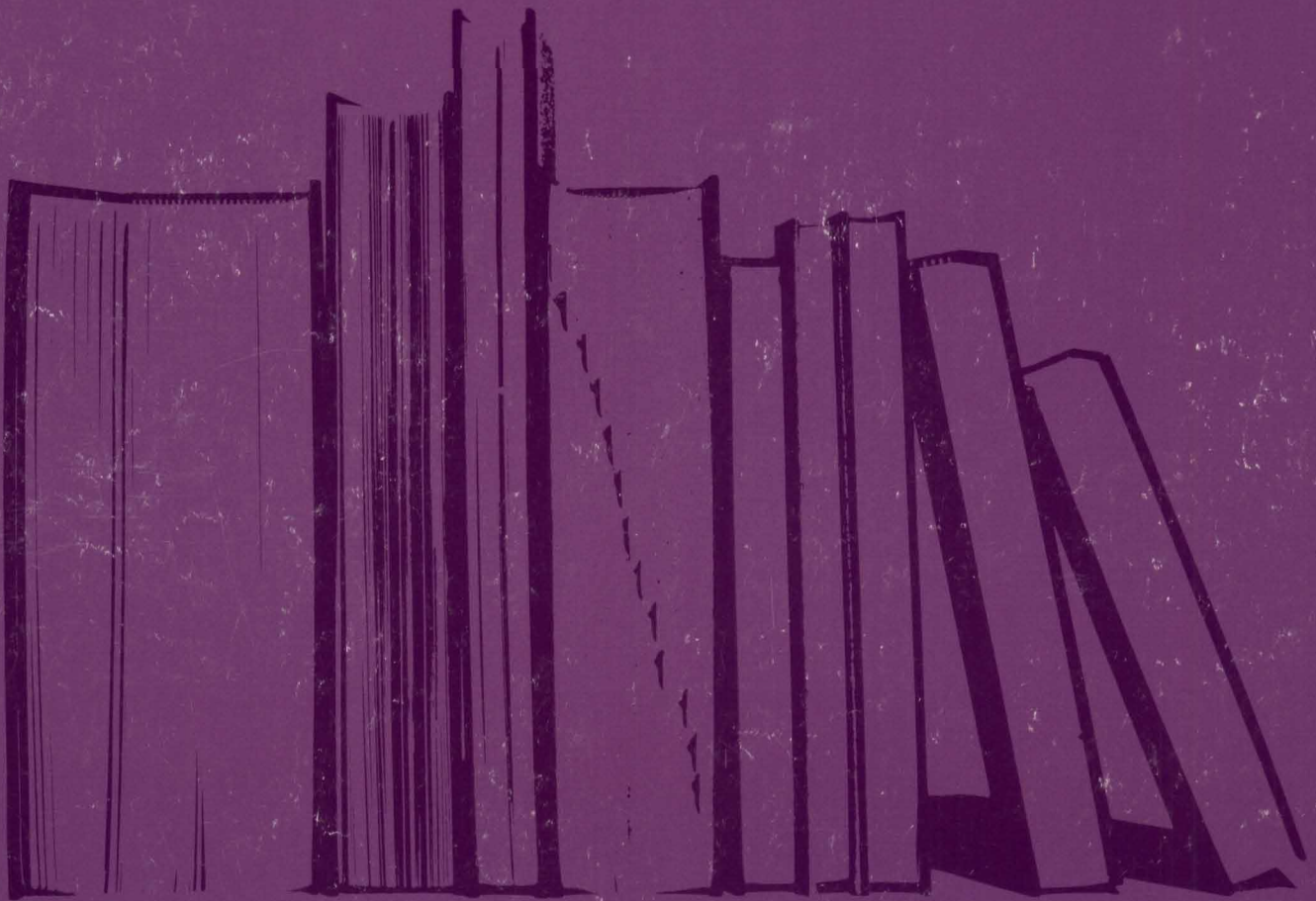


IBM

Guided Learning Center

**IBM System/38 RPG III and Structured Programming
Student Materials Book**





ZR30-0870-4

IBM System/38 RPG III and Structured Programming Student Materials Book

Preface

This Student Materials Book is designed to be used in an IBM Guided Learning Center environment. It is intended to be kept by the student on completion of the course.

This Student Materials Book is used with module texts for the IBM System/38 RPG III and Structured Programming course.

The module texts in this course are:

Module 1. Course Introduction

Module 2. Procedural File Operations

Module 3. Structured Programming Concepts

Module 4. Structured Programming Implementation

Module 5. Program Compilation and Testing

Module 6. Subprograms

Module 7. Subfile Programming

Module 8. Data Areas

Module 9. Data Structures

Module 10. Handling Exception Errors

Module 11. Additional RPG III Functions

Module 12. Program Described Work Station Files

Module 13. Printer Device Files

Fifth Edition (November 1986)

This is a major revision of, and obsoletes, ZR30-0870-3. Changes are periodically made to the information herein; any such changes will be reported in subsequent editions.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

You may communicate your comments about this publication, its organization or subject matter with the understanding the IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Address such comments to IBM Corporation, Department 7TF, P.O. Box 2150, Atlanta, Georgia 30055.

Curriculum Flowchart v

Course Overview 1
 Introduction 1
 Course Objectives 1
 Course Topics 2
 Course Structure 3
 Video Presentations 3
 Module Texts 3
 Student Materials Book 4
 Exercises 5
 Reference Material 5
 Progress Checks 5

Chapter 1. Module 1-Course

Introduction 1-1
 Module Purpose 1-1
 Time Estimate 1-1
 Module Objectives 1-1
 Progress Check: Unit 1 1-2

Chapter 2. Module 2-Procedural File

Operations 2-1
 Module Purpose 2-1
 Time Estimate 2-1
 Module Objectives 2-1
 Terms 2-2
 Full Procedural Processing 2-3
 Unit 2 Summary 2-4
 Progress Check: Unit 2 2-5
 Work Station File Processing 2-7
 Progress Check: Unit 3 2-8
 Module Summary 2-10

Chapter 3. Module 3-Structured

Programming Concepts 3-1
 Module Purpose 3-1
 Time Estimate 3-1
 Module Objectives 3-1
 Terms 3-2
 Video Introduction 3-3
 Video Summary 3-3
 Flowchart 3-4
 Progress Check 3-5
 Desk Exercise 3-8
 Module Summary 3-14

Chapter 4. Module-4 Structured Programming Implementation 4-1

Module Purpose 4-1
 Time Estimate 4-1
 Module Objectives 4-1
 Terms 4-2
 Video Introduction 4-3
 Video Summary 4-3
 Unit 2 Summary 4-4
 Progress Check: Unit 2 4-5
 Desk Exercise—Open Purchase Order
 File Update 4-8
 Module Summary 4-13

Chapter 5. Module 5-Program Compilation and Testing 5-1

Module Purpose 5-1
 Time Estimate 5-1
 Module Objectives 5-1
 Terms 5-2
 Machine Exercises Introduction 5-3
 Machine Exercise - Open Purchase
 Order File Update 5-7
 Progress Check: Unit 2 5-12
 Coding Exercise - Open Purchase Order
 File Maintenance 5-14
 Machine Exercise - Open Purchase Order
 File Maintenance 5-20
 Module Summary 5-23

Chapter 6. Module 6-Subprograms 6-1

Module Purpose 6-1
 Time Estimate 6-1
 Module Objectives 6-1
 Terms 6-2
 Coding Exercise - Subprograms 6-3
 Machine Exercise - Subprograms 6-12
 Module Summary 6-17

Chapter 7. Module 7-Subfile

Programming 7-1
 Module Purpose 7-1
 Time Estimate 7-1
 Module Objectives 7-1
 Terms 7-2
 Video Introduction 7-3

Video Summary	7-3
Subfile Review	7-4
Pseudo Code	7-6
Coding Exercise - Subfiles-Inquiry	7-8
Machine Exercise - Subfiles-Inquiry	7-15
Coding Exercise - Subfiles-Update	7-18
Machine Exercise - Subfiles-Update	7-26
Module Summary	7-33
Chapter 8. Module 8-Data Areas	8-1
Module Purpose	8-1
Time Estimate	8-1
Module Objectives	8-1
Terms	8-2
Video Introduction	8-3
Video Summary	8-3
Coding Exercise - Data Area with IN and OUT	8-4
Machine Exercise - Data Area with IN and OUT	8-7
Module Summary	8-9
Chapter 9. Module 9-Data Structures	9-1
Module Purpose	9-1
Time Estimate	9-1
Module Objectives	9-1
Terms	9-2
Video Introduction	9-3
Video Summary	9-3
Types of Data Structures	9-4
Coding Exercise - Simple Data Structure	9-5
Machine Exercise - Simple Data Structure	9-8
Progress Check: Unit 2	9-13
Module Summary	9-17
Chapter 10. Module 10-Handling Exception	
Errors	10-1
Module Purpose	10-1
Time Estimate	10-1
Module Objectives	10-1
Terms	10-2
Exception Errors	10-3
Managing Exception Errors	10-4
Coding Exercise - Exception Error	10-5
Machine Exercise - Exception Error	10-10
Module Summary	10-13

Chapter 11. Module 11-Additional RPG III	
Functions	11-1
Module Purpose	11-1
Time Estimate	11-1
Module Objectives	11-1
Terms	11-2
Input/Output Operation Codes	11-3
Unit 2 Summary	11-4
Progress Check: Unit 2	11-5
Applications For Multiple Device Files	11-7
Multiple Device Files	11-8
Using Commitment Control	11-9
Overview	11-9
Module Summary	11-10

Chapter 12. Program Described Work	
Station Files	12-1
Module Purpose	12-1
Time Estimate	12-1
Module Objectives	12-1
Terms	12-2
General Concepts	12-3
Progress Check: Unit 2	12-4
Module Summary	12-6

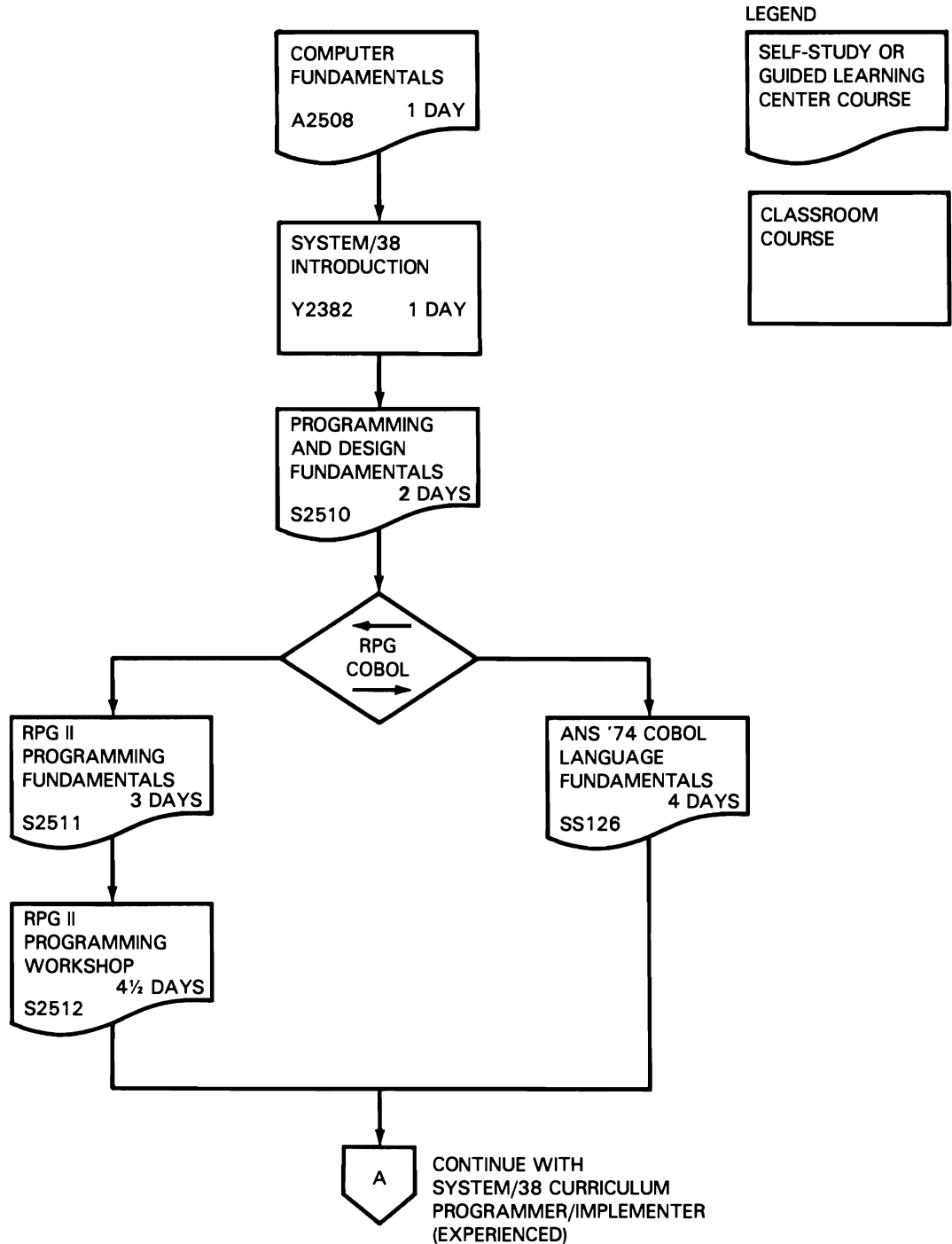
Chapter 13. Module 13-Printer Device	
Files	13-1
Module Purpose	13-1
Time Estimate	13-1
Module Objectives	13-1
Terms	13-2
Overview	13-3
Progress Check: Unit 2	13-4
Module Summary	13-6

Chapter 14. Course Summary	14-1
-----------------------------------	------

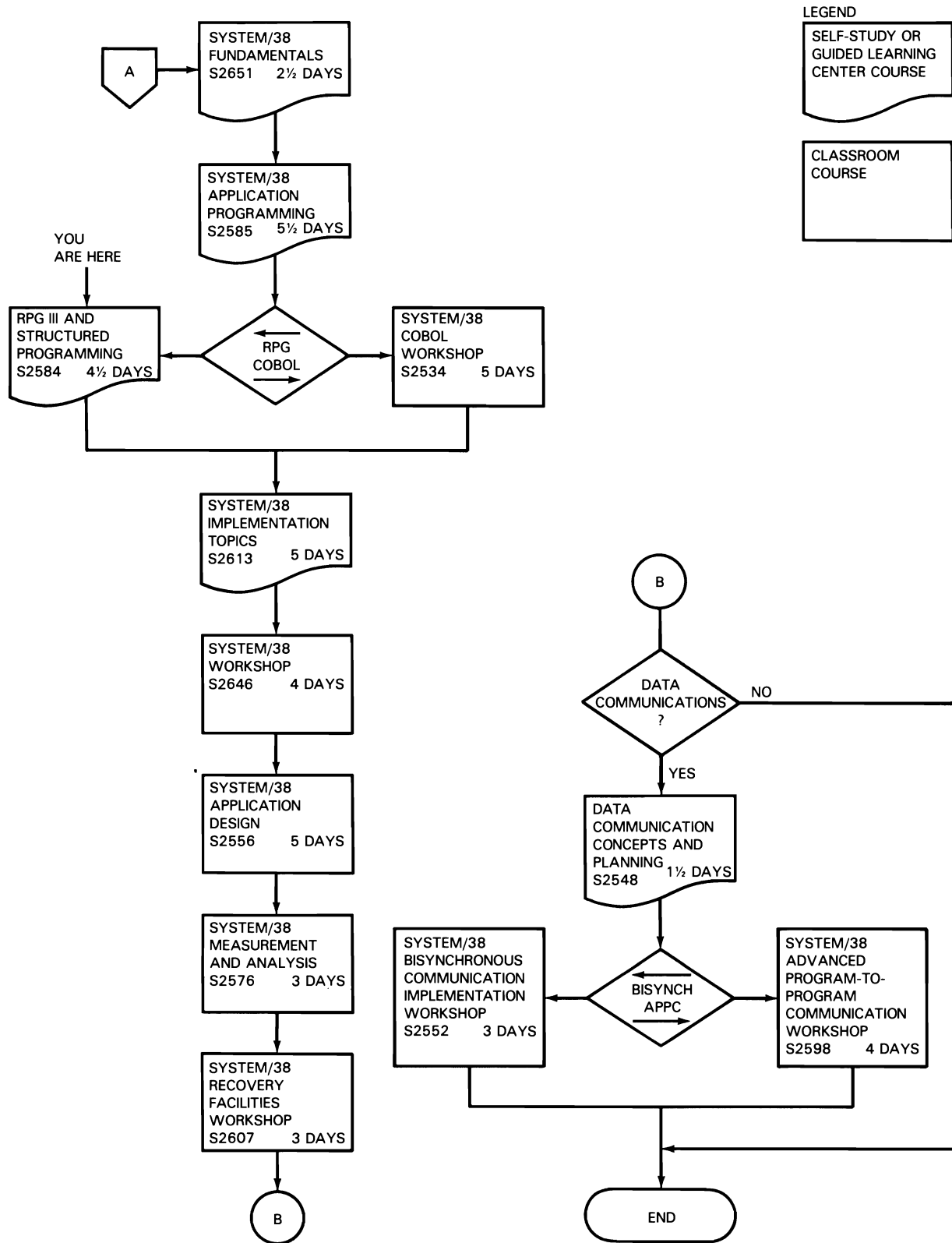
Appendix A. References	A-1
Field Reference File	A-2
Job Description	A-5
Profile	A-6

This flowchart is provided as a guide to the educational offerings for the System/38.

