2) the average length of text expected to be sent to or returned from a terminal.

- Invite parameter list area: The length of this area is calculated as follows:

  (N x 20) + 4

  N is the number of input capable terminals.

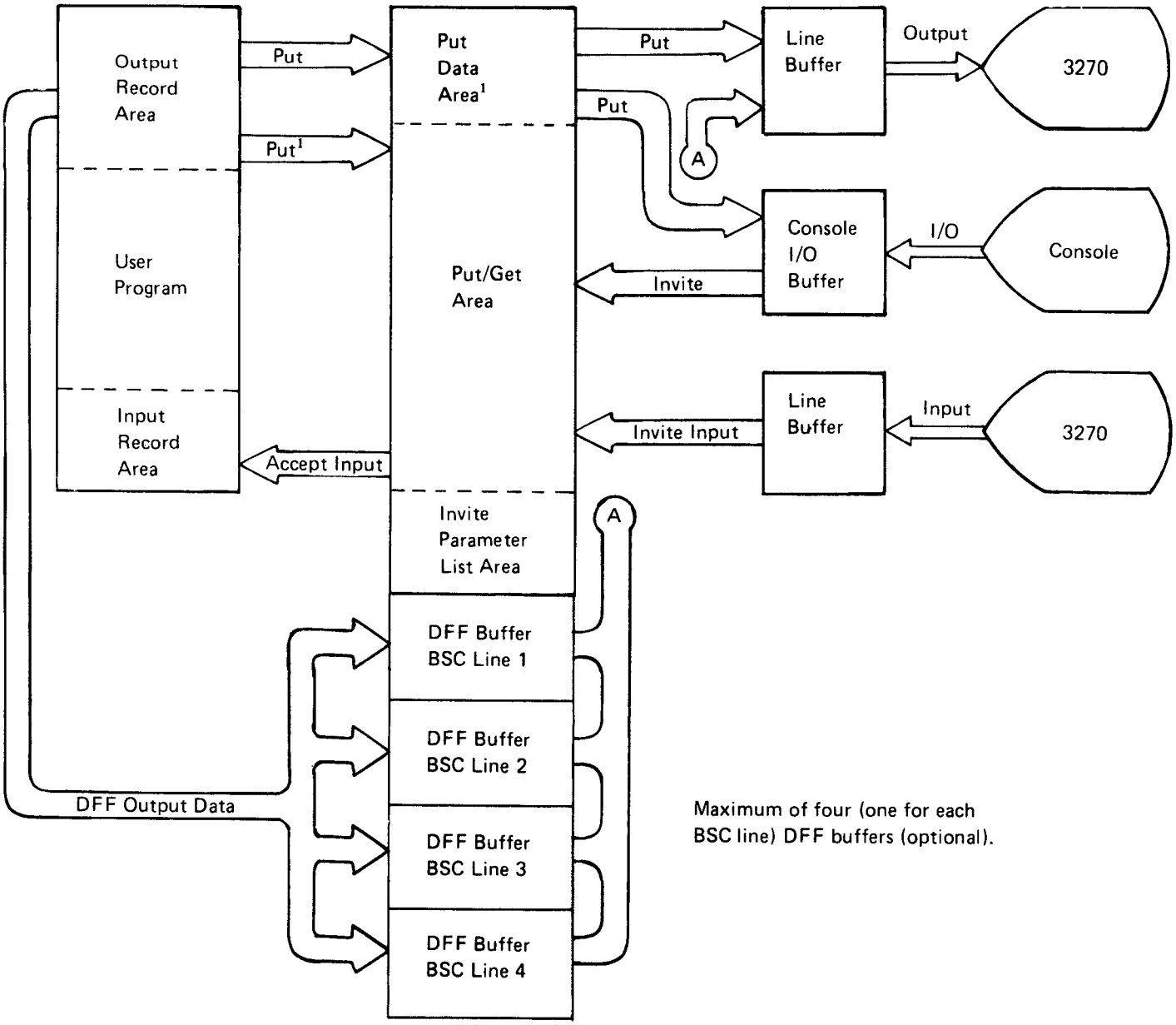
- DFF buffer: The length of each allocated buffer is 2,048 bytes (2K).

DFF output for any BSC line with an optional DFF buffer is processed through that buffer. Each buffer can handle a DFF format that is up to 2,048 bytes long if the line buffer is large enough to contain the entire format (see *Line Buffer* in this section for information concerning the line buffer size). If a DFF format is longer than 2,048 bytes or the line buffer is smaller than the length of the longest format, the format is transmitted in blocks.

Specifying DFFBUF may enhance the performance if TP buffer utilization is at its maximum.

For maximum performance when using DFF buffer support, specify a DFF buffer for each BSC line used for a DFF put operation, and specify a line buffer size that is large enough to handle the longest format (DFF or non-DFF) expected. For example, if the longest DFF format is 512 bytes (or less), some improvement in performance may be possible if two or more BSC lines use a DFF buffer. This improvement is obtained because two or more DFF put operations can be scheduled and these operations can be in various stages of transmission at any given time.

Each DFF buffer allocated reduces the user program area (UPA) by 2K bytes.



[1] If on put operations the put data area is not available, the output data is placed in the put/get area.

Figure 19.1 TP Buffer Usage for Input/Output Operations on the System/3 Model 15D (5704-SC2) with Optional DFF Buffer Support

Line Buffer

All input and output text passes through the line buffer. There is a line buffer associated with each line. The size of the line buffer is determined by the BLKL value (rounded up to the next .25K boundary) of the TERMATTR assignment statement associated with this line.

This space is doubled if DBLBUF-YES is specified on the BSCALINE assignment statement or if ASCII transmission code is specified (see note). Double buffering may reduce data transmission time in a multiple block transmission environment (for example, if a 3270 sends data to the computer in blocks of 256 characters or less). However, double buffering uses more storage than single buffering.

The size of the line buffer is determined by the lengths of the input text and the output text. For 3270 terminals only 256 is needed for input, but any size could be needed for output text. Thus, if the system has limited main storage available, it is best to specify a single buffer of 512 bytes. To improve response time, specify a larger line buffer so that fewer of the output messages need to be blocked. If further improvement in response time is desired and the system has sufficient main storage, double buffering can be specified.

Note: If ASCII is specified, an additional buffer, equal to the line buffer, is allocated for translation of the output text from EBCDIC to ASCII.

Figure 20 shows the interaction of the line buffer, TP buffer, and the user area for CCP operations.