**SYSTEM/38 CURRICULUM
PROGRAMMER/IMPLEMENTER
(NEW TO DATA PROCESSING)**



SYSTEM/38 CURRICULUM PROGRAMMER/IMPLEMENTER (EXPERIENCED)



Course Overview

This is where you start the course.

INTRODUCTION

This course is designed to enhance the technical and professional skills of the RPG programmer by adding an understanding of RPG III. It assumes prior knowledge of the RPG II language and only addresses the enhancements found in RPG III.

The purpose of this course is to teach you to write, test, and maintain RPG III programs for the System/38. Another purpose of the course is to acquaint you with reference material applicable to writing RPG III programs. Machine exercises are provided to give you the experience of writing, entering, compiling, and testing your programs on an IBM System/38.

The RPG III language provides the programmer with many new capabilities such as full procedural file processing – the capability of controlling input through the calculation specifications. In addition, RPG III contains operation codes that support structured programming.

COURSE OBJECTIVES

After completing this course, using System/38 reference material, you should be able to:

- Code, enter, compile, and test RPG III programs for:
 - Normal inquiry
 - Inquiry with update
 - File maintenance
 - Use of data areas and data structures
- Receive and pass parameters in communicating with other programs.
- Structurally develop a flowchart for an application program.
- Use a structured flowchart as documentation to code, enter, compile and test RPG III programs.
- Process a subfile using the RPG III language.
- Write an error handling subroutine in an RPG III program.

COURSE TOPICS

The following topics are presented to help you meet the objectives of "System/38 RPG III and Structured Programming."

- Full Procedural Processing
- Work Station File Processing
- Data Base File Processing
- Structured Programming
- Structured Program Operation Codes
- Coding For Subprograms
- RPG III For Subfiles
- Data Areas
- Local Data Areas
- Data Structures
- File Exception Errors
- Program Exception Errors
- Multiple Device Files
- Commitment Control
- Program Described Work Station Files
- Printer Device Files

COURSE STRUCTURE

Video presentations are included in the study materials to further clarify and supplement some concepts covered in this course.

Video Presentations

The video presentations are used in the following ways:

- To introduce a topic
- To present a single concept
- To summarize key items

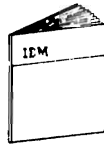


This symbol is used to indicate when you should view a video presentation.

Module Texts

The 13 module texts for this course are listed in the following table. The time shown is the estimated time for the completion of each module. Your time may differ from this time. Remember, one of the advantages of a self-study course is that you can study at your own pace, so take the time to understand the material, even though it may take you longer than estimated.

Module Number and Title	Time, hrs
1 Course Introduction	¾
2 Procedural File Operations	1½
3 Structured Programming Concepts	2
4 Structured Programming Implementation	3
5 Program Compilation and Testing	2
6 Subprograms	2
7 Subfile Programming	4
8 Data Areas	2
9 Data Structures	2
10 Handling Exception Errors	2
11 Additional RPG III Functions	1
12 Program Described Work Station Files	1
13 Printer Device Files	1



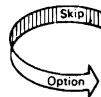
The Module Text is your primary source of course information. It guides you through the material and directs you to the other components of the course. The module texts are used by other students, so please do not write in them.

Each of the Modules is structured in the same manner.

Unit 1 is always an Overview Unit. It contains a purpose that provides a brief explanation as to why the module was written, the estimated time it takes to complete the module, a statement of the objectives that tell what you can expect to be able to do after completing the module, and a list of terms that are introduced in the module.

Unit 2 is a Topical Unit. You will find one or more topical units in a module. Most of the unit consists of detail material and a presentation of the topic. A topical unit ends with a summary of the major items presented in the unit. The summary may take the form of a written summary, a progress check (to be explained later), an exercise, or all three.

The last unit is a Summary Unit. Not every Module Text has one. When used, it will summarize all the units that were in the module text and/or will provide you with a module exercise.



This symbol is used in the module texts to indicate material you may want to skip because you are familiar with the material.

Student Materials Book

The Student Materials Book (SMB) is yours to keep as well as materials used in desk and machine exercises such as coding sheets and program listings. Each of the chapters in the SMB corresponds to one of the module texts. In each chapter, you will find a reprint of the Module Introduction and the Module Summary of the corresponding module text. You may also find illustrations, exercises, progress checks, and summaries relevant to the module.

Each chapter in the SMB contains pages for your personal notes. Use these pages rather than separate sheets of paper so that your notes automatically become part of the chapter for which they were written.

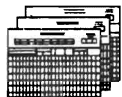
The Student Materials Book provides you with a future reference to the material presented in this course.



The symbol you see at the left is used throughout this course every time a reference is made to the Student Materials Book.

Exercises

To reinforce the material presented in this course, there are exercises for you to perform. These exercises take the form of coding (pencil and paper) exercises and machine (keying) exercises. You are provided with complete instructions in the module text as to when an exercise is to be done. The instructions for all exercises are in the Student Materials Book. Your Administrator has the solutions for the exercises.



This symbol is used to indicate a coding exercise.



This symbol is used to indicate a machine exercise.

Reference Material

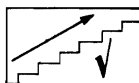
You may be directed to read some reference material. This may take the form of optional reading or additional information for a given unit.



The reference material indicated by the symbol to the left is found in the Guided Learning Center Resource Library and you are directed to it by topic, book title, and form number. Should you need help in locating any information, do not hesitate to ask your Administrator.

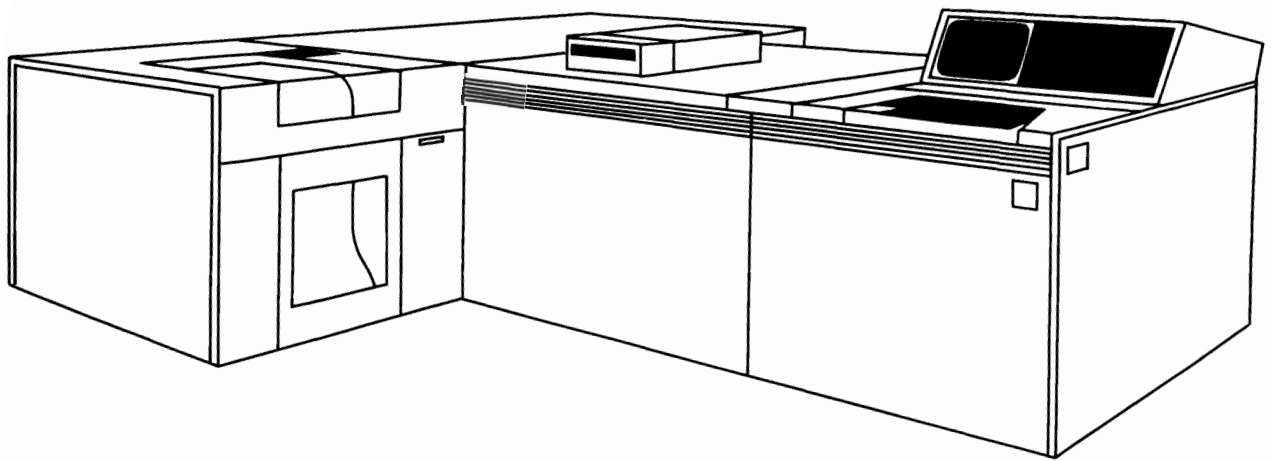
Progress Checks

The summary of a unit is a written review and an activity. The activity may be an exercise to perform or may take the form of something called a Progress Check. A progress check is a series of questions about the material you have just covered.



Progress Checks are identified with this symbol. The questions, along with space to answer them, are found in your Student Materials Book. The answers, along with any appropriate discussion, are found in the module text.

Continue with the Module 1 Course Introduction on the next page.



Chapter 1. Module 1 - Course Introduction

MODULE PURPOSE

This module is a short review and progress check of the pre-requisite material.

TIME ESTIMATE

45 minutes

MODULE OBJECTIVES

After completing this module, you should be able to:

- State four types of DDS entries.
- State the purpose of a subfile.
- State two testing functions.



At this time start your study in the Module 1 text of this course.

Progress Check: Unit 1.

Read each question carefully. Record your answers in the space provided.

1. State how records may be accessed from a data base file.

2. List four types of entries for the DDS when describing data base files?

3. State why the record format name should not be the same as the file name.

4. What are the functions of the Source Entry Utility?

5. Where is the source program stored using the Source Entry Utility?

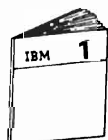
6. How is the RPG III object program given a name?

7. What is the purpose of a subfile?

8. DDS require two record formats for a subfile. What are the two formats and the function of each?

9. Why are breakpoints used in a program?

10. In debug mode, will files located in production libraries be updated?



After you complete the questions, return to the Module 1 Text to review your answers.

Chapter 2. Module 2 - Procedural File Operations

MODULE PURPOSE

This module discusses the operation codes for work station file processing and data base file maintenance. It further shows the flexibility of RPG III enabling you to control input/output operations from calculation specifications within the RPG III logic cycle.

TIME ESTIMATE

1-1/2 hours

MODULE OBJECTIVES

Upon completion of this module with the aid of available reference material you should be able to:

- State two benefits of externally described files.
- Code the file description statements for data base and work station files.
- Name at least one operation code that must precede a data base file update operation.
- List the operations to force records out to the data base.

TERMS

The following terms are defined/described in this module:

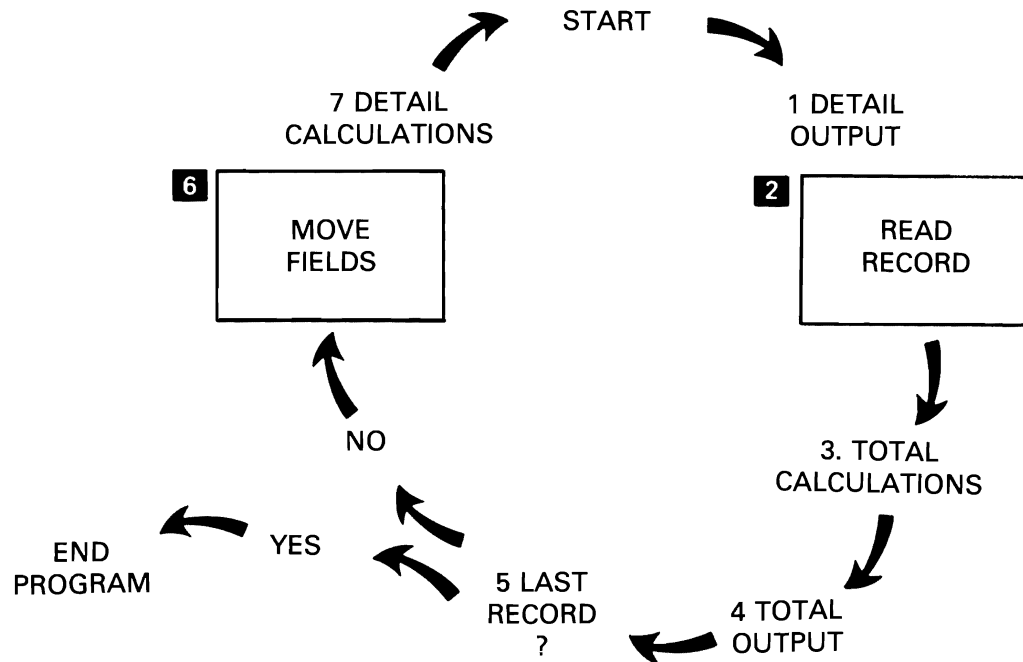
- Full Procedural File
- Program Described File
- Externally Described File
- EXFMT
- UPDAT
- CHAIN
- WRITE
- DELET
- KLIST
- KFLD
- FEOD
- CLOSE
- OPEN

FULL PROCEDURAL PROCESSING

What is it?

It is programmer control of file processing. A full procedural file tells the compiler input is controlled by the programmer in calculation operations. The normal program cycle exists, but file processing occurs at detail or total calculation time.

In the figure below, the shaded areas are bypassed for a full procedural file.



Steps **2 and **6** - segments of the logic cycle that are ignored when full procedural file has been specified as file designation. A file designated as full procedural must have its records read by the calculation specifications.



For additional information, see the "RPG Logic Cycle" chapter in the IBM System/38 RPG III Reference Manual and Programmers Guide, SC21-7725. You will find this reference manual in the Guided Learning Center Resource Library.

UNIT 2 SUMMARY

You select your programming choice through the file description specifications.

FILE HANDLING WITHIN A PROGRAM
1. USING THE PROGRAM CYCLE
2. FULL PROCEDURAL PROGRAMMING

1. USING THE PROGRAM CYCLE
2. FULL PROCEDURAL PROGRAMMING

Both program described and externally described files can be processed in the two ways indicated above. Full procedural and externally described choices are independent of each other. In other words, they can be used together in a program or one without the other.

The file description specification also determines how data is referenced.

- | |
|---|
| <ul style="list-style-type: none">• PROGRAM DESCRIBED<ul style="list-style-type: none">— DATA IS ACCESSED BY FILE NAME ONLY— INPUT/OUTPUT SPECIFICATIONS ARE REQUIRED• EXTERNALLY DESCRIBED<ul style="list-style-type: none">— DATA IS ACCESSED BY FILE NAME <u>OR</u> RECORD FORMAT NAME— INPUT/OUTPUT SPECIFICATIONS ARE <u>NOT</u> REQUIRED |
|---|

Progress Check: Unit 2

Record your answers in the space provided.

- When using full procedural file processing why must the LR indicator be set on to end the program?

- List two benefits of externally described files.

- Using the coding sheet below, code: the file descriptions for:

- A full procedural data base file named POLITM used for file maintenance (add, change, delete).
- A work station input/output file named PODUPD.

Both files have been externally described.

Form Type	Filename	File Type					Mode of Processing							Device	File Addition/Unordered																																																						
		File Designation					Length of Key Field or of Record Address Field								Number of Tracks for Cylinder Overflow																																																						
		End of File					Record Address Type								Number of Extents																																																						
		Sequence					Type of File Organization or Additional Area								Tape Rewind																																																						
		File Format					Overflow Indicator								File Condition U1-U8, UC																																																						
	Block Length					Record Length					L/R		A/P/I/K		I/X/D/T/R/?			Key Field Starting Location		Extension Code E/L			A/U		R/U/N																																												
	External Record Name																																																																				
	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	66	67	68	69	70	71	72	73	74																		
F																																																																					
F																																																																					
F																																																																					

WORK STATION FILE PROCESSING

You may use the following operations to process your work station files.

WORK STATION INPUT/OUTPUT OPERATION		
Operation Code	Program Cycle	Full Procedural
EXFMT		X
READ		X
WRITE	X	X
OPEN	X	X
CLOSE	X	X
FEOD	X	X

Progress Check: Unit 3

Record your answers in the space provided.

1. Why does ADD on an output specification provide you with more control than WRITE in calculation specifications?

2. Name at least one operation code that must precede the UPDAT operation code and why?

3. Why must you reposition the file after an unsuccessful READ?

4. What are the operation codes to force data base records to auxiliary storage?



When you have completed the questions, return to Module Text 2 to review your answers.

MODULE SUMMARY

Module 2 has illustrated that the need for input and output specifications is reduced or eliminated using RPG III. The file processing operations allow flexibility and control.

- Full procedural processing.
- Records may be updated in calculations or on output specifications.
- Records may be accessed by file name or record format name (choice only for externally described files).
- You can open your own files or let RPG III do it.
- File pointer can be set/reset by operation codes.
- Files can be logically closed by FEOD.

You now know how to control what records are read and when they are read. Next, you will look at a method of providing this control when full procedural file processing is used.

The following chart summarizes the operation codes discussed in module 2.

OPERATION				
OP CODE	PROGRAM CYCLE	FULL PROCEDURAL	WORK STATION	DATA BASE
CHAIN		X		X
CLOSE	X	X	X	X
DELET	X	X		X
EXFMT		X	X	
FEOD	X	X	X	X
KLIST	X	X		X
KFLD	X	X		X
OPEN	X	X	X	X
READ		X	X	X
UPDAT	X	X		X
WRITE	X	X	X	X

Chapter 3. Module 3 - Structured Programming Concepts

MODULE PURPOSE

Module 3 discusses the theory and the practical application of structured programming. The module further shows why structured programming was developed and what benefits you should achieved by using it.

TIME ESTIMATE

2 hours

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- List the elements of a proper program.
- Name and draw the three node types.
- Use the nodes to construct the five structured programming figures.
- Using structured programming figures, draw a structured flowchart.

TERMS

These terms are described/defined within the module.

- Structured Theorem
- Boolean Logic
- Proper Program
- Node
- Predicate
- Collector
- Function
- IFTHENELSE
- DOWHILE
- DUNTIL
- CASE



VIDEO INTRODUCTION

Structured programming is a style of programming in which the program is organized based on the structure of a simple sequence of functions, selection of functions, and loop control.


The video you are about to see introduces the concept of structured programming and the building blocks of the structured programming figures.

Video Summary

The video presented the fundamental elements (nodes) of the structured programming figures:

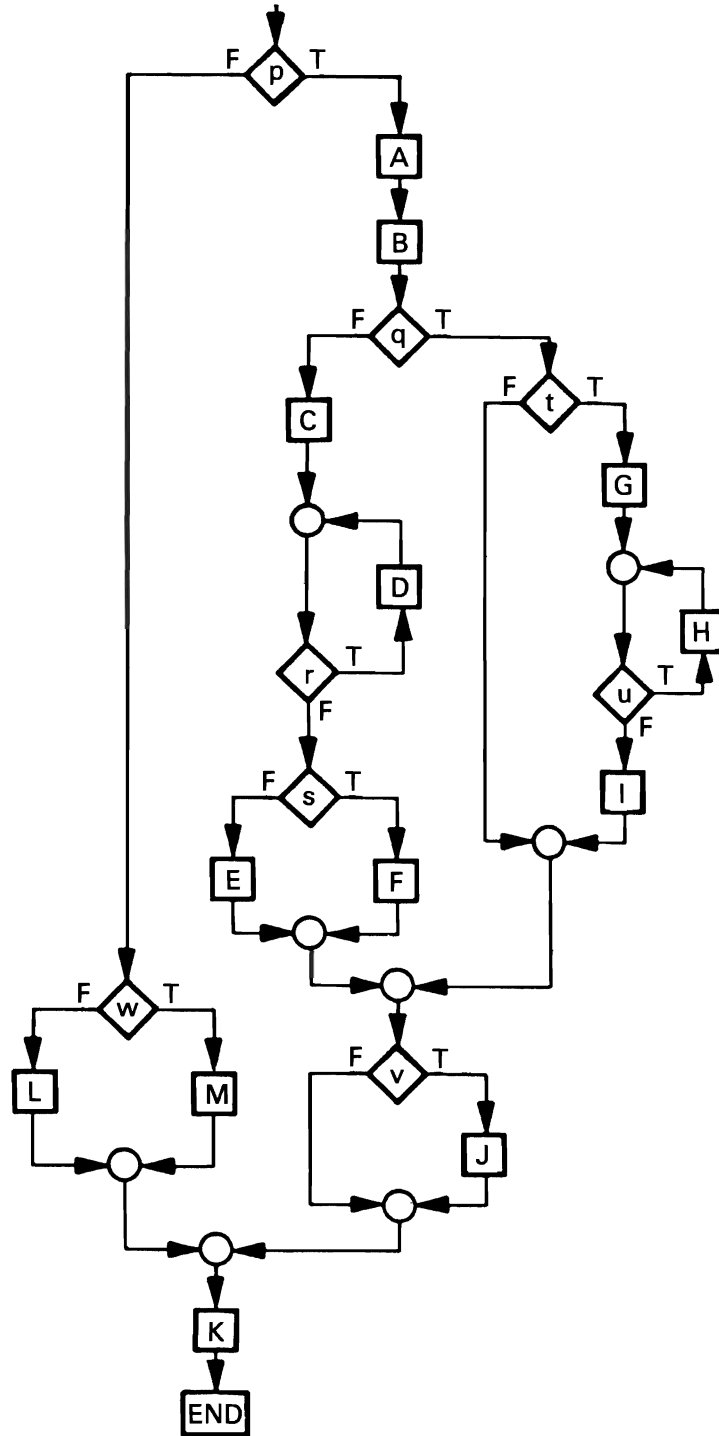
- the function or process mode
- the predicate mode
- the collector node

The remainder of this module concentrates on using these control figures in developing a structured programming flow-chart for a data base application.



FLOWCHART

STRUCTURED PROGRAM



Progress Check

Record your answers in the space provided.

1. What are the two main characteristics of a proper program?

2. Name and draw the three fundamental nodes that are used to construct the structured programming logic figures.

3. Name and draw the structured programming logic figures, using the fundamental nodes.



When you have completed the questions, return to Module Text 3 to review your answers.

Desk Exercise

Structured Program Flowchart: Open Purchase Order File Update.

The purpose of this exercise is for you to become familiar with the structured programming technique. You may wish to refer to the example that we did in Module Text 3 as you do this exercise.

This exercise is a file update program of an Open Purchase Order file. The system has an Open Purchase Order file containing outstanding purchase orders. The Open Purchase Order file is a logical file POLITM which is built over two physical files: a purchase order summary file, POPSUM, and a purchase order detail file, POPDTL.

See diagram on next page for illustration of the logical and physical files.

There is one summary record and possibly multiple detail records for each purchase order in the Open Purchase Order file. A summary record always has at least one detail record. As the merchandise arrives in the receiving department, a clerk uses the Open Purchase Order File Update program to enter the quantity received.

DISPLAY RECORD LAYOUT

Display Screen Layout Sheet

		COLUMN																																																																															
		1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80																																																																								
ROW	01																																																																																
	02																																																																																
	03																																																																																
	04																																																																																
	05																																																																																
	06																																																																																
	07																																																																																
	08	GOOD MORNING. TODAY IS XX/XX/XX																																																																															
	09																																																																																
	10	PLEASE ENTER MERCHANDISE DELIVERED																																																																															
	11																																																																																
	12																																																																																
	13	PURCHASE ITEM QUANTITY																																																																															
	14	ORDER NO. NUMBER DELIVERED																																																																															
	15																																																																																
	16	<u>XXXXXX</u> <u>XXXXX</u> <u>XXXXXX</u>																																																																															
	17																																																																																
	18																																																																																
	19																																																																																
	20	TO END THE JOB, PRESS CMD KEY 1																																																																															
	21																																																																																
	22																																																																																
	23																																																																																
	24																																																																																

The following blank pages are provided for you to draw your flowchart.

FLOWCHART EXERCISE



FILES

```

SOURCE FILE:      RPGSRC.GLCRPG                MEMBER:  POLITM
SEQNBR#... .. 1 ... .. 2 ... .. 3 ... .. 4 ... .. 5 ... .. 6 ... .. 7 ...
100      A          R ORDSUM                    PFILE(POPSUM)
200      A          K PCRNBR
300      A          R ORDDTL                    PFILE(PCPDTL)
400      A          K PCRNBR
500      A          K ITMNR
    
```

```

SOURCE FILE:      RPGSRC.GLCRPG                MEMBER:  POPSUM
SEQNBR#... .. 1 ... .. 2 ... .. 3 ... .. 4 ... .. 5 ... .. 6 ... .. 7 ...
100      A          REF(APPREF)
200      A          R ORDSUM                    TEXT(*PO SUMMARY RECORD*)
300      A          PCRNBR                      R
400      A          VNCNBR                      R
500      A          VNDNAM                      R
600      A          MERCH                      R
700      A          DATOR                      R
800      A          VNSLS                      R
900      A          VNDACD                     R
1000     A          VNDPHN                     R
1100     A          STATUS                     R
    
```

```

SOURCE FILE:      RPGSRC.GLCRPG                MEMBER:  POPDTL
SEQNBR#... .. 1 ... .. 2 ... .. 3 ... .. 4 ... .. 5 ... .. 6 ... .. 7 ...
100      A          REF(APPREF)
200      A          R ORDDTL                    TEXT(*PO LINE ITEM RECORD*)
300      A          PCRNBR                      R
400      A          VNDNBR                      R
500      A          ITMNR                      R
600      A          ITMDCS                     R
700      A          QTYCRD                     R
800      A          ITMCST                     R
900      A          ITMAMT                     R
1000     A          DEPTNC                     R
1100     A          CATNBR                      R
1200     A          DATREC                     R
1300     A          QTYREC                     R
1400     A          STATUS                     R
    
```


Given the following program requirements, structurally flow-chart the program logic for the Purchase Order Update program.

1. The program initially displays a prompt screen which allows the operator to enter the purchase order number, the item number, and the quantity delivered or to end the job. The prompt screen layout is on the next page.
2. The program is to validate the purchase order number and item number that are entered.
 - a. First access the detail record which is keyed on purchase order number and item number. If the detail record cannot be found, access the summary record (which is keyed on purchase order number).
 - b. If neither the detail nor the summary record can be found, display the message: "Invalid purchase order number."
 - c. If the summary record can be found but not the detail record, display the message: "Invalid item number."
3. If the detail record is found:
 - a. Update the quantity received field (QTYREC) in the detail record by adding to it the quantity delivered (QTYDEL) that was entered by the operator.
 - b. If the quantity received (QTYREC) is equal to the quantity ordered (QTYORD), update the status code with a "C" and the date received with today's date.
 - c. Update the detail record.
4. Redisplay the prompt screen.

FLOWCHART EXERCISE

MODULE SUMMARY

The goal of structured programming is to provide a systematic, quality controlled method of writing programs.

The use of structured programming brings the opportunity of greater precision and reliability to programming than has been attainable in the past. This occurs because its use results in programs which can be more easily read, modified and maintained by other programmers. As you have studied the structured programming conventions, you should use the basic structures for:

- Sequential functions
 - SEQUENCE
- Conditional operations
 - IFTHENELSE
 - CASE
- Repetition
 - DOWHILE
 - DUNTIL

The next module (Module Text 4) discusses implementing the structured programming theory in RPG III - how your flow-chart is converted into RPG III code.

Chapter 4. Module 4 - Structured Programming Implementation

MODULE PURPOSE

Module 4 illustrates how structured programming is implemented in RPG III.

TIME ESTIMATE

2 hours

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- List 4 benefits of structured programming.
- Write the difference between the DOWHILE and the DOUNTIL operations.
- Code an RPG III structured program from a structured flow-chart and the associated file and record information.

TERMS

These operation codes are described/defined within the module.

- IFxx
- ELSE
- END
- DOUxx
- DOWxx
- CASxx





VIDEO INTRODUCTION

Structured programming is a style of programming in which the program is organized based on structured programming logic figures.

The video you are about to see illustrates the RPG III operation codes for these structured programming logic figures.

Video Summary

The video presented the structured programming figures with their associated operation codes. The presentation described and illustrated the use of the structured statements.



UNIT 2 SUMMARY

You have studied the relation of the RPG III structured operation codes to the structured logic figures. These figures provide a set of guidelines which when understood and followed result in the creation of structured programs.

It is entirely possible that you have been organizing your programs into a logical structure. In that case, you are already using a form of structured programming. However, if everyone uses the same convention or guidelines, then it would be easier for us to read and understand anyone's programs.

Progress Check: Unit 2

Record your answers in the space provided.

1. List two ways in which structured programming aid the programmer in debugging and maintaining a program?

2. How does a DOWHILE differ in logic from DOUNTIL?

3. List at least four programming benefits of structured programming.

4. What instructions are executed if the test is false in an IFxx operation?

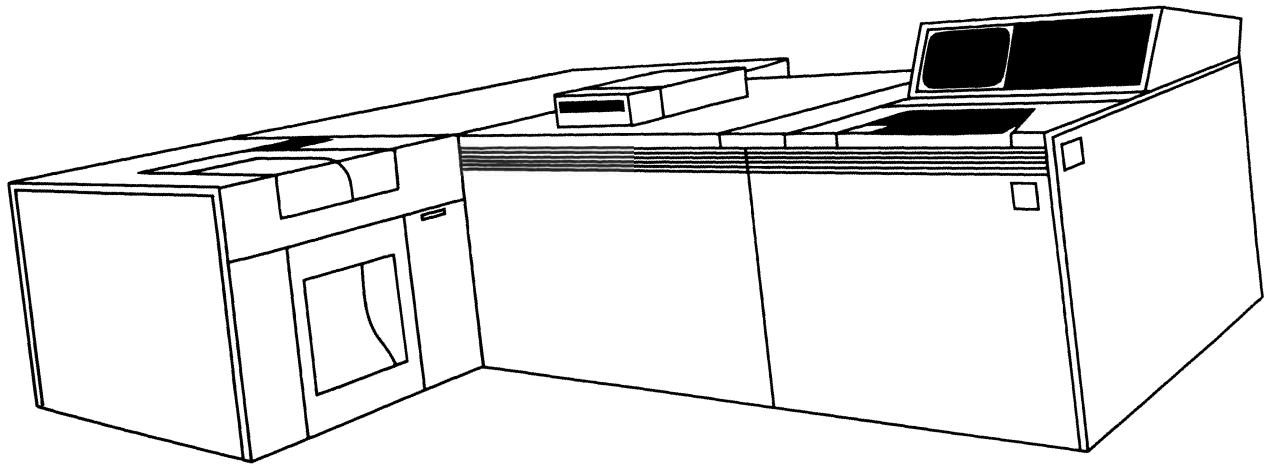


When you have completed the questions, return to Module Text 4 to review your answers.

NOTES

Lined area for notes, consisting of horizontal ruling lines across the page.





Desk Exercise – Open Purchase Order File Update

The purpose of this exercise is for you to use the RPG III operation codes that support structured programming (DOWxx, IFxx, etc.) and to become familiar with the function and coding of those operation codes.

You may use either the structured flowchart you developed in the Desk Exercise for Module 3 or the flowchart and HIPO charts given on the next two pages.

1. Name your program POR1xx

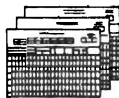
NOTE: Replace xx with your first and last initials. This is true throughout this course. Do not confuse this xx with the xx on the structured programming operation codes.

2. The display file specifications are provided for you in the RPGSRC source file in the member POD010. See the screen layout and the data description specifications following the HIPO charts. Name your display file POD1xx.
3. The Open Purchase Order file POLITMxx you will update is a logical file based on the two physical files POPSUMxx (summary records) and POPDTLxx (detail records). You will have your own set of files.
4. See the data descriptions specifications for these files in this exercise.

The field names you use are as follows:

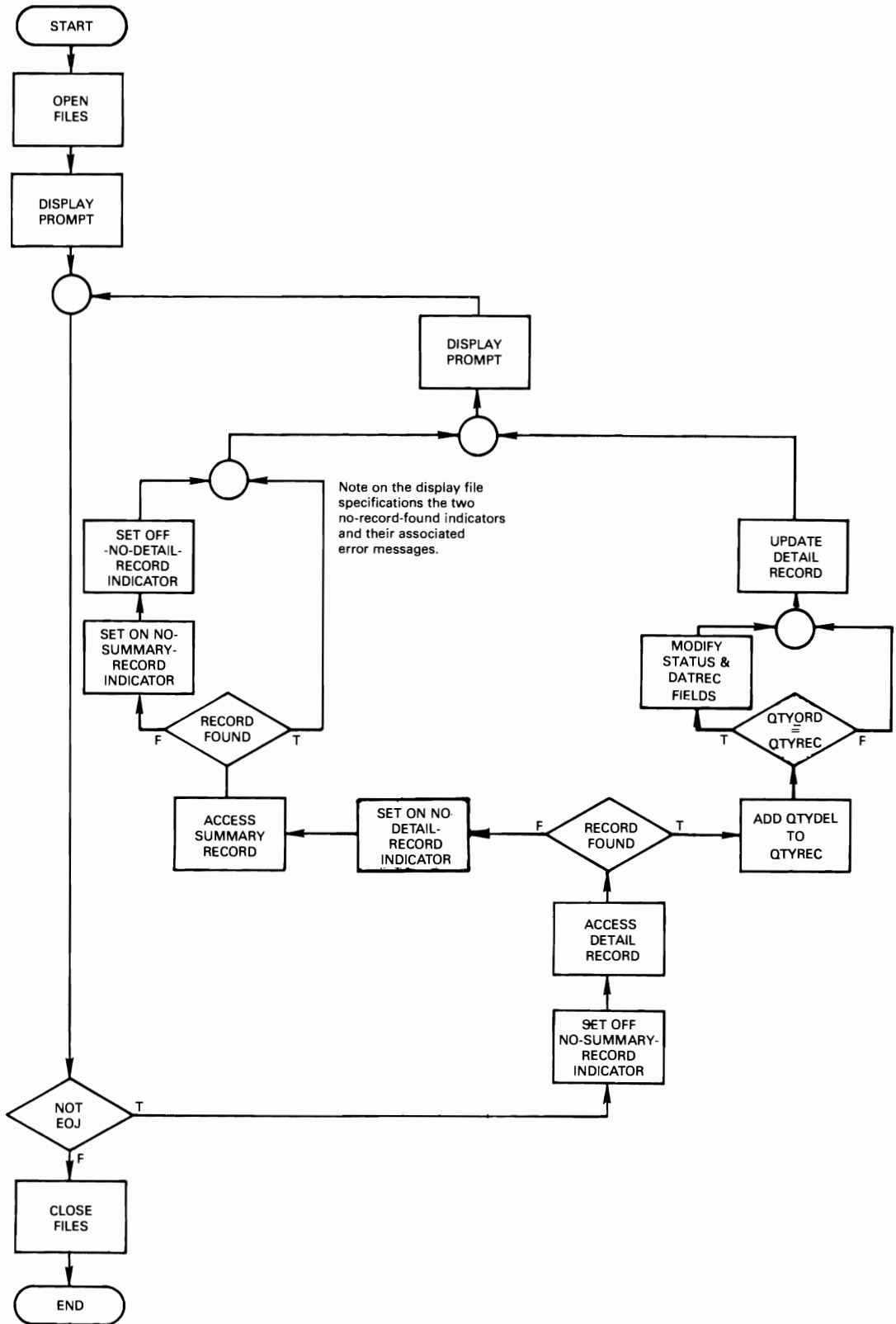
Purchase Order Number	PORNBR
Item Number	ITMNBR
Quantity Delivered	QTYDEL
Quantity Received	QTYREC
Quantity Ordered	QTYORD
Date Received	DATREC
Status of Record	STATUS

5. You will enter, compile and execute this program in Module 5.
6. When you are finished coding, you may wish to review your program with a solution that the Administrator has.