CCP TASK SIZES

In addition to specifying the size of the TP buffer, the user must specify a minimum size for an area to load and execute user tasks. This area is called the minimum user program area (MINUPA) and is specified on the SYSTEM assignment statement. The size of this area depends on the size of the user tasks that are being executed. The storage requirements of each user task are calculated using the following elements:

- The size of the object program
- The size of the largest DFF field descriptor table (FDT) used by the user task
- The number of terminals and formats used by the task

The last two elements form the program appended storage (PAS) for DFF. The minimum size for each element is .25K.

The TASKSIZE parameter (5704-SC2 only) should be considered for calculating user program storage requirements.

The size of the object program is obtained from the compiler listing. The size of the FDT is obtained from the display format generator listing. The amount of storage required for terminal information and format information is calculated as follows:

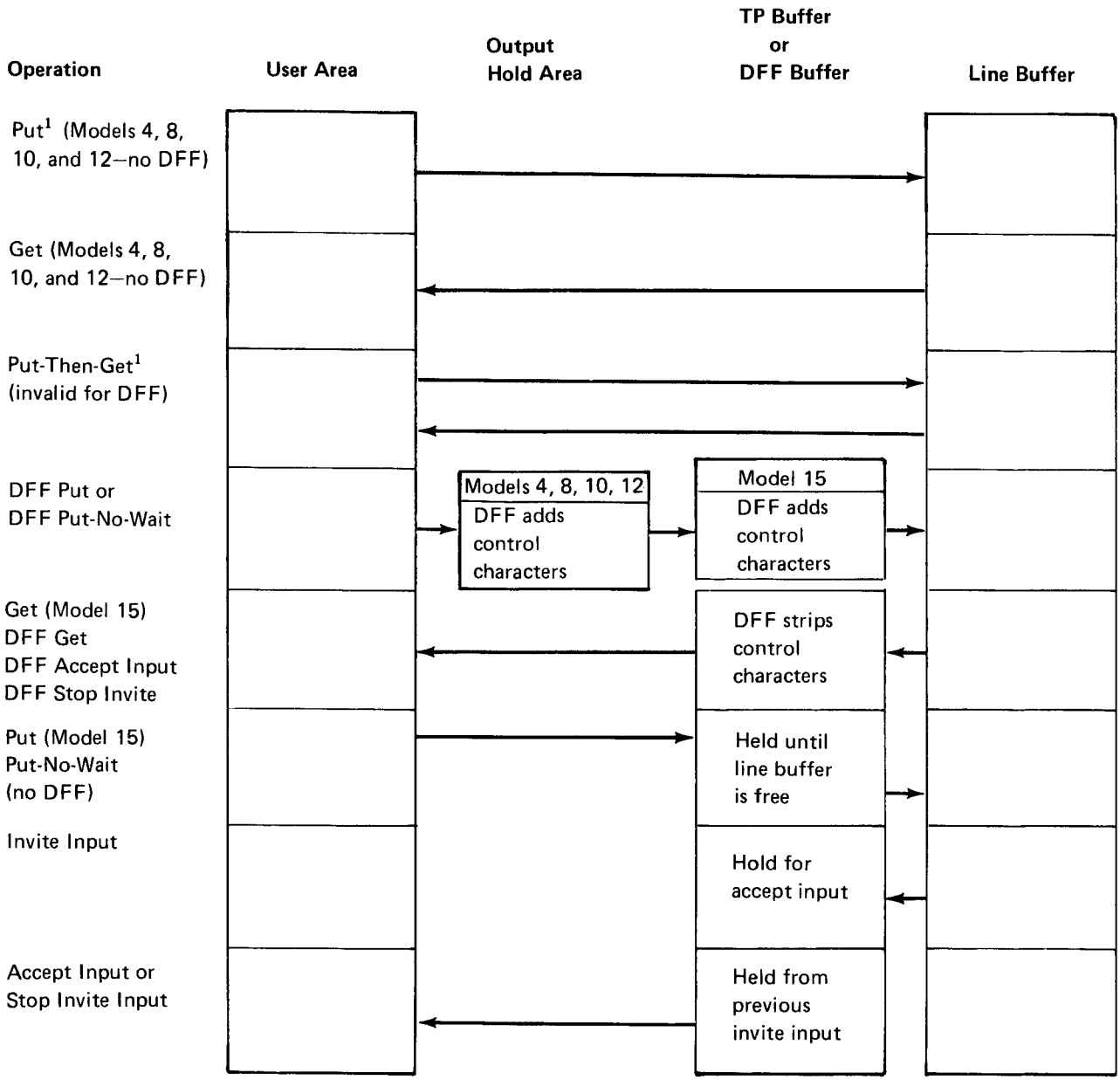
Storage required = 127 + (37 x NT) + (18 x NF)

where:

NT is the number of terminals

NF is the number of formats

Once the sizes of the elements are known, the size of each is rounded up to the next .25K boundary. The sizes of the elements are then added together to form the storage requirements of the user task. On a Model 15, this total is rounded up to the next 2K boundary.



[1] For console operations, console buffer is used instead of the line buffer.

Figure 20. Buffer Interaction

Minimizing Storage Requirements

Knowing how each element is developed allows some judgment to be made on where storage requirements can be reduced, especially if one of the elements is just over a .25K boundary. Some techniques are to:

- Define fields with a type F output field class. This reduces the FDT by 14 bytes for each field defined as type F, but the fields defined with this classification cannot be modified using a put override operation.
- Combine two fields into one. This eliminates an FDT entry.
- Redesign a format as two formats. This reduces the size of the FDT, but input data can be received only from the format last sent to a terminal.
- Define each line of data as a unique format. This reduces storage requirements, but this method increases line and disk activity.
- Use the most efficient program design (MRT, SRT, etc) for the application. This affects storage utilization (see Chapter 2 for additional information).

Techniques for reducing RPG II program size are given in the *RPG II Reference Manual*, SC21-7504.

Model 12 Examples

For example, a task to be executed on a Model 12 has an object program size of 7,158 bytes (7.0K or 7,168 bytes, rounded to the next .25K), an FDT size of 140 bytes (256 bytes rounded), and two terminals with one format for a size of 219 bytes (256 bytes rounded). The total storage required for this user task is 7.5K bytes. The FDT and terminal/format requirements for this task are already at minimum sizes (.25K each); the only way to reduce the storage requirements for this task would be to reduce the size of the object program by at least 246 bytes, to 6.75K.

If another Model 12 task has an object program size of 5,510 bytes (5.5K rounded), an FDT size of 284 bytes (.5K rounded), two terminals and one format for a size of 219 bytes (.25K rounded), storage requirements can be reduced by decreasing the size of the object program and the FDT. Reducing the object program by 134 bytes reduces the storage requirement by 256 bytes. Reducing the FDT by 28 bytes reduces the storage requirement by another 256 bytes.