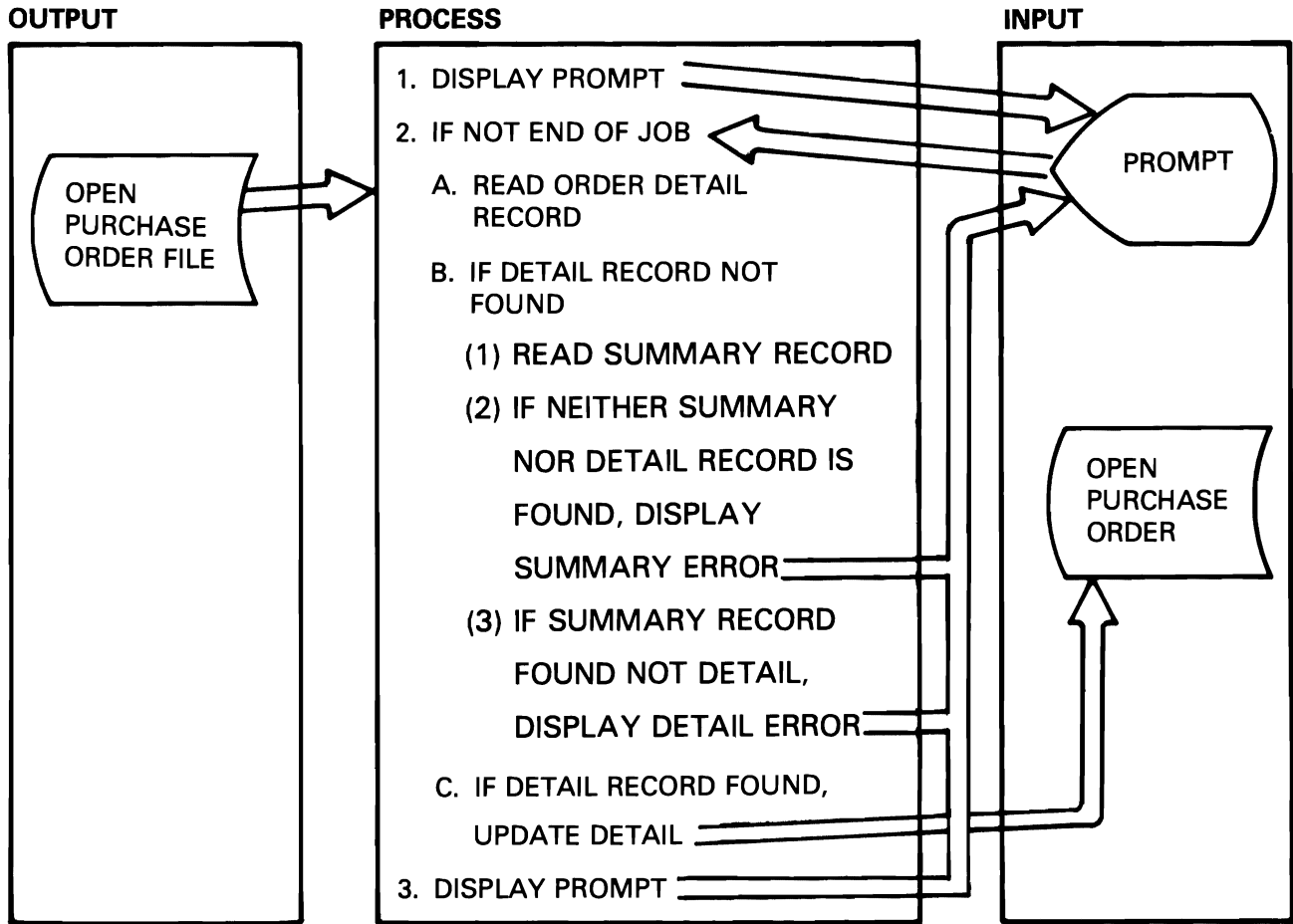


You will need file description and calculation specification coding sheets for this exercise. Please ask your Administrator for these forms.

MODULE 3 DESK EXERCISE SOLUTION: STRUCTURED FLOWCHART



DETAIL DIAGRAM



EXTENDED DESCRIPTION

NOTE	REFERENCE
DISPLAY PROMPT FOR INPUT DATA (PURCHASE ORDER NUMBER ITEM NUMBER, AND QUANTITY DELIVERED).	1
IF END OF JOB, EXIT PROGRAM.	2
READ ORDER DETAIL RECORD KEYED ON P.O. AND ITEM NUMBERS.	2.A
IF DETAIL RECORD NOT FOUND, ACCESS SUMMARY RECORD KEYED ON P.O. NO.	2.B.1
IF NEITHER DETAIL NOR SUMMARY, DISPLAY "INVALID PURCHASE ORDER NUMBER."	2.B.2
IF SUMMARY NOT DETAIL, DISPLAY "INVALID ITEM NUMBER."	2.B.3
IF DETAIL RECORD IS FOUND, UPDATE QTYREC FIELD WITH QTYDEL FIELD. IF QTYREC = QTYORD, UPDATE STATUS CODE WITH A "C" AND DATE RECEIVED WITH TODAY'S DATE.	2.C
REDISPLAY PROMPT SCREEN.	3

FILES

PROMPT DISPLAY - PURCHASE ORDER UPDATE

Display Screen Layout Sheet

		1-10		11-20		21-30		31-40		41-50		51-60		61-70		71-80					
		1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
01																					
02																					
03																					
04																					
05																					
06																					
07																					
08																					
09																					
10																					
11																					
12																					
13																					
14																					
15																					
16																					
17																					
18																					
19																					
20																					
21																					
22																					
23																					
24																					

```

MEMBER:  POD010

... 1 ... 2 ... 3 ... 4 ... 5 ... 6 ... 7 ... 8

A          REF(APPFREF)
A          CA01(01 'END OF JOB')
A          R PRCPMT
A          08 21'GOOD MORNING!  TODAY IS'
A          08 45DATE  EDTCDE(Y)
A          10 21'PLEASE ENTER MERCHANDISE DELIVERED'
A          13 21'PURCHASE      ITEM      QUANTITY'
A          14 21'ORDER NO.    NUMBER    DELIVERED'
A          16 22CHECK(ME) CMP(NE 0)
A 10          ERRMSG('INVALID PURCHASE ORDER +
A          NUMBER')
A          16 35CHECK(ME) CMP(NE 0)
A 20          ITMNBR      R          I 16 35CHECK(ME) CMP(NE 0)
A          ERRMSG('INVALID ITEM NUMBER')
A          QTYDEL      5  OI 16 48CHECK(ME) CMP(NE 0)
A          20 21'TO END THE JOB, PRESS CMD KEY 1.'
    
```

FILES

DATA BASE

POLITMMB

SOURCE FILE: RPGSRC.GLCRPG MEMBER: POLITM

SEQNBR#... .. 1 2 3 4 5 6 7 ...

100	A	R	ORDSUM	PFIL	(POPSUM)
200	A	K	PORNBR		
300	A	R	ORDDTL	PFIL	(POPDTL)
400	A	K	PORNBR		
500	A	K	ITMNBR		

physical file

POPSUMMB

SOURCE FILE: RPGSRC.GLCRPG MEMBER: POPSUM

SEQNBR#... .. 1 2 3 4 5 6 7 ...

100	A			REF	(APPFREF)
200	A	R	ORDSUM	TEXT	(*PO SUMMARY RECORD*)
300	A		PORNBR		R
400	A		VNDNBR		R
500	A		VNDNAM		R
600	A		MERCH		R
700	A		DATOR		R
800	A		VNDSLS		R
900	A		VNDACD		R
1000	A		VNDPHN		R
1100	A		STATUS		R

physical file

DTLMB

SOURCE FILE: RPGSRC.GLCRPG MEMBER: POPDTL

SEQNBR#... .. 1 2 3 4 5 6 7 ...

100	A			REF	(APPFREF)
200	A	R	ORDDTL	TEXT	(*PO LINE ITEM RECORD*)
300	A		PORNBR		R
400	A		VNDNBR		R
500	A		ITMNBR		R
600	A		ITMDSC		R
700	A		QTYORD		R
800	A		ITMCST		R
900	A		ITMAMT		R
1000	A		DEPTNO		R
1100	A		CATNBR		R
1200	A		DATREC		R
1300	A		QTYREC		R
1400	A		STATUS		R

MODULE SUMMARY

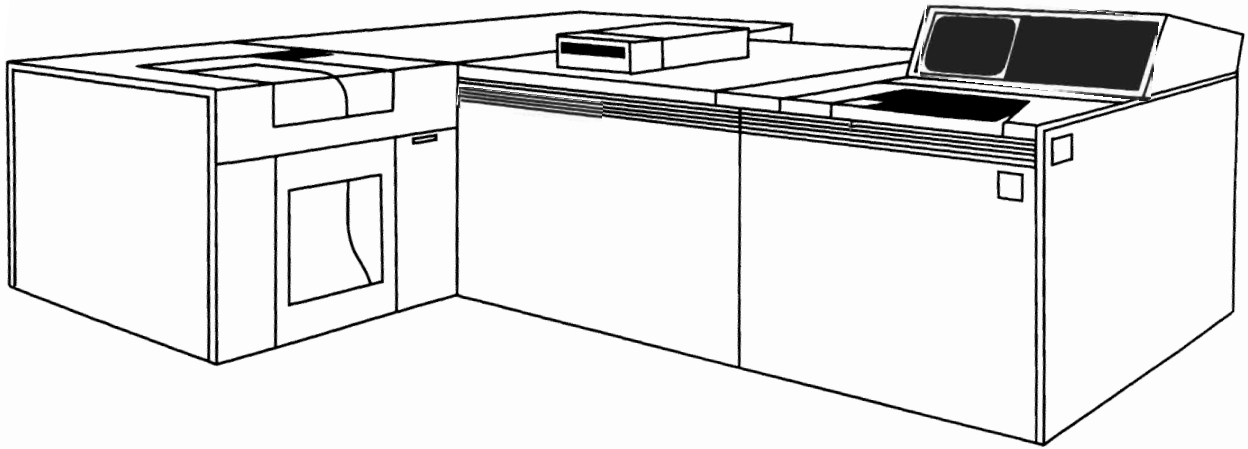
Module 4 discussed the RPG III operation codes for the structured programming figures. You are now at the stage where you can begin to test your structured programming techniques.

Flowcharting does add time to the design process, however the coded program should have fewer errors when tested.

The total number of compilations should be reduced as a result of your taking the time to think through your program logic.

Structured programming is a programmer's tool. Another tool available to you as a programmer is the RPG III compiler.

Module 5 discusses the compiler options, the information on the compiler listing, and the System/38 Messages Guide.



Chapter 5. Module 5 - Program Compilation and Testing

MODULE PURPOSE

Module 5 discusses the information contained on the RPG III compiler listing, options available on the Control Language Create RPG Program (CRTRPGPGM) command, and the use of the System/38 Messages Guide.

TIME ESTIMATE

3 hours and 30 minutes

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- Identify the sections of the compiler listing.
- Use the listed information in debugging, if needed.
- List two reasons why it would be necessary to reference the System/38 Messages Guide.
- Compile, test, and debug RPG III programs.

TERMS

These terms are described/defined within the module.

- Cross Reference Listing
- Decimal Data Error
- Level Checking
- Generation Level



MACHINE EXERCISES INTRODUCTION

The exercises used in this course are based on an Accounts Payable/Purchase Ordering application. This application contains realistic requirements that have been simplified - so that the exercises are practical and yet workable within the constraints of this course.

The following naming conventions are to be used in this course for your objects:

For programs - AALSxx

Where AA is the application

Example: AP - Accounts Payable
PO - Purchase Orders
BI - Billing

L is the source language

Example: R - RPG III
L - Control Language
C - COBOL
D - DFU
Q - Query

S is the program number.

xx is your identification (first initial of your first and last names.)

Example: APR4BT is Accounts Payable program number 4 for student BT (Bill Tate) written in RPG III.

For files - AATDDDxx

Where AA is the application

Example: PO - Purchase Orders
SA - Sales Analysis
IN - Inventory

T is the file type

Example: P - Physical Data Base File
L - Logical Data Base File
D - Display File
W - Printer File

DDD is the alphameric file name

Example: SUM - Summary File
DTL - Detail File

You may use the following objects in the GLCRPG library:

APPFREF - Accounts Payable/Purchase Order field reference file

APLDUE - Logical Open Vendor Invoice File - keyed on invoice number within due month/day within due year

APPOPY - Open Vendor Invoice File - in arrival sequence

POPSUM - Purchase Order Summary File - in arrival sequence

POPDTL - Purchase Order Detail File - in arrival sequence

POLITM - Logical Purchase Order File - built over the two physical files POPSUM and POPDTL. The summary records (from POPSUM) are keyed on purchase order number and the detail records (from POPDTL) are keyed on item number within purchase order number.

POLONO - Logical Purchase Order File - built over the two physical files POPSUM and POPDTL. The file has an access path keyed on purchase order number.

RPGSRC - Source File - contains all of your source members for the machine exercises; each member will have the same name as the object you are creating.

The library you will be working with is the GLCRPG library. Your source file is RPGSRC.

Use APPFREF field reference file for field definitions. The field reference file is listed in Appendix A.



Other objects that you will be using in the machine exercises:

- Job Description - GLCRPG
- Output Queue - GLCOUTQ
- Profile and Password - GLCRPG

If you wish to know more about your field reference file, job description, and user profile, please look in Appendix A of this book.

Please note the following for all exercises:

All objects are to be created in the GLCRPG library. All source members are to be created in RPGSRC source file of the GLCRPG library.

You will get the Programmer Menu shown below when you sign on with the password GLCRPG to do your machine exercises. Note that all the entries described above are already on this menu and that you will be using the GLCRPG job description for your batch jobs.

```

                                PROGRAMMER MENU                                SYSTEM: ATLEDDEV
Select one of the following:
1. Design/execute DFU app      (app), ,(options)
2. Design/execute query app   (app), ,(options)
3. Create object              object name, type, pgm for CMD, (text)
4. Call program                program name
5. Execute command            command
6. Submit job                  (job name), (command)
7. Display submitted jobs
8. Edit source                 (srcmbr), (type), (text)
9. Design display format      (srcmbr)
80. Display Menu              (menu)
90. Sign off                   (*NDLIST *LIST)
Types: BAS, BASP, BSCF, CBL, CL, CLP, CMD, CMNF, DFU, DSPF, LF, MXDF, PF, PLI,
      PRTF, QRY, RPG, RPT, TXT
Option: __ Parm: _____ Type: ____ Parm 2: _____
Command: _____
-----
Text: _____ Log requests: *YES
Src file: RPGSRC      Src lib: GLCRPG      Obj lib: GLCRPG      Jobd: GLCRPG
CF3-Command entry    CF4-Prompt (3,5 & 6 only)  CF6-DSPMSG
```

After you have signed on, select option 5 and enter the following command:

CALL RPGID

The program RPGID presents a display for you to enter your first and last initials in order to uniquely identify your exercise material in the GLCRPG library. If the initials you enter are already being used, you will be asked to enter different identification.

The program produces a report similar to the sample below.

```
WELCOME TO THE SYSTEM/38 RPG III AND STRUCTURED PROGRAMMING COURSE

START OF MACHINE EXERCISES
YOUR IDENTIFICATION IS: (XX)
                        3/17/83
                        17:18:52
```

Your compile listings and reports are printed on the work station printer in the Guided Learning Center. Please remove the report from the printer and give this report to your Administrator.