Model 15 Example

On a Model 15, the total storage requirement for a user task is rounded up to the next 2K boundary. Here it is even more important to take a closer look at the elements in a task for storage saving techniques. For example, a Model 15 task has an object program size of 6,236 bytes (6.25K rounded), an FDT size of 154 bytes (.25K rounded), and two terminals and three formats for a size of 255 bytes (.25K rounded). The total storage for this task is 6.75K, rounded to the next 2K boundary is 8.0K. Removing 768 bytes (.75K) from the object program (all other elements are already at minimum) reduces the total user task storage requirements by 2,048 bytes (2K).

DFF CONSIDERATIONS

Figure 21 shows the disk accesses for DFF operations. The table indicates the following considerations:

- If only one format is used in a program, there are fewer accesses for the succeeding uses of the format (only one disk read from the format index in CCPFILE and only one read of the FDT from the library).
- The proximity of CCPFILE to the object library affects response time. This is important not only for reading the format index, but for every program load (DFF or non-DFF) since the PCT must be read for each program load.
- Processing transients is faster than reading from the object library since the disk head is more likely over the transient area. Thus a put override operation is faster than a put operation.
- There are 42 format entries in each sector of the format index. The index is read into the FDT area, which is part of the storage area that is appended to user programs for DFF operations. Thus the size of the FDT area determines how much of the index is read in at one time. The index is in the order in which the format directory entries exist in the library. Thus, if frequently used entries are loaded into the library first, they will be found faster and executed faster.
- Unused formats should be deleted from the library because they may cause unnecessary reads of the format index.

Operation	Item Accessed	Number of Accesses	Area Accessed
Put	Read format index③	1⑥	CCPFILE
	Read FDT from disk②	1	Object library
	Read 3270 text [1]	1④	Object library
Put override	Read FDT②	1	Object library
	Build text①	2	Transient⑤
Copy	Build text①	1	Transient⑤
Accept input	Read format index (PRUF)③	1	CCPFILE
	Read FDT from disk②	1	Object library
Invite input	No disk accesses		

[1] Always done.
[2] Read only if last format used is not the same as the current format.
[3] Read only for the first time format is used in the program.
[4] One access if not blocking, or one access for each block, if blocking.
[5] These reads are in addition to those required to support the operation.
[6] Additional reads required if index is not in the first read. The index is read into the FDT area.

Figure 21. Disk Accesses for DFF Operations

CCP DISK ACCESSES

Figure 22 shows the number of disk accesses for program loads and terminations. It is usually slightly faster for the user task to release a requesting terminal on a Model 15A, B, and C, and then execute termination than to allow CCP termination routines to release the terminal (seven transient loads rather than eight). On the Model 15D, however, it is faster to allow CCP termination to release the terminal (two transient loads rather than five).

Following is a listing of the number of transient loads for certain CCP operations:

Operation	Number of Transients
Get attributes	1
Acquire terminal	3
Acquire/set attributes	4
Release terminal	3
Release/keep	4
Accept input (nonresident)	1
Stop invite	3
Task chain	2

For example, an SRT program that has disk files and releases the requesting terminal requires the following number of accesses, assuming no performance options:

	Models 4, 8, 10, and 12	Models 15A, B, and C	Model 15D
Load:			
Transients	20	14	14
Other		1	1
$CCPFILE	1	1	1
Object library	1	1	1
Terminate	12	12	7
Total	34	29	24

Operation	Program	Number of Disk Accesses		
		Models 4, 8, 10, and 12	Models 15 A, B, and C	Model 15D
Program load	MRT (already loaded)	5	3	3(6)
	MRT or SRT (not in storage)			
	Program request and allocation	11	5	5(7)
	If disk	1	—	—
	If required terminals	1	—	—
	If unit record devices	1	1	1
	File allocate (each chain of DTFs)	1	1	1
	Open (each chain of DTFs):			
	Disk(1)(5)	7	8	8(9), (10), (11)
	Printer(1)	4	4	4(9)
	Card device(1)	5	5	5(9)
	Disk and printer(1)	9	9	9(9)
	Disk, printer and card device(1)	10	10	10(9)
	Disk (FORTRAN files)(2)	4	3	3(9)
	PTAM files(4)	5	5	5(9)
	Buffer prime	Wait	Wait	No wait
	Relative record file	—	1	—
	CCP/Disk Sort work file	2	5	5(9)
	Read PCT from $CCPFILE	1	1	1
	Program load	1	1	1
Program termination	Termination	5	4	2
	Terminals (each)	2 each	2 + 2 each	0(8)
	Close (one of the following):			
	Disk files(5)	5	4	6(9), (12)
	CCP/Disk Sort work file	1	2	2
	MFCU/1442	4	3	3
	Printer	4	3	3
	Disk, printer, 1442, and MFCU	7	5	7(3)
	PTAM	4	3	3

[1] Select one of these.
[2] Add this to normal disk open.
[3] If there is not a mixture of 5444 and 5445 type files, subtract 1.
[4] Only one per chain of DTFs (if two files in program, double the disk accesses, etc).
[5] For CCP/Disk Sort modules, each output and input file is opened and closed separately.
[6] None if RESREQ option selected.
[7] One if RESREQ option selected.
[8] 2 + 2 each if abnormal termination.
[9] None if resident open/close selected.
[10] Add one for multivolume direct.
[11] Add one for multivolume indexed.
[12] Add one for multivolume files.

Figure 22. Number of Disk Accesses for Program Loads and Termination

PLACEMENT OF PROGRAMS, FORMATS, AND FILES ON DISK

The *IBM System/3 Model 15 3340 Direct Access Storage Facility Reference Manual*, GC21-5111, contains a comparison of the 5444, 5445 and 3340 characteristics, such as access and rotation times, bytes of storage, track and cylinder capacities.

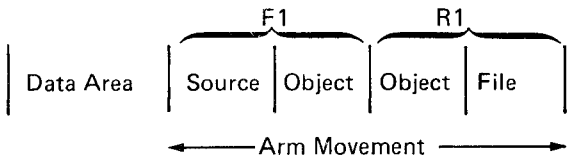
Without movement of the read head, a 5444 can access 1 cylinder (2 tracks or 48 256-byte records). A 5445 read head can also access 1 cylinder without moving (20 tracks or 400 records). A 3340 can access .6 logical cylinders (12 tracks or 576 records). A logical cylinder on a 3340 (20 tracks) thus cannot be accessed without some arm movement–five arm movements will read 3 logical cylinders. On a 3344, 1.5 logical cylinders can be read at one time, thus two arm movements will read 3 logical cylinders.