All references to the suffix xx in the Student Materials Book refer to your identification.

Machine Exercise - Open Purchase Order File Update

In this exercise you are to enter, compile and execute the Open Purchase Order File Update program you coded in Module 4. Before you can compile your program, you must create the files used by your program. The first seven steps of the following instructions cover file creation.

Please do not continue with this exercise until you have read and followed the instructions above.

You may find it helpful to have the Source Entry Utility (SEU) Reference Manual and User's Guide (SC21-7722) open to Chapter 5 when you are instructed to use the Services display. Also, don't forget about the HELP key when you need more information about a display.

1. On the Programmer Menu select option 8, enter the name POD1xx for your display file and DSPF for the type. You are creating a new source member for your display file in the RPGSRC source file.
2. When your empty source member is displayed, press Command key 5 (CMD 5) to get the Services display of SEU. Now use Browse/Copy to copy all the data description specifications from the source member POD010 in the RPGSRC file into your member.

After a successful copy, press CMD 5 to get to the Services display. Exit the Services display by entering N after Browse/Copy. Exit SEU with CMD 1 and by selecting option 2 on the Exit display. You may want to request a listing of your source member with the Exit display.

3. Create your display file POD1xx in the GLCRPG library by selecting option 3 on the Programmer Menu. All the other information needed is already on the display.
4. Next, you create your own set of data files for this exercise. To create your two physical files, POPSUMxx and POPDTLxx, select option 5 and enter the following command. Be sure to enter the () and your initials with PARM.

CALL LAB1 PARM(xx)

5. Select option 8 to create a source member in the RPGSRC file for your logical file to be named POLITMxx. Be sure to specify the type as LF.
6. As you did in Item 2, use the Services display to copy all the data description specifications from the member POLITM in the RPGSRC file into your member. After a successful copy, return to the edit source display and modify the PFILE parameters by changing the POPSUM and POPDTL file names to your file names POPSUMxx and POPDTLxx. The following example shows the changes.

FILE NAME -	POLITMWW.GLCRPG	TYPE OF FILE -	LOGICAL	
SOURCE FILE -	RPGSRC.GLCRPG	MEMBER -	POLITMWW	83/05/05
TYPE OF DATA -	*DATA			
OPTIONS -	*SRC *LIST			
AUTHORITY -	*NORMAL			
TEXT -				
COMPILER -	IBM SYSTEM/38 DATA DESCRIPTION PROCESSOR			
	DATA DESCRIPTION SOURCE			
SEQNBR	*... .. 1	2	3	4
	5	6	7	
100	A	R ORDSUM	PFILE(POPSUM <u>xx</u> .GLCRPG)	
200	A	K PORNBR		
300	A	R ORDDTL	PFILE(POPDTL <u>xx</u> .GLCRPG)	
400	A	K PORNBR		
500	A	K ITMNB		

7. Exit SEU as you did in Item 2 and create your logical file POLITMxx in the GLCRPG library by selecting option 3 on the Programming Menu. Get the listing for the file creation to be sure it was successfully created.
8. Select option 8 on the Programmer Menu in order to enter your program POR1xx as a member in the source file RPGSRC. Be sure you enter your specifications in upper case because the RPG III compiler will not accept lower case specifications. When you have entered your program, exit SEU. Remember you can request a listing of your program from the Exit menu, which you may want to do in order to check your program before going on to the next step.
9. Create your object program POR1xx in the GLCRPG library by selecting option 3 on the Programmer Menu. Be sure your program compiles successfully by checking the compiler listing and the messages (CF6-DSPMSG). Do not proceed until it does.
10. Run your program by entering option 4 on the Programmer Menu. Enter the following receipts. Remember to press Field Exit after you enter each value to properly adjust the value in the field.

<u>Purchase Order No.</u>	<u>Item Number</u>	<u>Quantity</u>
400005	50	5
400010	75	500
400015	275	2

In addition to the above entries, test your program's error logic by entering an invalid purchase order number of 999999 and then an invalid item number of 99999.

The following are suggestions if your program does not run correctly.

If your program seems to be in an endless loop, use the SYS REQ key, then press ENTER and cancel your job from the System Request display.

Usually you can find and correct errors using your program listing. Sometimes, though, you may find it helpful to run your program in debug mode to locate a problem. If you do, you need to specify UPDPROD (YES) with the ENTDBG command. For example:

```
ENTDBG PGM (POR1xx) UPDPROD (*YES)
ADDBKP STMT (          )
```

Also, be sure to execute the ENDDBG command after you finish testing your program.

11. If you need fresh data in your files (you must have created the files as previously described), enter the following command:

```
CALL RESTR1 PARM (xx)
```

12. After you have successfully run and exited your program, select option 5 and enter the following command:

CALL LIST1 PARM (xx)

The program LIST1 prints a formatted listing of your logical file POLITMxx. Sample before and after formatted listings are shown below.

BEFORE

EXERCISE 1 TEAMW*									
PO NUMBER	VENDOR NUMBER	VENDOR NAME	ORDER AMOUNT	STATUS	ITEM NUMBER	ITEM DESCRIPTION	QUANTITY ORDERED	QUANTITY RECEIVED	DATE RECEIVED
400001	10	JOHN M. SMITH & SONS	500.00	C	25	LINED PAPER	25		C/C0/CC
					150	BEAKERS	100		C/C0/CC
					375	EGGERS	30	30	3/15/80
					475	RIBBONS	100	100	2/26/80
400005	11	FETZNER & FETZNER	110.00	C	50	SPOOLS	10	5	2/22/80
					100	INVITATIONS	15	15	2/12/80
					200	LADDERS	6	6	2/24/80
					400	FILES	8	8	2/16/80
400010	16	BRAND X BEER	350.00	C	75	BOTTLES	500		C/C0/CC
					450	CANS	25		C/C0/CC
400015	431	MARGARET HART INC.	121.00	C	175	CARPETS	3	3	2/22/80
					250	SPINDLES	5	5	2/17/80
					275	MCLOS	2		C/C0/CC
400020	5075	PETER VAN ROTH CORP.	169.50	C	300	DISPLAYS	6	6	1/23/80
					425	CABINETS	9		C/C0/CC
400025	7374	PHOTO LABS INC.	45.50	C	500	PICTURES	13	13	2/05/80

AFTER

EXERCISE 1 TEAMW*									
PO NUMBER	VENDOR NUMBER	VENDOR NAME	ORDER AMOUNT	STATUS	ITEM NUMBER	ITEM DESCRIPTION	QUANTITY ORDERED	QUANTITY RECEIVED	DATE RECEIVED
400001	10	JOHN M. SMITH & SONS	500.00	C	25	LINED PAPER	25		C/C0/CC
					150	BEAKERS	100		C/C0/CC
					375	EGGERS	30	30	3/15/80
					475	RIBBONS	100	100	2/26/80
400005	11	FETZNER & FETZNER	110.00	C	50	SPOOLS	10	10	2/05/80
					100	INVITATIONS	15	15	2/12/80
					200	LADDERS	6	6	2/24/80
					400	FILES	8	8	2/16/80
400010	16	BRAND X BEER	350.00	C	75	BOTTLES	500	500	2/05/80
					450	CANS	25		C/C0/CC
400015	431	MARGARET HART INC.	121.00	C	175	CARPETS	3	3	2/22/80
					250	SPINDLES	5	5	2/17/80
					275	MCLOS	2	2	2/05/80
400020	5075	PETER VAN ROTH CORP.	169.50	C	300	DISPLAYS	6	6	1/23/80
					425	CABINETS	9		C/C0/CC
400025	7374	PHOTO LABS INC.	45.50	C	500	PICTURES	13	13	2/05/80



When you have completed the Open Purchase Order Update machine exercise, return to Module Text 5 to continue the course.

Progress Check: Unit 2

Record your answers in the space provided.

1. What information does the Cross-Reference listing provide?

2. How can you change the terminal severity level of a program?

3. List the reasons why it would be necessary to reference the System/38 Messages Guide?

Coding Exercise - Open Purchase Order File Maintenance

The purpose of this exercise is for you to use the RPG III file processing operation codes WRITE, DELET and UPDAT. You are to use structured programming to implement your program logic.

This exercise is an Open Purchase Order File Maintenance program. In Machine Exercise 1 you updated the open purchase orders with the quantity received. Now you are to maintain that file: 1) by adding a new item to an existing purchase order, 2) by deleting an item from a purchase order, or 3) by modifying the quantity ordered of an item in a purchase order. You use the same data base files used in the file update exercise.

You will find documentation (flowchart, screen layout, display DDS coding) on the following pages of the exercise.

1. Develop your program logic using the supplied structured programming flowchart.
2. The main functions of the maintenance program are to process the three types of transaction.
 - A. To add a new item record:

The purchase order number, the item number, and the quantity are to be entered. Check to make sure the record does not already exist before it is added to the purchase order. If the record is found, display error message: CANNOT ADD - ITEM ALREADY EXISTS IN PURCHASE ORDER.

NOTES: Normally a maintenance program would also reference an Item Master file for editing functions. To minimize coding, this function has been omitted as well as the requirement to include master file information in a new item record.

Remember to put an A in position 66 of the File Description Specifications to allow you to add records to an existing file. See Position 66 in Chapter 4 of the RPG III Reference Manual and Programmers Guide (SC21-7725) for information on this entry.

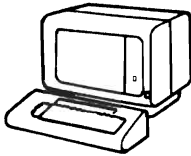
B. To delete an item record:

Only the purchase order number and the item number need be entered. Verify that the record exists before attempting to delete it. If it does not exist, display error message: ITEM# OR P.O.# NOT FOUND.

C. To modify an item record:

The purchase order number, the item number and the quantity are to be entered. To simplify editing, assume that the quantity keyed will not be zero. The new quantity is to replace the quantity ordered (QTYORD) currently in the item record. Verify that the record exists. If the record does not exist, display error message: ITEM# or P.O.# NOT FOUND.

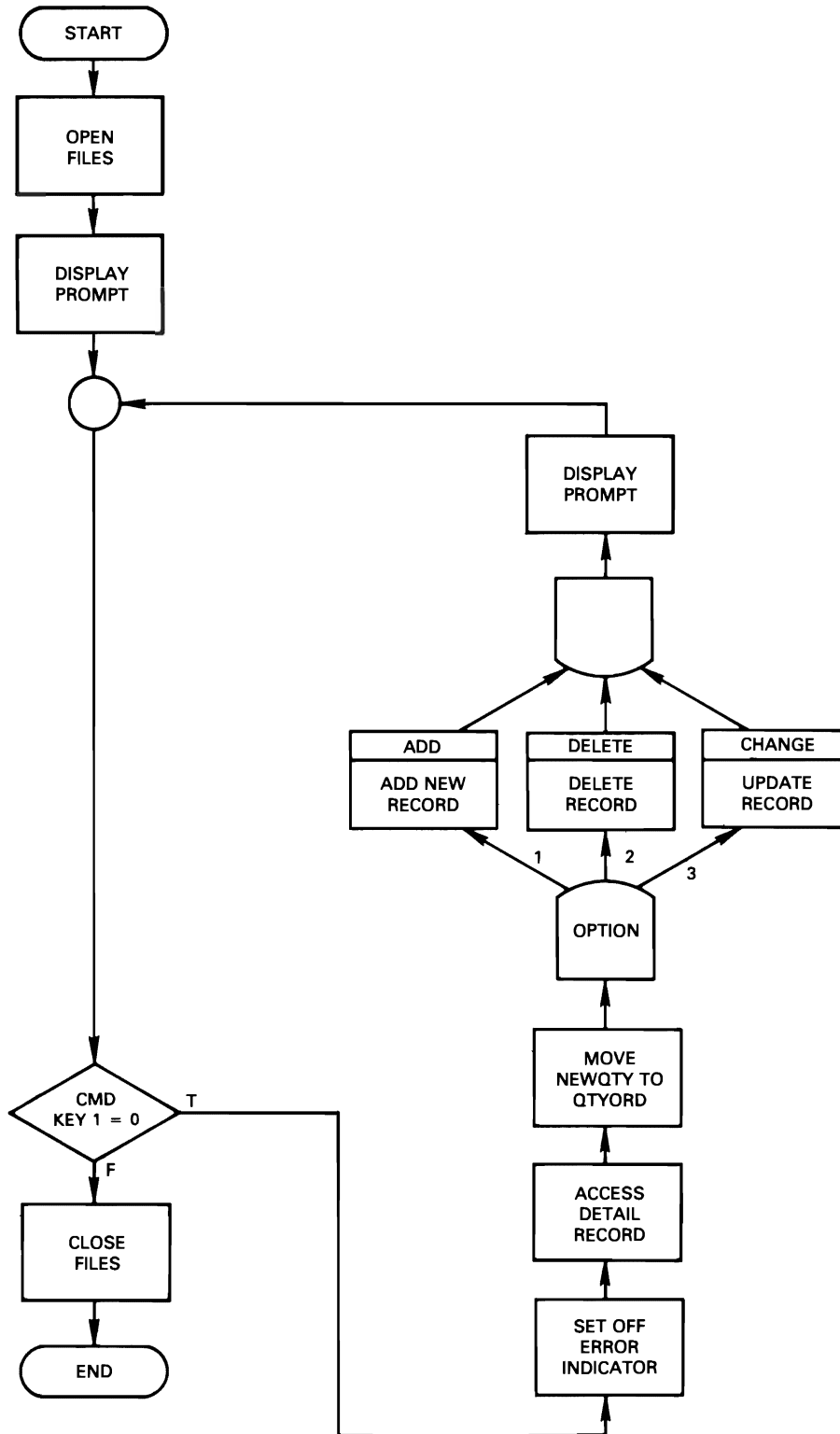
3. Test your logic by checking your program against the structured flowchart.



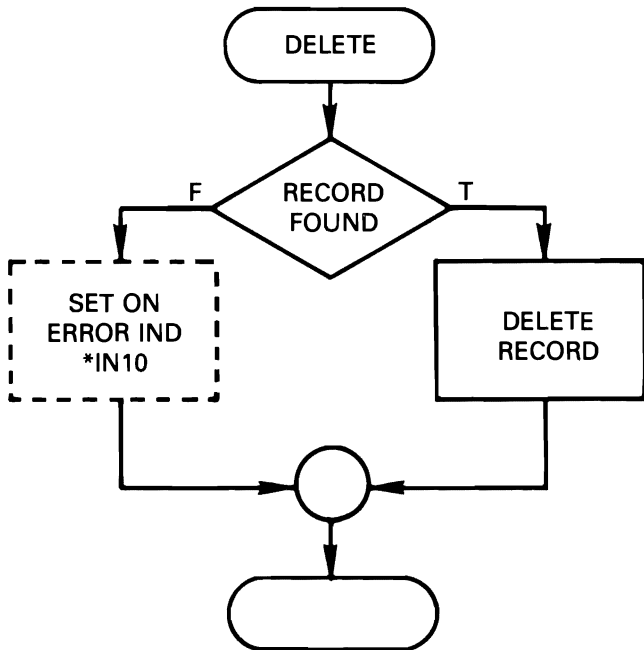
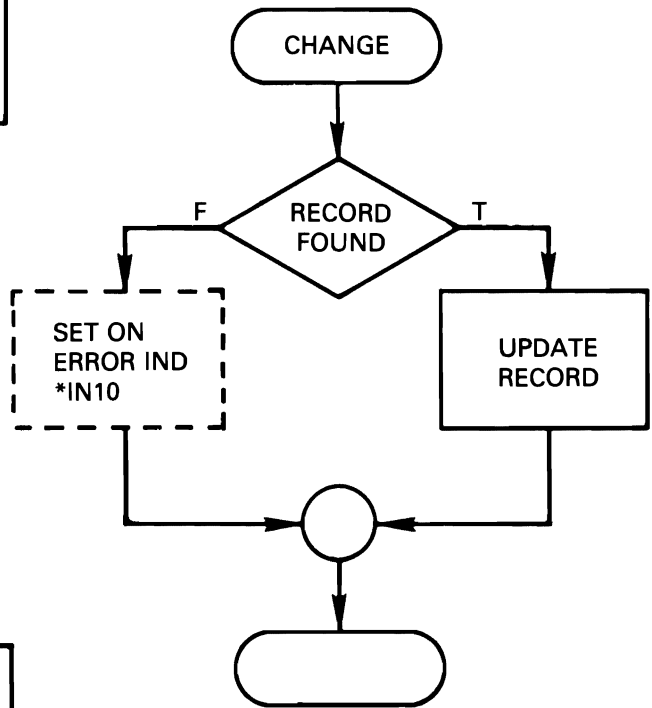
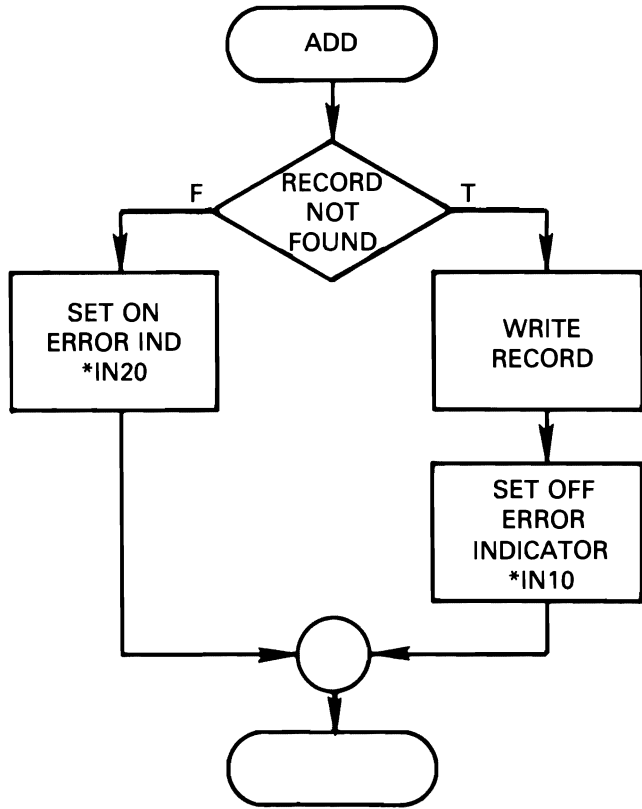
When you are done with the coding, ask your Administrator for Open Purchase Order File Maintenance coding solution in order to check your coding.

When you are ready, do the Open Purchase Order File Maintenance machine exercise that follows.

OPEN PURCHASE ORDER FILE MAINTENANCE FLOWCHART (Part 1)



OPEN PURCHASE ORDER FILE MAINTENANCE FLOWCHART (Part 2)



SCREEN LAYOUT

Display Screen Layout Sheet

	PROMPT SCREEN - PURCHASE ORDER MAINTENANCE COLUMN																																																																																									
	1-10										11-20										21-30										31-40										41-50										51-60										61-70										71-80																			
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
01																																																																																										
02	OPEN PURCHASE ORDER FILE MAINTENANCE																																																																																									
03																																																																																										
04	MAINTENANCE																																																																																									
05	CODE OPTIONS																																																																																									
06																																																																																										
07	1. ADD AN ITEM (ENTER P.O.#, ITEM #, QTY)																																																																																									
08	2. DELETE AN ITEM (ENTER P.O.#, ITEM #)																																																																																									
09	3. MODIFY A QUANTITY ORDERED (ENTER P.O.#, ITEM #, QTY)																																																																																									
10																																																																																										
11																																																																																										
12																																																																																										
13	OPTION P.O.# ITEM # QTY																																																																																									
14	X XXXXXX XXXXX XXXXX																																																																																									
15																																																																																										
16	TO END THE JOB, PRESS CMD KEY 1																																																																																									
17																																																																																										
18	TO ENTER A TRANSACTION, PRESS ENTER																																																																																									
19																																																																																										
20																																																																																										
21																																																																																										
22																																																																																										
23																																																																																										
24	ERROR MESSAGES																																																																																									
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0										

DATA DESCRIPTION SPECIFICATIONS

		MEMBER: PODUPDT															
SEQNBR*	...	1	...	2	...	3	...	4	...	5	...	6	...	7	...	8	
100	A																REF(APPREF)
200	A																CA01(01 *END OF JOB*)
300	A																BLINK
400	A																2 22*OPEN PURCHASE ORDER FILE MAINTENAN+
500	A																CE*
600	A																4 10*MAINTENANCE*
700	A																5 10*CODE OPTIONS*
800	A																7 15*1. ADD AN ITEM (ENTER P.O.#, ITEM +
900	A																#, QTY)*
1000	A																8 15*2. DELETE AN ITEM (ENTER P.O.#, +
1100	A																ITEM #)*
1200	A																9 15*3. MODIFY A QUANTITY ORDERED +
1300	A																(ENTER P.O.#, ITEM #, QTY)*
1400	A																13 13*OPTION P.O.# ITEM # QTY*
1500	A																DSPATR(HI)
1600	A																15VALUES(1 2 3)
1700	A																I 14 2ICHECK(ME)
1800	A	10															ERRMSG(*ITEM # OR P.O.# NOT FOUND*)
1900	A	20															ERRMSG(*CAN NOT ADD - ITEM ALREADY +
2000	A																EXISTS IN PURCHASE ORDER*)
2100	A																I 14 3I
2200	A																5 OI 14 4I
2300	A																17 15*TO END THE JOB, PRESS CMD KEY 1.*
2400	A																19 15*TO ENTER A TRANSACTION, PRESS +
2500	A																ENTER*

Machine Exercise - Open Purchase Order File Maintenance

Sign on by entering the password GLCRPG.

This exercise provides practice in entering, compiling and running the Open Purchase Order File Maintenance program you just coded in this module.

Please read all instructions before doing them. In following the instructions you may find it helpful to refer to the more detailed instructions of the machine exercise of Module 4.

1. Select option 8, enter the name PODUPDxx and the type DSPF for your display file.
2. Copy all of the data description specification statements from the member PODUPDT in the RPGSRC file into your member. These are the only statements you need to create your display file.
3. Select option 3 to create the display file PODUPDxx in the GLCRPG library.
4. You will use the same data base files (POLITMxx, POPSUMxx, POPDTLxx) for this exercise that were used in Machine Exercise 1.

Restore these files to their original state by selecting option 5 and entering the command:

```
CALL RESTR1 PARM(xx)
```

5. Create a source member in the RPGSRC file by the name POR2xx, and enter your program.
6. Create your object program in the GLCRPG library using the name POR2xx. Before proceeding be sure your program compiled successfully.
7. Before executing your program, see the "BEFORE" listing.

BEFORE EXECUTING POR2xx

EXERCISE 2 TEAM#									
PJ NUMBER	VENDOR NUMBER	VENDOR NAME	ORDER AMOUNT	STATUS	ITEM NUMBER	ITEM DESCRIPTION	QUANTITY ORDERED	QUANTITY RECEIVED	DATE RECEIVED
400001	10	JOHN M. SMITH & SONS	500.00	C	25	LINED PAPER	25		C/C0/00
					150	BEAKERS	100		C/C0/00
					375	ECGERS	30	30	3/15/80
					475	RIBBONS	100	100	2/26/80
400005	11	FETZNER & FETZNER	110.00	C	50	SPECLS	10	5	2/02/80
					100	INVITATIONS	15	15	2/12/80
					200	LADDERS	6	6	2/24/80
					400	FILES	8	8	2/16/80
400010	16	BRAND X BEER	350.00	C	75	BOTTLES	500		C/C0/00
					450	CANS	25		C/C0/00
400015	431	MARGARET HART INC.	121.00	C	175	CARPETS	3	3	2/22/80
					250	SPIRDLERS	5	5	2/17/80
					275	MOLDS	2		C/C0/00
400020	5075	PETER VAN ROTH CORP.	159.50	C	300	DISPLAYS	6	6	1/23/80
					425	CABINETS	9		C/C0/00
400025	7374	PHOTO LABS INC.	45.50	C	500	PICTURES	13	13	2/05/80

8. Execute your program by selecting option 4 on the Programmer Menu. Enter the following maintenance transactions:

(NOTE: When testing error conditions do not attempt to correct the data by entering it as a transaction.)

Maintenance Code	Purchase Order No.	Item Number	Quantity
1	400001	100	50
2	400001	25	
3	400001	150	150
1	400001	375	10
		ERROR-ALREADY EXISTS	
2	400001	99999	
		ERROR-NOT FOUND	
3	400001	99999	10
		ERROR-NOT FOUND	

9. Enter the command:

CALL LIST2 PARM(xx)

and compare your listing to the "AFTER" listing. Your listing should reflect the maintenance transactions entered.

AFTER EXECUTING POR2xx

EXERCISE 2 TEAMWW											
PO NUMBER	VENDOR NUMBER	VENDOR NAME	ORDER AMOUNT	STATUS	ITEM NUMBER	ITEM DESCRIPTION	QUANTITY ORDERED	QUANTITY RECEIVED	DATE RECEIVED		
400001	10	JOHN M. SMITH & SONS	500.00		25	Deleted					
					100	New Record	50		C/C0/CC		
					150	BEAKERS Change	150		C/C0/CC		
				C	375	EDGERS	30	30	3/15/80		
				C	475	RIBBONS	100	100	2/26/80		
400005	11	FETZNER EFETZNER	110.00		50	SPCCLS	10	5	2/22/80		
				C	100	INVITATIONS	15	15	2/12/80		
				C	200	LADDERS	6	6	2/24/80		
				C	400	FILES	8	8	2/16/80		
400010	16	BRAND X BEER	350.00		75	BCTTLES	500		C/C0/CC		
					450	CANS	25		C/C0/CC		
400015	431	MARGARET HART INC.	121.00		175	CARPETS	3	3	2/22/80		
				C	250	SPINDLES	5	5	2/17/80		
				C	275	MCLDS	2		C/C0/CC		
400020	5075	PETER VAN ROTH CORP.	169.50		300	DISPLAYS	6	6	1/23/80		
				C	425	CABINETS	9		C/C0/CC		
400025	7374	PHOTO LABS INC.	45.50		500	PICTURES	13	13	2/05/80		

10. If you need fresh data in your files, enter the command:

CALL RESTR1 PARM(xx)



When you have completed the Open Purchase Order File Maintenance exercise, return to Module Text 5 to continue the course.

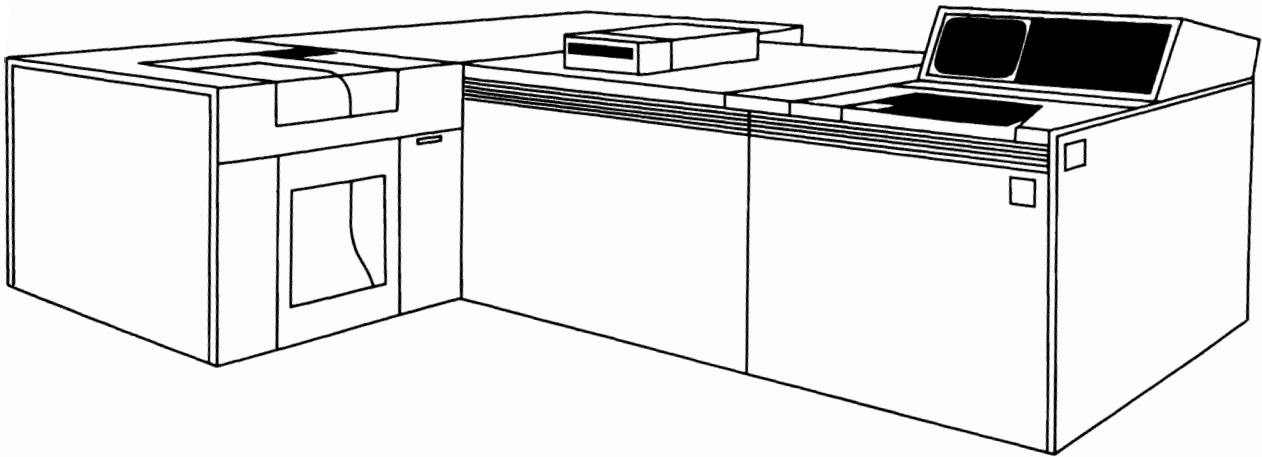


MODULE SUMMARY

You have discovered that computer programs can be written with a high degree of structure, which permits them to be more easily understood for testing, maintenance, and modification. Also with structured programming, you have found control branching is entirely standardized so that code can be read from top to bottom, without having to trace the branching logic as is typical without structured code.

You may have the requirement to write a program to perform a function useful to many programs in an application. This type of function is a candidate for a subprogram. Module 6 discusses how subprograms are coded and executed on the System/38.





Chapter 6. Module 6 - Subprograms

MODULE PURPOSE

Module 6 discusses subprograms and coding for sending and receiving parameters in RPG III programs.

TIME ESTIMATE

2 hours

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- Code an RPG III program using instructions to pass parameters to a subprogram.
- Code an RPG III subprogram using instructions to receive parameters from a mainline program.

TERMS

These terms are described/defined within the module.

- PLIST
- PARM
- CALL
- RETRN
- FREE

Coding Exercise - Subprograms

The purpose of this exercise is for you to use the operation codes to execute subroutines and transfer control to another program. You are to use the CASxx, CALL, PARM and PLIST operations and structured programming in your program logic.

This exercise is an extension of the previous machine exercise where you maintained the Open Purchase Order file. Now you are required to delete an entire purchase order from the file. This requires modifying your file maintenance program POR2xx to call a new program to be named PORSxx, which will delete an entire purchase order.

You will find documentation (flowchart, screen layout, display DDS coding) on the following pages for use in coding the exercise.

1. Using structured programming, modify your program logic for POR2xx by adding the option to delete a purchase order. The other maintenance functions remain the same as in the file maintenance exercise. Name the modified program POR3xx.
2. Develop your program logic for PORSxx using the supplied structured programming flowchart.

The function of this new program is to delete both the summary and item records of a purchase order. If the purchase order number entered for deletion does not exist, this program is to pass an error condition back to the main program.

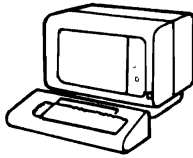
The display layout is included on the following pages. The new delete function added to the display is shown boxed on the display layout.

The new delete function has been added to the display file source specifications for you. The complete display file specifications are located in the source member PODDELT and are included in the documentation.

3. Code a new program to delete an entire purchase order and name it PORSxx. This program receives the purchase order number (PORNBR) from POR3xx. If PORNBR is invalid, the program is to return to POR3xx with an error indicator on. If PORNBR is valid, the program deletes the entire purchase order and then returns to POR3xx.

The program PORSxx is to access the Purchase Order logical file POLONOxx which is built over two physical files POPSUM and POPDTL. The logical file has an access path keyed on purchase order number. The data description specifications are included in the documentation.

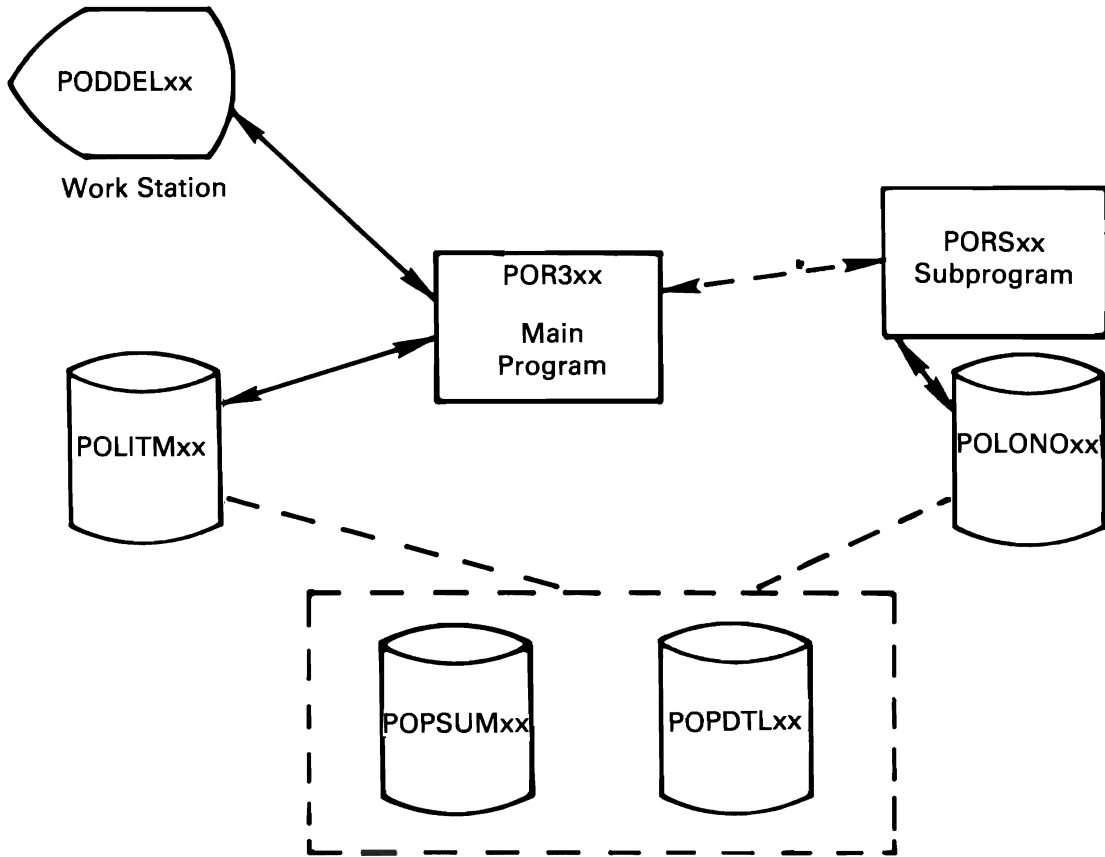
4. Test your logic by checking the flowchart with your program.