For performance reasons, all of the CCP modules except the transients are loaded onto the CCP production pack by CCP generation and then the transients are loaded. The transients require approximately 320 records or sectors. Thus, if the object library starts on a particular boundary on a 3348 data module (depending on the support you require) it is possible for all of the transients to reside in an area which would not require any disk arm movement. It is also possible on a 3348 data module for the transients to overlap a physical area and require an extra 10 ms (minimum arm movement time) to access some transients.

The most active formats and programs should be copied to the CCP production pack after a CCP generation so that minimal arm movement is required between the CCP transients and your programs and formats. Less active programs and formats should be copied to the library last.

CCP Modules ($CC. . .modules)	Most Active User Programs	Most Active Formats	Least Active Programs and Formats

If a significant amount of space is required for programs and formats, it could be more efficient in terms of minimal disk access time to place formats and $CCPFILE on the CCP program pack (for example R1), and put application programs at the end of the other simulation area on the same drive.



This would require a minimal source library size on the CCP simulation area.

The $CCPFILE should be placed (by giving the LOCATION parameter) immediately against the end of the object library. The F1 or IPL pack should only contain those modules really needed to run the system. The compiler, the sort modules and utility programs such as $COPY, $LABEL, $MAINT should be placed on another drive along with batch application programs. This effort will reduce arm contention and allow more space for CCP programs and formats.

DISK UTILIZATION

CCP performance is determined to a great degree by the demand upon the disk facility as well as the speed of the disk facility. Following are some considerations affecting the demand on the disk facility.

File and Library Placement: In order to avoid disk arm contention, CCP, DSM, and data files should be on separate disk drives if possible. Disk access times increase if CCP and the data files are on the same drive, because there can be no disk I/O overlap.

If multiple files are located on one data area, place the most active files in the center of the area so that accesses to the infrequently used files will average out. The more evenly the data files are distributed, the less waiting there will be and the better the response time. This applies to all System/3 models, especially those that have 5444 and 5445 disk drives.

For systems that use spooling, the spool file should be on a separate drive because of high disk activity when spooling. Ideally, the record length of any file should be 256 characters in length, a size evenly divisible into 256, or a multiple of 256. This is because all data managements read from a disk into the program area in even multiples of 256 (256 characters equal one sector). If record length is 256 characters, a read of only 256 is necessary into the user program. If record length is 300, it is possible that the record could spread over 3 sectors. Thus 3 sectors would have to be read in. In this case, if program A read in a record to be updated, and program B is updating the file also and wanted the record immediately preceding the record read by program A, then B would have to wait until program A updated its record before B could read the record it wanted.

GENERATION/ASSIGNMENT CONSIDERATIONS

Generation and assignment statements can be used to balance the desired speed of CCP against the resulting size of CCP. The optimum size of the CCP code is achieved when more storage results in no appreciable decrease in response time and using less storage increases response time. It is usually best to set up for optimum CCP performance and then decrease the size of CCP if storage is exhausted.

Following are some tips that affect the speed/size relationship:

- Memory-resident polling (RESPOL-YES) improves response time but requires more storage. If RESPOL-NO is specified, the batch partition may be degraded due to transient finds for polling.

- Generating CCP with minimum resident code (MINRES-YES) decreases storage requirements but degrades response time. Specifying MINRES-YES can also degrade the performance of a batch partition, since the CCP partition and batch partition will compete for use of the system transient area.

- A lockout could result if BSCA, BSCC, and MINRES are specified simultaneously. Should this occur, regenerate CCP with MINRES-NO.

- Memory-resident accept input (ACCEPT-YES, Model 15 only) improves response time but increases main storage requirements; the code necessary for this often-used operation is memory resident rather than loaded from disk as a transient when needed.

- Do not include unused terminals in the assignment set. The more terminals in your assignment set, the more storage is required (approximately 90 bytes for each terminal).

- (5704-SC2 only) If TP buffer utilization is at a maximum, the memory resident DFF buffer area (BSCALINE DFFBUF-YES) may improve data transmission and terminal response time. It increases main storage requirements, however, since it exists in addition to the put area normally built into the TP buffer.

- Using the interval polling feature (INTPOL parameter at CCP generation and POLTIME parameter at assignment) may cause less registered processing unit meter time in a relatively inactive CCP environment. However, response time may be degraded, because CCP will be polling less frequently.

- Generating the system with busy printer support (BSYPRT-YES) simplifies the coding of terminal printer programs by eliminating the need to test for the busy condition (-14 return code) in the program. This support also allows CCP and other tasks to execute while the printer is busy since the printing task will not regain control until the print operation is complete.

CONSIDERATIONS USING PRUF

Some considerations for using PRUF programs follow (see *Program Request Under Format* in Chapter 5, *3270 Screen Design* for additional considerations):

- A full buffer of data can be passed from one program to another. In effect, the 3270 is used for storing intermediate data. This can increase line activity, but it decreases disk accesses by eliminating the need to pass data through disk storage.

- Main storage is more efficiently used; for example, instead of a single program to handle multiple terminals and multiple transactions, a series of 8K or 10K programs are loaded as if they are overlays. However, the time required for additional program loads may cause an increase in terminal response times.

- Sector protection is not in effect when one program in a PRUF string reads a disk sector and goes to end of job and another program updates the file. If a record is to be updated in a later program, the record may need to be temporarily flagged to reserve inventory quantities, or other data, to prevent another terminal from updating the same record.

- With PRUF, operators are entering data to a screen format put to a terminal by a program that has since gone to end of job. If the system operator requests shutdown at this time, there is no way to notify the system operator that the terminal operator still has work to perform; consequently, CCP proceeds with shutdown. The system operators in a PRUF environment must be aware of this possibility. Operating procedures should be established to ensure that the terminal operators are informed of a pending shutdown so they can complete their work as soon as logically possible. The system operator should issue a delay shutdown command that does the following:
  1. Sends a warning message to all the terminal operators informing them of the pending shutdown.
  2. Allows sufficient time for the terminal operators to complete their work before shutdown proceeds.

- Pressing the CLEAR key on a terminal clears the last format written to the screen. If the format on the screen was a PRUF format, it cannot be rewritten to the screen because the program that initially wrote the format to the terminal has gone to end of job. Logic, such as a *help* format, or *RECOVERY* program should be included in systems using PRUF so that the PRUF screen can be rewritten to the terminal.

- Pressing a PA key causes only the AID byte to be returned to CCP. No program request data is returned to CCP when a PA key is pressed. When requesting a PRUF program, the ENTER or PF keys must be used.

- Disk utilization is more efficient if a terminal operator using a PRUF format can enter the maximum amount of data before requiring program intervention. That is, as the time between transactions increases, disk utilization decreases. However, response time may increase if many separate transactions are entered on one screen, because editing them takes longer.

MISCELLANEOUS CCP TIPS

Terminals in ERP (Model 15 Only): Terminals left in ERP (error recovery procedure) may cause fragmentation of the TP buffer. Terminals should be either removed from ERP or varied offline.

Loops in a CCP Task: Beware of loops in CCP user tasks. An endless loop in a CCP user task stops the other level on Models 8, 10, and 12, and, if CCP is the highest priority partition, the other partition(s) on a Model 15. Within the CCP area on Models 4, 8, 10, and 12, an endless loop stops all other user tasks; on Model 15, an endless loop stops all lower priority tasks.

Attention Identification Keys: Be aware that attention identification keys are not available on all 3270 keyboards.

PA Keys: When the operator of a terminal presses a PA key to cause attention, only the AID character is transmitted from the 3270 to the processing unit; no data is transmitted. Only the PF and ENTER keys cause data to be transmitted. The ENTER key should be used as much as possible in the normal application flow–in fact, exclusively if possible. An operator is slowed and more prone to errors if other AID keys are required on a regular basis.

Right Adjustment and Negative Input: Right adjustment and negative input are handled automatically for numeric input fields as follows:

- A numeric field is written to the screen as null characters (hexadecimal 00) when the original format write is performed.
- On subsequent input, trailing nulls are removed, shifting the remaining field contents to the right.
- If the field contains a - (minus sign) as the right character, the - is removed and the number is treated as a negative number.
- Auto skip only works when the rightmost field position is entered; the right tab key must be used to get to additional input fields if a field is only partially filled.

Some examples of a 5-position numeric input field follow:

Keying Position					
1	**2**	**3**	**4**	**5**	**Read As**
7	5	(right tab)			00075
7	5	-	(right tab)		0007N (-75)
7	5	b	(right tab)		00750 (blank = zero)

Output/Input Fields Carrying Negative Values: Special processing is required to use negative values with output/input fields, because the 3270 hardware cannot represent a negative zero. For example, a 3-byte output/input field containing minus ten cents (unedited) is transmitted to the 3270 as X'F0F1D0', but is loaded to the 3270 buffer as X'F0F150', which displays as 01&. When this hexadecimal character string is received by DFF (to which an output/input numeric modified field cannot be defined), DFF passes it to a user program without signing or justifying it. Therefore, the field appears to be plus ten cents when the field is processed numerically by the user program (the low order X'50' is equivalent to X'F0' when used arithmetically in the System/3). To pass a negative value via a 3270 buffer, the user should edit the value with trailing minus sign to a DFF output/input field. To receive the field, the user should redefine the last character as an alphameric character and test it for an &. If the character is an &, then the whole field should be forced to a negative field. When using PRUF, the user has another option: separate display formats can be used to write the data and to receive the data. The formats should differ in that the field is defined as output/input for writing and as input for reading.

Clear Key and Data Mode Escape: Pressing the CLEAR key on a 3270 terminal sets the entire buffer to nulls (hexadecimal 00). Programs should be written to check for this return code and, upon receiving it, rewrite the previous screen format.

On systems without data mode escape, the CLEAR return code is immediately given to the program. On systems with data mode escape, the CLEAR return code is not indicated to the application program until the next time the operator presses an attention-causing key (ENTER, PF key, PA key). It is, therefore, recommended that data mode escape not be generated into CCP unless it is to be used.

Messages to the Terminal Operator: Messages (what to do next, what was just done, what error just occurred) to the terminal operator are easier to find if they are set up on a message line that is in a constant location on all display formats.

Link Editing CCP Programs: CCP programs must be link edited on a system that has been generated to support CCP because of the CCP subroutines (such as SUBR92 and CCPCIO) and, on Models 4, 8, 10, and 12, because of the special unit record data management.

DFF Formats: The time required to load additional display formats into program-appended storage can be minimized by having only one regularly used format.

Record Identifying Indicators: When using READ/EXCPT logic, remember that the record identifying indicators are turned off by RPG II when the program goes through the normal input cycle logic (after detail and LO calculations).

Model 15 Programs: A user program cannot exceed 32K including the program appended storage (but excluding external buffer size and the MORCOR option for memory resident overlays on the Model 15D). The RPG II H-specification *Size to Execute* (columns 12 through 14) should not be used, except to force the program into overlays. Any value specified in these columns that is less than the value needed by the program to execute forces overlays. On the Model 15, if a value is specified that is more than that required for the program to execute, the excess storage is wasted. If the program uses DFF, the program appended storage is added to the program when it is loaded.

Compiling Model 15 Programs: A COMPILE LINKADD-8000 must be provided when compiling a Model 15 CCP program. Batch programs are link edited at 4000. If an RPG II program is to execute in both a batch and CCP environment, the overlay linkage editor must be a separate step, with COMPILE LINKADD-8000 and RLD-YES specified. This step is automatic for FORTRAN and COBOL compilers and the CCP/Disk Sort program.

IDELETE Mode on the Model 15: If IDELETE mode is not specified before CCP start-up, the system will appear to be in a wait state, waiting for a response to the tenth message before another could appear on the system console.

Memory Resident Overlays: In some cases, using memory resident overlays is not efficient use of user program areas. The *IBM System/3 Model 15 Overlay Linkage Editor Reference Manual* contains additional information concerning the two memory resident overlay techniques, REMAP and MOVE.

This page is intentionally left blank.

Appendix A. Glossary

For definitions of communications and data processing terms that are not included in this glossary, see *IBM Data Processing Glossary*, GC20-1699, or publications listed in Appendix B, *Bibliography*.

$CCPFILE: A CCP control file on a disk in which, during CCP assignment stage, the user defines one or more specific operating environments for CCP. Each operating environment consists of a set of terminals, files, and programs that can be used during a particular run of the CCP.

addressing: In communications, the means whereby the originator or control station selects the unit to which it is going to send a message.

AID character: Attention identification character.

algorithm: A prescribed set of well-defined rules or processes for the solution of a problem in a finite number of steps. In using direct files, an access algorithm describes how the contents of a key field are used to determine a relative record location.

application: Data processing work that is accomplished with the assistance of a computer.

application program: A program written for or by a computer user that applies to his own work.

assignment stage: The special preparatory CCP run during which the user defines one or more sets of specific operating environments in which CCP can run.

attention identification (AID) character: A code that is set in a 3270 display station when the operator takes an action that produces an I/O pending condition. The character identifies the action or key that caused the condition to be generated. The AID is set when the display station operator presses a program access key, ENTER key, TEST REQ key, or program function key; when a selector pen attention occurs; or when a successful operator identification card read-in occurs. It also identifies device addresses assigned to printers.

attribute (3270): A characteristic of a display field. The attributes of a display field include: protected or unprotected; numeric-only or alphameric input control; displayed, nondisplayed, display intensified; selector-pen-detectable or nondetectable; and modified or not modified.

attribute character (3270): A code that defines the attributes of the display field that follows. An attribute character is the first character in a display field, but it is not a displayable character.

batch mode: The operating method in which programs are being executed such that each is completed before the next is started.

batch program: An application program that processes a series of related transactions that have been grouped together. Batch programs can run under disk system management control and under CCP control.

batch data: Data, such as transactions, that is grouped to be transmitted or processed in a continuous series.

binary synchronous transmission: Data transmission in which synchronization of characters is controlled by timing signals generated at the sending and receiving stations.

block mode operations: Operations that result in all data from an operation in the program up to ETB (end text block) being moved into or from the user programs's record area.