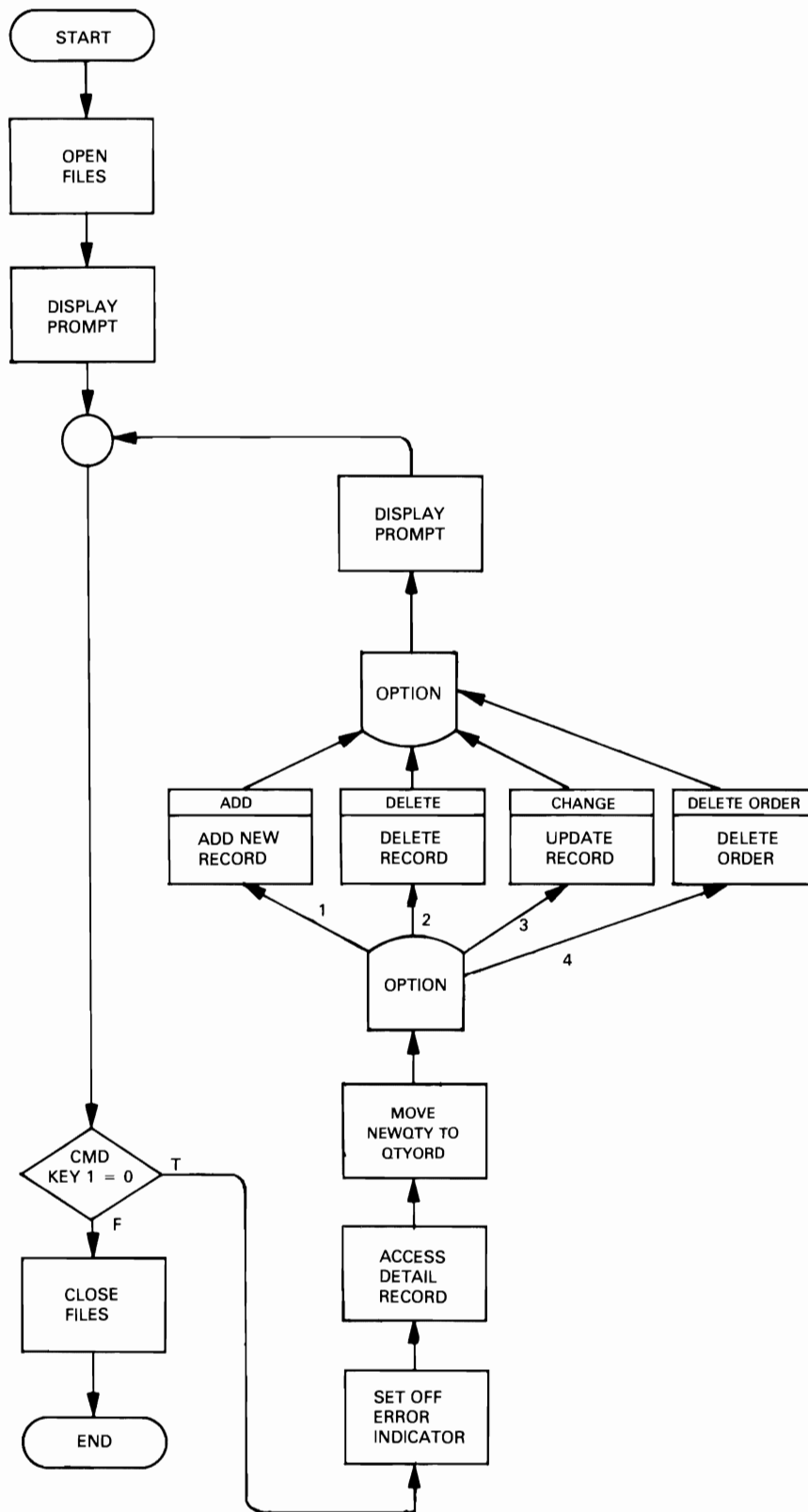
When you are done with the coding, ask your Administrator for the coding solutions.

When you are ready, do the Subprogram machine exercise that follows.

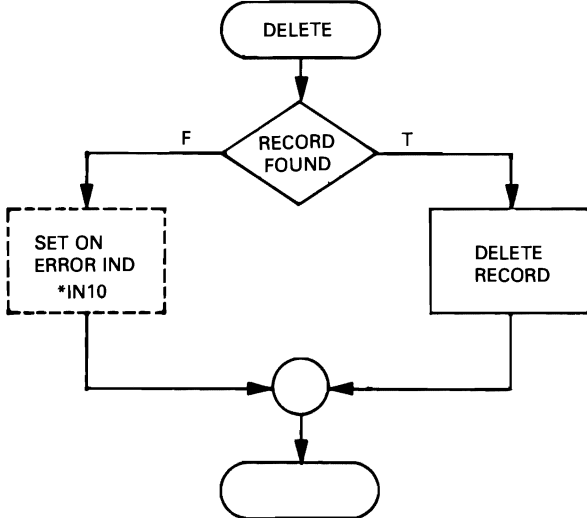
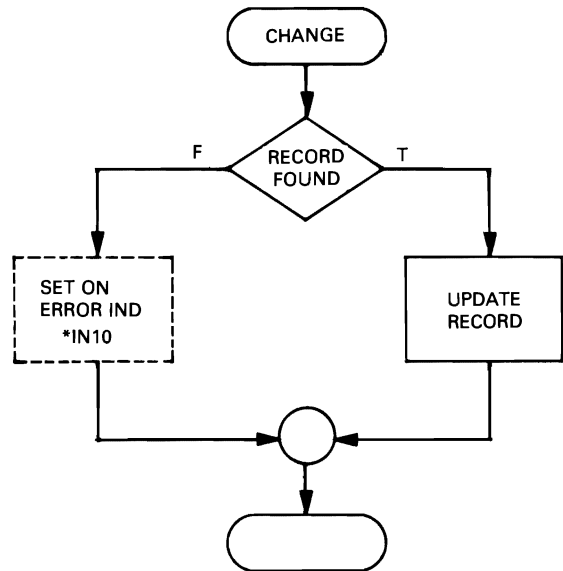
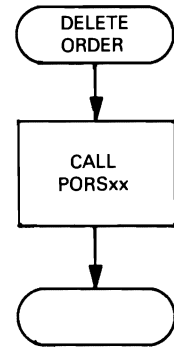
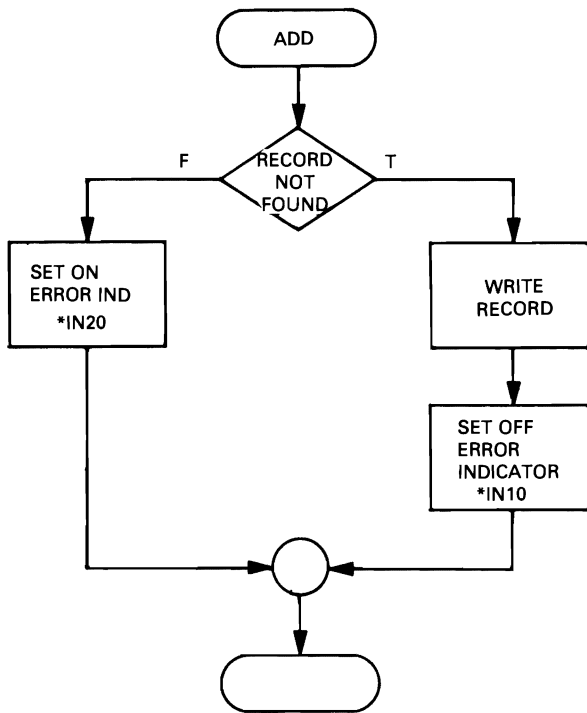
**SUBPROGRAM EXERCISE
SYSTEM FLOWCHART**



**CALLING PROGRAM – POR3xx
FLOWCHART (Part 1)**



CALLING PROGRAM - POR3xx
FLOWCHART (Part 2)



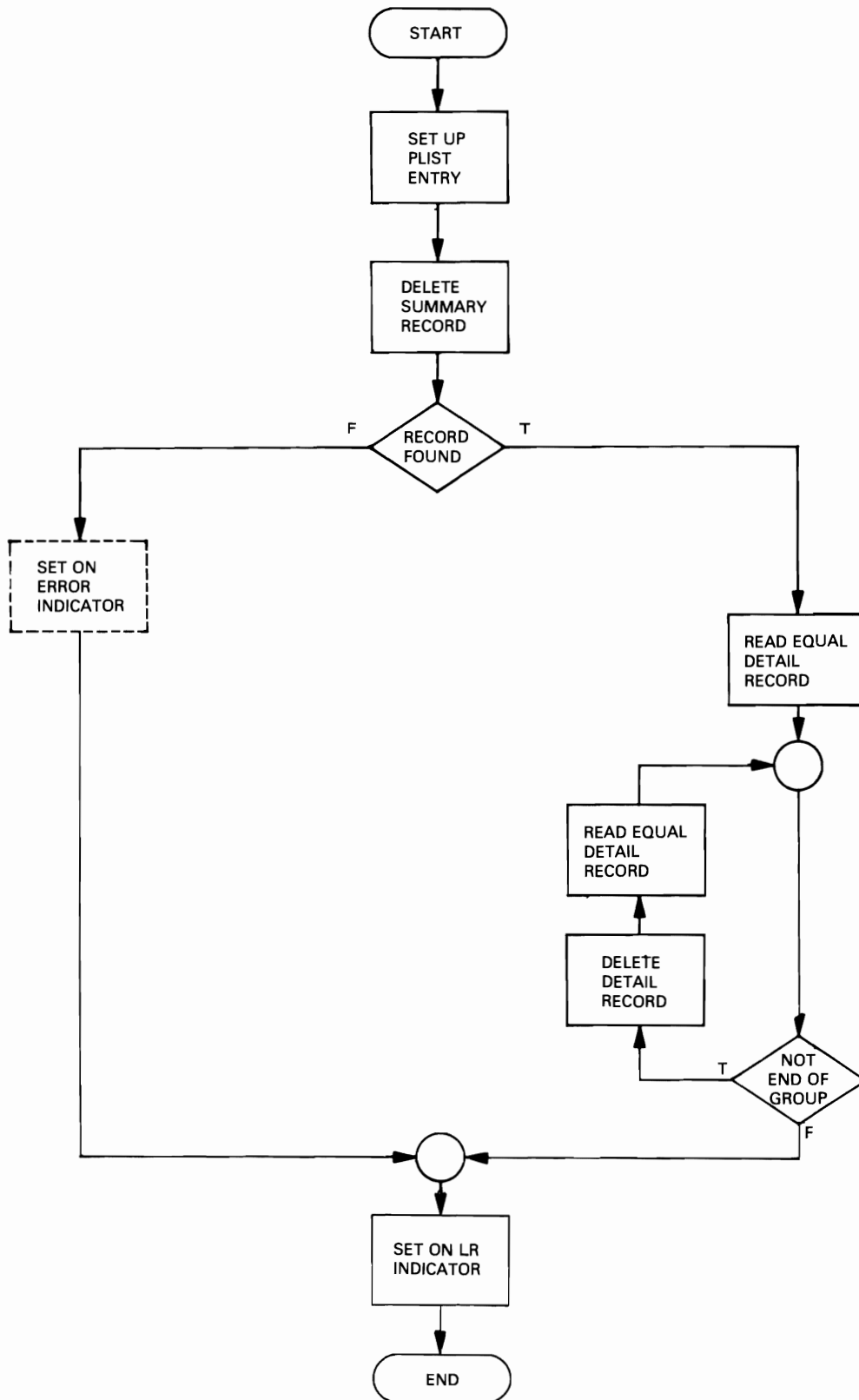
SCREEN LAYOUT

PROMPT SCREEN - PURCHASE ORDER MAINTENANCE										COLUMN																																																											
1-10		11-20		21-30		31-40		41-50		51-60		61-70		71-80																																																							
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
01																																																																					
02	OPEN PURCHASE ORDER FILE MAINTENANCE																																																																				
03																																																																					
04	MAINTENANCE																																																																				
05	CODE OPTIONS																																																																				
06																																																																					
07	1. ADD AN ITEM (ENTER P.O.#, ITEM #, QTY)																																																																				
08	2. DELETE AN ITEM (ENTER P.O.#, ITEM #)																																																																				
09	3. MODIFY A QUANTITY ORDERED (ENTER P.O.#, ITEM #, QTY)																																																																				
10	4. DELETE AN ENTIRE PURCHASE ORDER (ENTER P.O.#)																																																																				
11																																																																					
12																																																																					
13	OPTION P.O.# ITEM # QTY																																																																				
14	X XXXXX XXXX XXXX																																																																				
15																																																																					
16	TO END THE JOB, PRESS CMD KEY 1																																																																				
17																																																																					
18	TO ENTER A TRANSACTION, PRESS ENTER																																																																				
19																																																																					
20																																																																					
21																																																																					
22																																																																					
23																																																																					
24	ERROR MESSAGES																																																																				
1-10		11-20		21-30		31-40		41-50		51-60		61-70		71-80																																																							
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0

DATA DESCRIPTION SPECIFICATIONS

SEQNBR#	1	2	3	4	5	6	7	8
	MEMBER: PODDEL T							
100	A							REF(APPREF)
200	A							CA01(01 'END OF JOB')
300	A	R PROMPT						BLINK
400	A							2 22 'OPEN PURCHASE ORDER FILE MAINTENAN+
500	A							CE'
600	A							4 10 'MAINTENANCE'
700	A							5 10 'CODE OPTIONS'
800	A							7 15 '1. ADD AN ITEM (ENTER P.O.#, ITEM +
900	A							#, QTY)'
1000	A							8 15 '2. DELETE AN ITEM (ENTER P.O.#, +
1100	A							ITEM #)'
1200	A							9 15 '3. MODIFY A QUANTITY ORDERED +
1300	A							(ENTER P.O.#, ITEM #, QTY)'
1400	A							10 15 '4. DELETE AN ENTIRE PURCHASE +
1500	A							ORDER (ENTER P.O.#)'
1600	A							13 13 'OPTION P.O.# ITEM # QTY'
1700	A							DSPATR(HI)
1800	A	OPTION		1	OI	14	15	VALUES(1 2 3 4)
1900	A	PORNBR	R		I	14	21	CHECK(ME)
2000	A	10						ERRMSG('ITEM # OR P.O.# NOT FOUND')
2100	A	20						ERRMSG('CAN NOT ADD - ITEM ALREADY +
2200	A							EXISTS IN PURCHASE ORDER')
2300	A	ITMNR	R		I	14	31	
2400	A	NEWQTY		5	OI	14	41	
2500	A							17 15 'TO END THE JOB, PRESS CMD KEY 1.'
2600	A							19 15 'TO ENTER A TRANSACTION, PRESS +
2700	A							ENTER'

CALLED PROGRAM – PORSxx FLOWCHART



DATA BASE FILES FOR SUBPROGRAM PORSxx

SOURCE FILE: RPGSRC.GLCRPG MEMBER: POLONO

SEQNBR#..... 1 2 3 4 5 6 7

100	A	R	ORDSUM		PFIL(POPSUM)
200	A	K	PORNBR		
300	A	R	ORDDTL		PFIL(POPDTL)
400	A	K	PORNBR		

SOURCE FILE: RPGSRC.GLCRPG MEMBER: POPSUM

SEQNBR#..... 1 2 3 4 5 6 7

100	A				REF(APPREF)
200	A	R	ORDSUM		TEXT(*PO SUMMARY RECORD*)
300	A		PORNBR	R	
400	A		VNCNBR	R	
500	A		VNDNAM	R	
600	A		MERCH	R	
700	A		DATORD	R	
800	A		VNDSLS	R	
900	A		VNDACD	R	
1000	A		VNDPHN	R	
1100	A		STATUS	R	

SOURCE FILE: RPGSRC.GLCRPG MEMBER: POPDTL

SEQNBR#..... 1 2 3 4 5 6 7

100	A				REF(APPREF)
200	A	R	ORDDTL		TEXT(*PO LINE ITEM RECORD*)
300	A		PORNBR	R	
400	A		VNDNBR	R	
500	A		ITMNB	R	
600	A		ITMDCS	R	
700	A		QTYCRD	R	
800	A		ITMCST	R	
900	A		ITMAMT	R	
1000	A		DEPTNC	R	
1100	A		CATNBR	R	
1200	A		DATREC	R	
1300	A		QTYREC	R	
1400	A		STATUS	R	

Machine Exercise - Subprograms

Sign on by entering the password GLCRPG.

This exercise provides practice in entering, compiling and testing the Open Purchase Order File Maintenance program which you previously coded in this module.

Please read all instructions before you start to do them. To avoid major problems in doing this exercise be sure you perform all steps in sequence. The EXERCISE in module 5 MUST be completed successfully.

- / 1. Use option 8 to create a source member for your display file. Name the source member PODDELxx. (Substitute your first and last initials for xx.)
- / 2. Using Browse/Copy on the Services display, copy all of the data description specifications from PODDELT into your source member.
- ✓ 3. Select option 3 and create your new display file PODDELxx in the GLCRPG library.