BSCA: Binary synchronous communications adapter.

CCC: Copy control character.

command interrupt mode: The operating mode of a terminal following data mode escape until the program execution is resumed by a RUN command (the terminal reenters data mode) or until the terminal is released by a RELEASE command (terminal enters command mode).

command mode: The operating mode of a command terminal following a successful sign-on, up to and including the program request. Following program termination, a terminal returns to command mode until another program request is made or until sign-off.

command terminal: A terminal that is capable of commanding CCP services related to requesting a program. Terminals are designated as either command terminals or data terminals at assignment time.

common carrier: Any government-regulated company that furnishes communication services and facilities to the general public; for example, a telephone or telegraph company.

communication management: A major function of CCP that controls terminal input/output.

communications service subroutine: A relocatable subroutine provided by CCP that is link-edited to user programs. The subroutine is called by the user program whenever the program requires a communications service, such as sending and receiving messages.

consecutive processing: A mode of file processing in which records are processed in the order they appear in the file. Contrast with *random processing*.

control station: The primary or controlling computer in a multipoint telecommunications configuration.

copy control character (CCC): A character used in conjunction with the 3270 copy command to specify that a particular operation, or combination of operations, is to be performed at a display station or printer in the data that is to be copied.

copy operation: A 3270 DFF operation that copies the contents of the buffer from one display station or printer to another display station or printer attached to the same control unit.

cursor: A unique symbol (an underscore) that identifies a character position in a 3270 screen display, usually the character position at which the next character to be entered from the keyboard will be displayed.

data entry application: A communications-based system application in which terminals are in relatively prolonged communication with an application program (as opposed to the typical inquiry application), for example, entering data for document preparation (such as invoice preparation), or entering data directly into data files from a terminal.

data mode: The operating mode of a terminal when it is under control of a user program, until the program terminates, the terminal is released by the program, or the data mode escape characters are entered. While in data mode, a terminal is not in direct communication with CCP.

data mode escape: A special CCP command, consisting of a unique string of six characters entered at a requesting terminal while the terminal is in data mode. The data mode escape command temporarily suspends a terminal's communication with a program and places the terminal in command interrupt mode.

data security: The protection of data against damage, loss, unauthorized access, or unauthorized use.

data stream: All data transmitted through a communication channel on a single operation, including data link control, device control, and data characters.

data terminal: A terminal that is not capable of commanding CCP services. A data terminal is always either in standby mode (not polled for input by CCP) or in data mode (under control of an application program).

dedicated program: A program running under CCP that requires sole use of the CCP user program area.

designator character: A character that immediately follows the attribute character in a 3270 selector-pen-detectable field. The designator character controls whether a detect on the field will or will not cause an attention. For a nonattention-producing field, the designator character also determines whether the modified data tag for the field is to be set or reset as the result of a selector-pen detect.

DFF: 3270 Display Format Facility of CCP.

direct file: A disk file organization in which, for purposes of storage and retrieval, there is a relationship between the contents of the records and their locations in the file. Contrast with *sequential file* and *indexed file*. See also *algorithm*.

display adapter: An IBM device that converts the binary data stream from the device buffer into signals on the communication line, and vice versa. The display adapter provides for the local attachment of 3277 displays and of 3284, 3286, and 3288 printers to System/3.

file management: A major function of CCP that controls the use of data files by programs running under CCP.

home record: A record in a direct file that is stored in the location indicated by its relative record number (home location).

implied invite input: An invite input that is not actually issued by the user program, but exists because data is allowed with the program request. Implied invite inputs are included in the count of outstanding invite inputs in the communications parameter list for certain operations.

indexed file: A disk file organization in which records are arranged in logical sequence by key. Indexes to these keys permit records to be processed either randomly or sequentially. Contrast with *direct file* and *sequential file.*

initial mode: The operating mode of a command terminal before a sign-on at the terminal has been accepted by CCP.

inquiry: A communications-based system application in which, typically, a single transaction or request for information is entered from a terminal and a response is returned to the terminal.

inquiry with update: A communications-based system application in which records of transactions entered from terminals are used to interrogate and update one or more master files maintained by the system (synonymous with inquiry and transaction processing).

integrated display adapter: See *display adapter.*

integrity: See *system integrity.*

interface: In application programming under CCP, the data areas (parameter list and record area), communications service subroutines, and defined operations by which user programs and CCP communicate with each other.

line buffer: The internal main storage area associated with a communication line from which data is transmitted to a terminal or into which data is received from a terminal. Data in this area includes device and line control characters inserted or removed by CCP.

local display adapter: See *display adapter.*

local terminal: In the CCP environment, a 3270 Information Display System device attached to the system via the display adapter. Contrast with *remote terminal.*

master file: A file that is either relatively permanent or that is treated as an authority in a particular job.

MDT: Modified data tag.

message mode operations: Operations that result in all blocks of data including the EOT signal being sent or received in a single operation.

MLMP: Multiline/multipoint BSCA input/output control system (IOCS), the base data management and IOCS included in the CCP for binary synchronous communications.

modified data tag (MDT): A bit in the attribute character of a 3270 display field which, when set on, causes the field to be read on an input operation. The modified data tag may be set by (1) a keyboard input to the field, (2) a selector-pen detection in the field, (3) a card read-in operation, or (4) program control. The modified data tag may be reset by (1) a selector-pen detection in the field, (2) program control, or (3) ERASE INPUT key.

MRT program: Multiple requesting terminal program.

multipoint line: A line interconnecting several stations. Synonymous with multidrop line.

multiple requesting terminal (MRT) program: A type of application program under CCP that can process additional requests for it even though it is still processing an earlier request.

multitasking: (CCP) The concurrent execution of one or more user tasks under control of CCP.

NEP: Never-ending program.

never-ending program: A user application program, which, after it has been initiated, normally remains in main storage and does not go to end of job until CCP is shut down.

nonswitched line: A connection between a remote terminal and a computer that does not have to be established by dialing.

null character: A hex 00 character on a 3270 that occupies a position in the storage buffer and is displayed as a blank.

object library: An area on disk storage used to store object programs, routines, and, if DFF is used, display formats.

object program: A fully compiled program that is ready to be loaded into the computer system.

OCL: Operation control language.

offline: Pertaining to equipment, devices, or processes not under active control of the processing unit.

online: Pertaining to equipment, devices, and processes that are under active control of the processing unit.

online system: A system in which the input data enters the computer directly from the point of origin or in which output data is transmitted directly to where it is used.

order entry application: A form of data entry application in which transactions (such as sales orders) are entered into the system from terminals.

output/input field: One of four classes of fields defined under the 3270 Display Format Facility. Output/input fields contain data that has been supplied either during format generation or during execution of the application program; this data can be changed by the terminal operator using the keyboard.

password security option: An optional CCP feature, selected during assignment, which requires a terminal operator to enter a predetermined password before CCP will allow the terminal to enter commands.

physical file: See *symbolic file.*

point-to-point line: A line that connects a single remote station to the computer; it may be either switched or nonswitched.

polling: A technique by which each of the terminals on a multipoint line is periodically interrogated to determined whether it requires servicing.

procedure: A named collection of related OCL statements, and possibly utility control statements, that perform a particular task.

program management: The major function of CCP that fetches programs, allocates system resources to programs, manages the concurrent execution of two or more programs, purges programs from main storage, and optionally maintains a count of the number of times each application program is requested.

program request: A command, consisting of a program name entered at a terminal or the system operator's console, that causes CCP to initiate execution of an application program.

program request count: The optional CCP program management function of maintaining a count of the number of times each application program is requested.

program request under format (PRUF): A method of requesting a program from a display format on a 3277 or 3275. The entire screen can be used to pass data with the program request. The name of the program to be requested appears as the first input field from the 3270 terminal.

program-selected terminal: From the point of view of the application program, a terminal that is selected by an application program for input/output, as opposed to a terminal that requested the program (see *requesting terminal*). Program-selected terminals can be either required (must be allocated to the program before the program can run) or acquired (allocated dynamically to the program as it is running).

program termination code: A two character code provided by CCP when an application program has been cancelled by CCP because of certain coding errors or program logic errors, or because the system operator requested cancellation of the program. This code identifies the reason for the cancellation.

protected field: A 3270 display field for which the display operator cannot use the keyboard or operator identification card reader to enter, modify, or erase data.

PRUF: Program request under format.

queue: A waiting line or list formed in a system by items that are waiting for service.

random processing: The treatment of data without respect to its location in external storage and in an arbitrary sequence, governed by the input against which it is to be processed. Contrast with *consecutive processing* and *sequential processing.*

record mode operations: Application program input and output operations that result in a single record of a data block being moved into or out of the program's record area.

relative record number: In a direct file, the location of a record in relation to the beginning of the file.

remote job entry: Submission of job control statements and data from a remote terminal to a central system, causing the jobs to be scheduled and executed by the central system.

remote terminal: In the CCP environment, a device attached to the system via the BSCA or MLTA. Contrast with *local terminal.*

requesting terminal: From the point of view of the application program, a terminal that requested the program, as opposed to a terminal that is selected by the program (see *program-selected terminal*). Requesting terminals are always command terminals.

response time: See *terminal response time.*

RVI: A signal from a receiving device to a device that is transmitting to interrupt its transmission as soon as possible.

SCP: System control program.

security: See *data security* and *system security.*

seek: To position the access mechanism of a direct access device at a specified position.

selector-pen-detectable (SPD) field: One of four classes of fields defined under the 3270 Display Format Facility. SPD fields allow the terminal operator to enter data by using the selector pen.

sequential file: A file organization in which records are arranged in a physical sequence. The records are not necessarily in logical sequence. Contrast with *direct file* and *indexed file.*

sequential processing: A treatment of data with respect to its location in external storage, and in a sequence governed by the logical order of the data in the file. Contrast with *consecutive processing* and *random processing.*

serial printer: A printer that prints characters one at a time. Contrast with a line printer, which prints a line at a time. Synonymous with character printer.

shutdown: The final stage of CCP operation, during which CCP allows programs currently executing or scheduled to finish processing, then closes files, adapters, and communication lines.

sign-on: The procedure performed at a terminal while it is in initial mode. This procedure may include entering only the /ON command, or entering the /ON command with a password or other user-specified security data.

single requesting terminal (SRT) program: A type of application program under CCP that can process a request from only one requesting terminal during its execution.

source library: An area on disk used to store source programs, OCL procedures, and control statements. Contrast with *object library.*

source program: A computer program written in a source language, such as RPG II, before the program is compiled.

SPD field: Selector-pen-detectable field.

SRT program: Single requesting terminal program.

standby mode: The mode of a data (noncommand) terminal when it is not under control of a user program.

startup: The initial phase of CCP operational stage, during which all necessary initialization occurs, including opening of disk files, adapters, and communication lines, and the completion of various tables and control blocks.

subhost: A telecommunications system which, while directly controlling a group of terminals, is itself a tributary station to another central processor.

switched line: A communication line in which the connection between the computer and remote station is established by dialing.

symbolic file: A file reference (symbolic name) which allows, on separate executions of a program, reference to different files, known as *physical files.* A symbolic file is related by the terminal operator to a specific physical file by means of a /FILE command.

synonym record: A record in a direct file whose control field yields the same relative record number as another record.

system control programming: IBM-supplied programming that is fundamental to the operation and maintenance of the system. It serves as an interface with program products and user programs.

system integrity: Preservation of the accuracy and completeness of data and programs.

system security: Protection of computer data, programs, and devices against damage, loss, unauthorized access, or unauthorized use.

system task: A unit of work for the processing unit from the standpoint of CCP, consisting of a CCP function (as opposed to a user application, or *user task*) that must be performed by CCP, such as communications management.

system throughput: The total volume of work performed by a computing system over a period of time.

task: See *system task* and *user task*.

task identification: An identifying character associated with a task which differentiates between that task and other tasks running concurrently under CCP.

terminal: A device capable of sending and/or receiving information over a communication channel.

terminal attributes: Characteristics of a terminal from the point of view of CCP and CCP application programs, including block length, record length, data format, and other information.

terminal reference identifier: A unique two-character identifier, assigned to each terminal during CCP assignment stage, that is used by CCP and the system operator to refer to a specific terminal. Any of the 64 graphic EBCDIC characters may be used.

terminal response time: The time interval from when the terminal operator enters data to the system until the keyboard is opened to permit more data to be entered.

throughput: See *system throughput*.

transaction: The entry of some request or unit of data, the processing of the request or unit, and the return of some response or acknowledgment.

transaction file: A file containing relatively transient data to be processed in combination with a master file. For example, in a payroll application, a transaction file indicating hours worked might be processed with a master file containing employee name and rate of pay.

transaction-oriented processing: A method of processing data in which each different type of application transaction is processed by a separate program, as opposed to processing multiple transaction types in a single program.

translation: Under CCP, conversion of the transmission line data code (if not EBCDIC) into EBCDIC or conversion from EBCDIC into transmission line data code.

tributary station: A secondary or noncontrolling device in a multipoint telecommunications configuration.

truncation: Loss of excess data when the length of data received from a terminal is greater than the maximum input length specified in the parameter list or when more data is provided in an output operation than the line buffer for the terminal can hold (in record mode output operations, if the output length exceeds the record length specified in the terminal attributes set).

unprotected field: A 3270 display field for which the terminal operator can manually enter, modify, or erase data.

user task: A unit of work for the processing unit from the standpoint of CCP, consisting of a user program (as opposed to a system function, or *system task*) that must be executed by CCP.

WCC: Write control character.

work station: Elements of data processing equipment through which a system's end user has access to a computer as required for the performance of his job (work) at the physical location (station) where he performs his job tasks.

write control character (WCC): A character used in conjunction with 3270 write operations to specify that a particular operation, or combination of operations, is to be performed at a display station or printer.

Appendix B. Bibliography

This appendix contains a list of CCP publications and related programming and data communications publications. For a complete list of System/3 publications, see *IBM System/3 Bibliography*, GC20-8080.

CCP

- *IBM System/3 Communications Control Program General Information Manual*, GC21-7578
- *IBM System/3 Communications Control Program Terminal Operator's Guide*, GC21-7580
- *IBM System/3 Communications Control Program Messages Manual*, GC21-5170
- *IBM System/3 Communications Control Program Programmer's Reference Manual*, GC21-7579
- *IBM System/3 Models 8, 10, and 12 Communications Control Program System Reference Manual*, GC21-7588
- *IBM System/3 Models 8, 10, and 12 Communications Control Program System Operator's Guide*, GC21-7581
- *IBM System/3 Model 15 Communications Control Program System Reference Manual*, GC21-7620
- *IBM System/3 Model 15 Communications Control Program System Operator's Guide*, GC21-7619
- *IBM System/3 Model 4 Introduction*, GC21-5146
- *IBM System/3 Model 4 Communications Control Program Concepts and System Design Guide*, GC21-5148
- *IBM System/3 Model 4 CCP Programmer's Reference Manual*, GC21-5150
- *IBM System/3 Model 4 Operator's Guide*, GC21-5149
- ***IBM System/3 Model 15D System Measurement Facility Reference and Logic Manual***, GC21-5207
- ***IBM System/34 and System/3 Model 15D Distributed Disk File Facility Reference Manual***, SC21-7869

Programming

- *IBM System/3 RPG II Reference Manual*, SC21-7504
- *IBM System/3 Models 4 and 6 RPG II Reference Manual*, SC21-7517
- *IBM System/3 RPG II Telecommunications Programming Reference Manual*, SC21-7507
- *IBM System/3 RPG II Auto Report Feature Reference Manual*, SC21-5057
- *IBM System/3 RPG II 3270 Display Control Feature Reference and Logic Manual*, SC21-5161
- *Introduction to RPG II*, GC21-7514
- *IBM System/3 RPG II Disk File Processing Programmer's Guide*, GC21-7566
- *IBM System/3 Disk Concepts and Planning Guide*, GC21-7571
- *IBM System/3 RPG II Additional Topics Programmer's Guide*, GC21-7567
- *IBM System/3 Subset American National Standard COBOL Reference Manual*, GC28-6452
- *IBM System/3 FORTRAN IV Reference Manual*, SC28-6874
- *IBM System/3 Basic Assembler Reference Manual*, SC21-7509
- *IBM System/3 Disk Sort Reference Manual*, SC21-7522

MLTA and Supported Terminals

- *IBM System/3 Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual*, GC21-7560
- *IBM 2740 Communications Terminal Models 1 and 2 Component Description*, GA24-3403
- *IBM 2741 Communication Terminal*, GA24-3415
- *IBM 1050 Data Communication System Principles of Operation*, GA24-3474
- *IBM 3767 Models 1 and 2 Communications Terminal Component Description Manual*, GA27-3096

BSCA and Supported Terminals/Systems

- *General Information: Binary Synchronous Communications*, GA27-3004
- *IBM System/3 Models 4 and 6 Components Reference Manual*, GA34-0001
- *IBM System/3 Models 8, 10, 12, and 15 Components Reference Manual*, GA21-9236
- *IBM 3270 Information Display System Component Description*, GA27-2749
- *An Introduction to the IBM 3270 Information Display System*, GA27-2739
- *Operator's Guide for IBM 3270 Information Display Systems*, GA27-2742
- *IBM System/3 3735 Support Program Coding Manual*, GC21-5096
- *IBM 3735 Programmer's Guide*, GC30-3001
- *IBM 3740 Data Entry Systems Programmers Guide*, GC21-5071
- *IBM 3741 Data Station Reference Manual*, GA21-9183
- *IBM System/7 Binary Synchronous Communications Module (RPQ) Programming Guide and Reference Manual*, SC34-1510
- *IBM System/7 System Summary*, GA34-0002
- *IBM System/3 Multiline/Multipoint Binary Synchronous Communications Reference Manual*, GC21-7573
- *IBM System/3 MULTI-LEAVING Remote Job Entry Work Station Support Reference Manual*, GC21-7621
- *IBM System/3 Model 15 MULTI-LEAVING Remote Job Entry Work Station Support Reference Manual*, GC21-5115
- *IBM System/3 DATA/3 Reference Manual*, SC21-5102

General Data Communications

- *Data Communication Concepts*, GC21-5169
- *Introduction to Data Communications Systems*, ZR20-4542
- *IBM Terminals —Student Text*, SR20-4452
- *Introduction to Data Communications Network Design—Student Text*, SR20-4482
- *IBM 3270 Screen Design—Student Text*, SR20-4441

READER'S [illegible]VMENT FORM

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Name _____

Company or Organization _____

Address _____

_____

● No postage necessary if mailed in the U.S.A.

GC21-5165-1

Fold and tape

Please do not staple

Fold and tape

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N. Y.

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold and tape

Please do not staple

Fold and tape

IBM

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30055
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30055
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

IBM S/3 Communications Control Program System Design Guide (File No. S3-36) Printed in USA GC21-5165-1

GC21-5165-1