4. You will use the same set of data base files (POLITMxx, POPSUMxx, POPDTLxx) as you did in the previous machine exercises.
5. To obtain a fresh set of files, enter the command:

```
CALL LAB3 PARM(xx)
```
6. Create a source member in the RPGSRC file by the name of POR3xx, and using Browse/Copy in the Services display, copy your source code from POR2xx into this new member.
7. Add the code for the new delete function to POR3xx and change the file name for the work station from PODUPDxx to PODDELxx. When option 4 is selected on the display, POR3xx is to call your subprogram PORSxx and specify the purchase order number (PORNBR) and an indicator (*INxx) as parameters. The indicator is to be set to true ('1') when the purchase order number is invalid.
8. Create your object program POR3xx in the GLCRPG library.
9. Create a source member for a logical file in the RPGSRC file and call it POLONOxx.
10. Copy all the data description statements from the member POLONO into POLONOxx. Modify the PFILE parameters by changing POPSUM and POPDTL to your file names POPSUMxx and POPDTLxx.
11. Create your logical file POLONOxx in the GLCRPG library.
12. Create a source member in the RPGSRC file by the name of PORSxx and enter your program.
13. Create your object program PORSxx in the GLCRPG library.

14. Before executing your program, see the BEFORE listing:

BEFORE EXECUTING POR3xx

EXERCISE 3 TEAMWW									
PJ NUMBER	VENDOR NUMBER	VENDOR NAME	ORDER AMOUNT	STATUS	ITEM NUMBER	ITEM DESCRIPTION	QUANTITY ORDERED	QUANTITY RECEIVED	DATE RECEIVED
400001	10	JOHN M. SMITH & SONS	500.00		25	LINED PAPER	25		0/00/00
					150	BEAKERS	100		0/00/00
				C	375	EDGERS	30	30	3/15/80
				C	475	RIBBONS	100	100	2/26/80
400005	11	FETZNER & FETZNER	110.00		50	SPOOLS	10	5	2/02/80
				C	100	INVITATIONS	15	15	2/12/80
				C	200	LADDERS	6	6	2/24/80
				C	400	FILES	8	8	2/16/80
400010	16	BRAND X BEER	350.00		75	BOTTLES	500		0/00/00
					450	CANS	25		0/00/00
400015	431	MARGARET HART INC.	121.00		175	CARPETS	3	3	2/22/80
				C	250	SPINDLES	5	5	2/17/80
					275	MCLDS	2		0/00/00
400020	5075	PETER VAN ROTH CCRP.	169.50		300	DISPLAYS	6	6	1/23/80
				C	425	CABINETS	9		0/00/00
400025	7374	PHOTO LABS INC.	45.50		500	PICTURES	13	13	2/05/80

15. Execute your programs by calling POR3xx and entering these transactions:

<u>Maintenance Code</u>	<u>Purchase Order No.</u>	<u>Item Number</u>	<u>Quantity</u>
1	400001	100	50
2	400001	25	
3	400001	150	150
4	400015		
4	400016	ERROR-NOT FOUND	
4	400010		

16. If you have problems and need a fresh set of data in your files, enter the command:

CALL RESTR3 PARM(xx)

17. When you are finished entering the transactions, enter the command:

CALL LIST3 PARM(xx)

and compare your listing to the following listing. Your listing should reflect the maintenance transactions entered.

AFTER EXECUTING POR3xx

EXERCISE 3 TEAMWW									
PO NUMBER	VENDGR NUMBER	VENDOR NAME	ORDER AMOUNT	STATUS	ITEM NUMBER	ITEM DESCRIPTION	QUANTITY ORDERED	QUANTITY RECEIVED	DATE RECEIVED
400001	10	JOHN M. SMITH & SONS	500.00		100		50		C/C0/00
					150	BEAKERS	150		C/C0/00
				C	375	EDGERS	30	30	3/15/80
				C	475	RIBBONS	100	100	2/26/80
400005	11	FETZNER &FETZNER	110.00		50	SPOOLS	10	5	2/C2/80
				C	100	INVITATIONS	15	15	2/12/80
				C	200	LADDERS	6	6	2/24/80
				C	400	FILES	3	8	2/16/80
400020	5075	PETER VAN ROTH CORP.	169.50	C	300	DISPLAYS	6	6	1/23/80
					425	CABINETS	9		C/C0/00
400025	7374	PHOTO LABS INC.	45.50	C	500	PICTURES	13	13	2/C5/80



When you have completed the Subprogram machine exercise, return to Module Text 6 to continue the course.

NOTES

When is a program a candidate as an external subroutine (subprogram)?

ans. When it is useful to many programs in the application.

The two RPG III operation codes CALL & RETRN link programs together.

CALL operation passes control to the program you have specified in factor 2. The name of the called program may be specified as

1. literal CALL 'APL25' or
2. ^{variable name} field name CALL APL20

* For performance you should use literals.

If you use a literal to specify your program name without a library name, RPG III searches the library list on only the first execution. Subsequent executions within your program of that CALL operation invokes the same program again without searching your library list.

If you qualify your program name, the program name must be immediately followed by a period and then the library name. The literal cannot exceed eight characters including the period 'APL2.LIB'

A field including the period cannot exceed 21 characters.

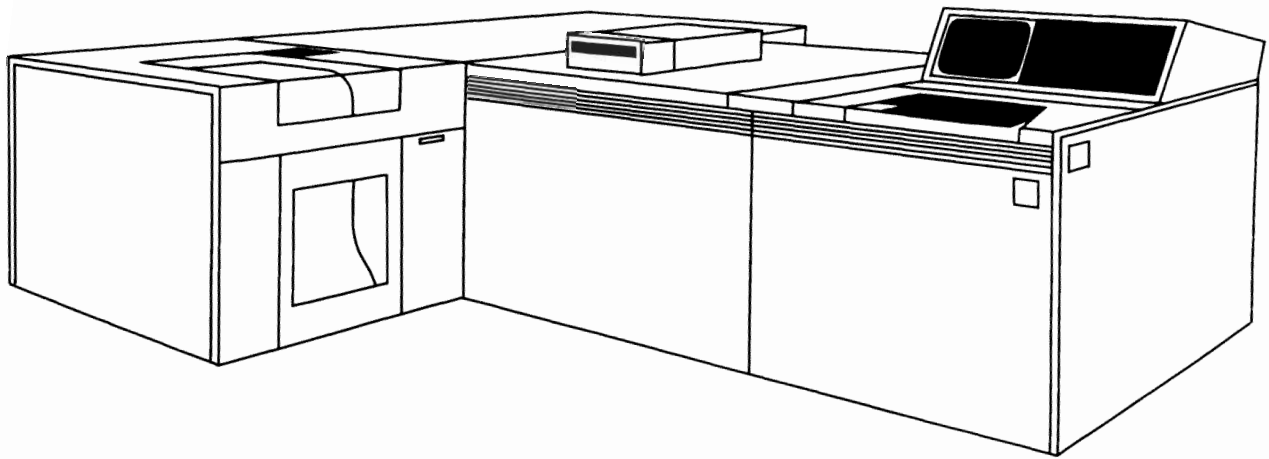


MODULE SUMMARY

Subprograms are programs that are invoked by other programs. In the calling program, there are several ways to name the subprogram or pass the parameters. The most straight forward way is to CALL, naming the subprogram as a literal and following it by PARM statements.

Module 7 discusses subfiles and subfile coding. You have studied in other courses the data description specifications for subfiles. Module 7 will teach you the tasks involved in controlling subfiles and the operation codes available to implement those tasks.





Chapter 7. Module 7 - Subfile Programming

MODULE PURPOSE

Module 7 discusses managing subfile operations in RPG III for inquiry from and update to a data base file.

TIME ESTIMATE

4 hours

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- Code an RPG III program to create and display a subfile.
- Code an RPG III program to update a data base file using a subfile.

TERMS

These terms are described/defined within the module.

- Selective Subfile DDS keywords
- Pseudo Code
- SETxx
- READC



VIDEO INTRODUCTION


You have studied in the Application Programming course that a subfile is a group of records that have the same record format and are read from and written to a display device in one operation. You also found that a subfile is a special kind of display file. Subfiles are useful when multiple records that are alike must be displayed.

The video you are about to see illustrates the RPG III operations supporting subfiles for:

- Display, which will allow you to review the subfile records.
- Update, which will allow you to modify one or more records in the subfile for updating your data base.

Video Summary

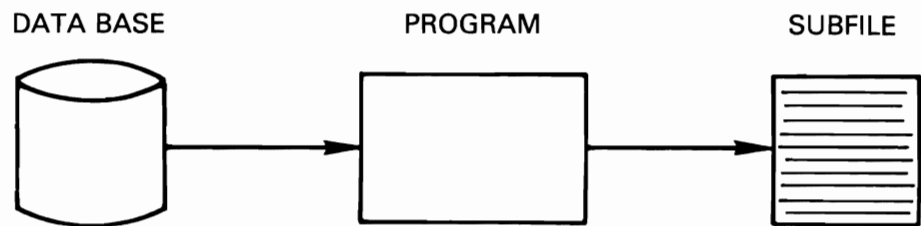
You have been given a brief overview of how subfile operations are managed through RPG III. The following operations were illustrated:

- Displaying the subfile control record.
 - Filling the subfile.
 - Displaying the subfile records.
 - Reading changed subfile records for update.
- 

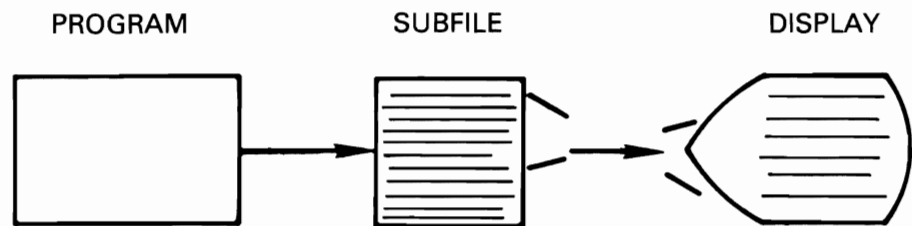
SUBFILE REVIEW

You saw in the video on subfiles that there are a few basic operations in using subfiles.

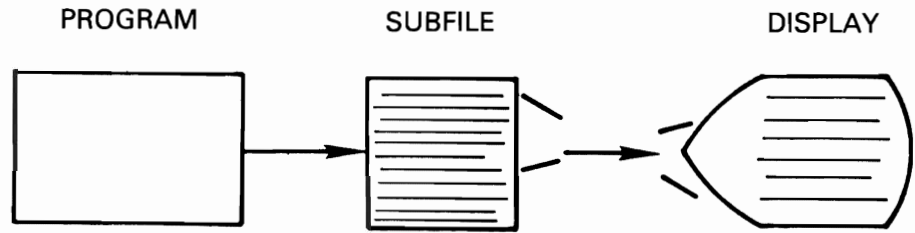
- 1** You read records from a data base file and write them to a subfile. The records are placed in the subfile one at a time, until the subfile is full or until there are no more records to read.



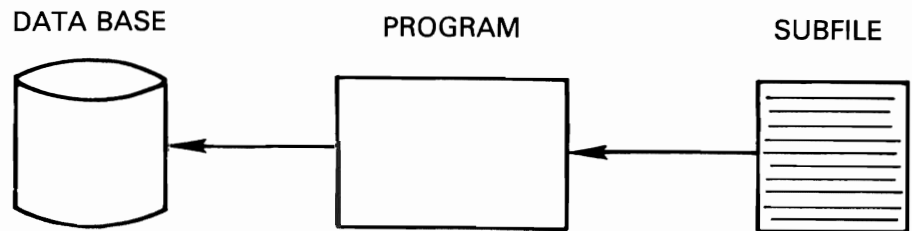
- 2** Now you send the subfile to the display station with one operation to the subfile.



3 Your work station operator at this time may review the records and/or modify the records.



4 You read the subfile and process each modified record, updating the data base file as required.



Pseudo Code

The following illustration represents our Vendor Master File Inquiry in pseudo code.

Pseudo code is written from top to bottom.

Subordinate functions are indented. You will find that indentation makes it easy to read.

```
OPEN FILES (IMPLICIT BY RPG)
DISPLAY SEARCH PROMPT
DOWHILE EOJ NOT INDICATED
| CLEAR SUBFILE
|
| POSITION THE FILE POINTER
|
| INITIALIZE RRN
| SET SUBFILE FILL SWITCH OFF
| DOWHILE SUBFILE FILL SWITCH IS OFF
|
|   READ VENDOR RECORD
|   IF EOF NOT SENSED
|   | INCREMENT RRN BY ONE
|   | WRITE VENDOR RECORD TO SUBFILE
|   | IF SUBFILE FULL
|   |   SET SUBFILE FILL SWITCH ON
|   | (ELSE)
|   |   ENDIF
|   ELSE
|   | SET SUBFILE FILL SWITCH ON
|   | ENDIF
|   ENDDO
|   DISPLAY THE SUBFILE
| ENDDO
CLOSE FILES
```

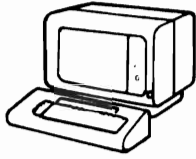

Coding Exercise - Subfiles-Inquiry

The purpose of this exercise is for you to write an RPG III program that displays multiple records from a data base file. In this exercise you are to write a program to display a vendor invoice.

You will find documentation (HIPO diagrams, flowchart, DDS coding for the display, logical, and physical files, display layout) on the following pages.

1. Review the HIPO diagram that follows.
2. The program logic has been structurally flowcharted for you from the HIPO diagram.
3. Your program APR4xx will access your Vendor Invoice logical file APLDUExx which is built over your physical file APPOPYxx. The logical file is keyed on invoice number within due date. See the data description specifications for these files.
4. The subfile has been partially coded for you. In reviewing these specifications, you will find the subfile control record is provided except for the command keys and the SFL keywords. You need to add this code as well as the code for the subfile record. Code a value of 20 for subfile size and a value of 8 for subfile page. You are to name your subfile APDSLcxx.
5. Using the supplied structured flowchart, code your program.

6. Test your logic by checking the flowchart with your program.



When you are done with the coding, ask Administrator for the Subfile-Inquiry coding solution.

When you are ready, do the Subfile-Inquiry machine exercise which follows.

HIPO DIAGRAMS

DIAGRAM ID: 3.2.1

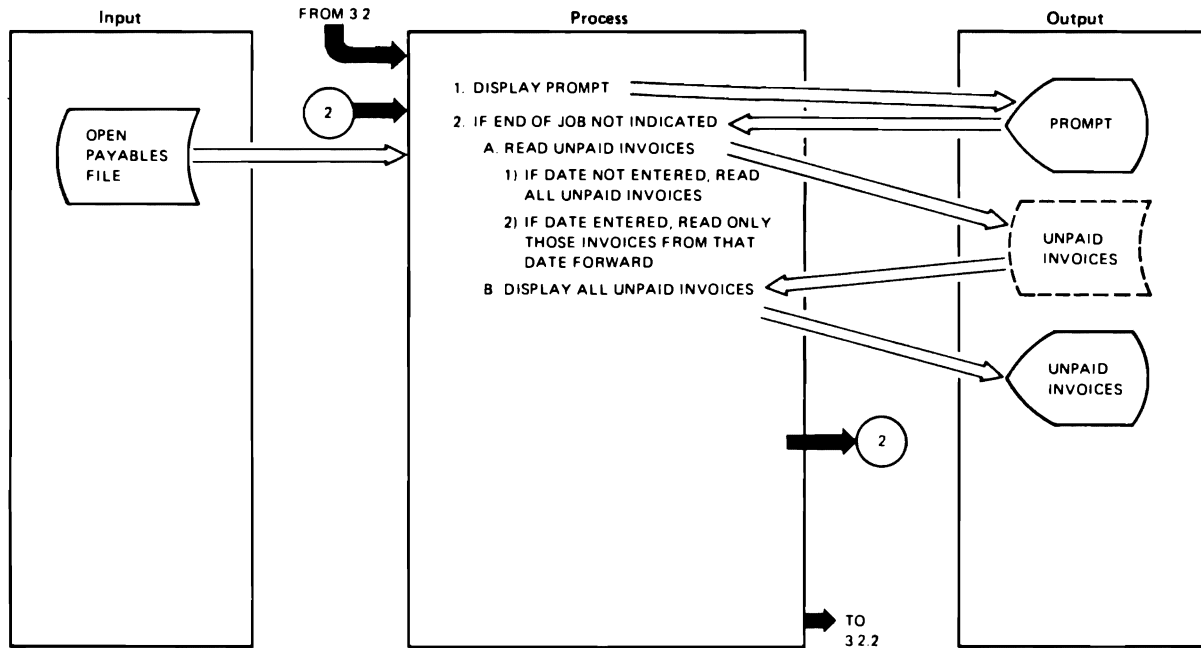
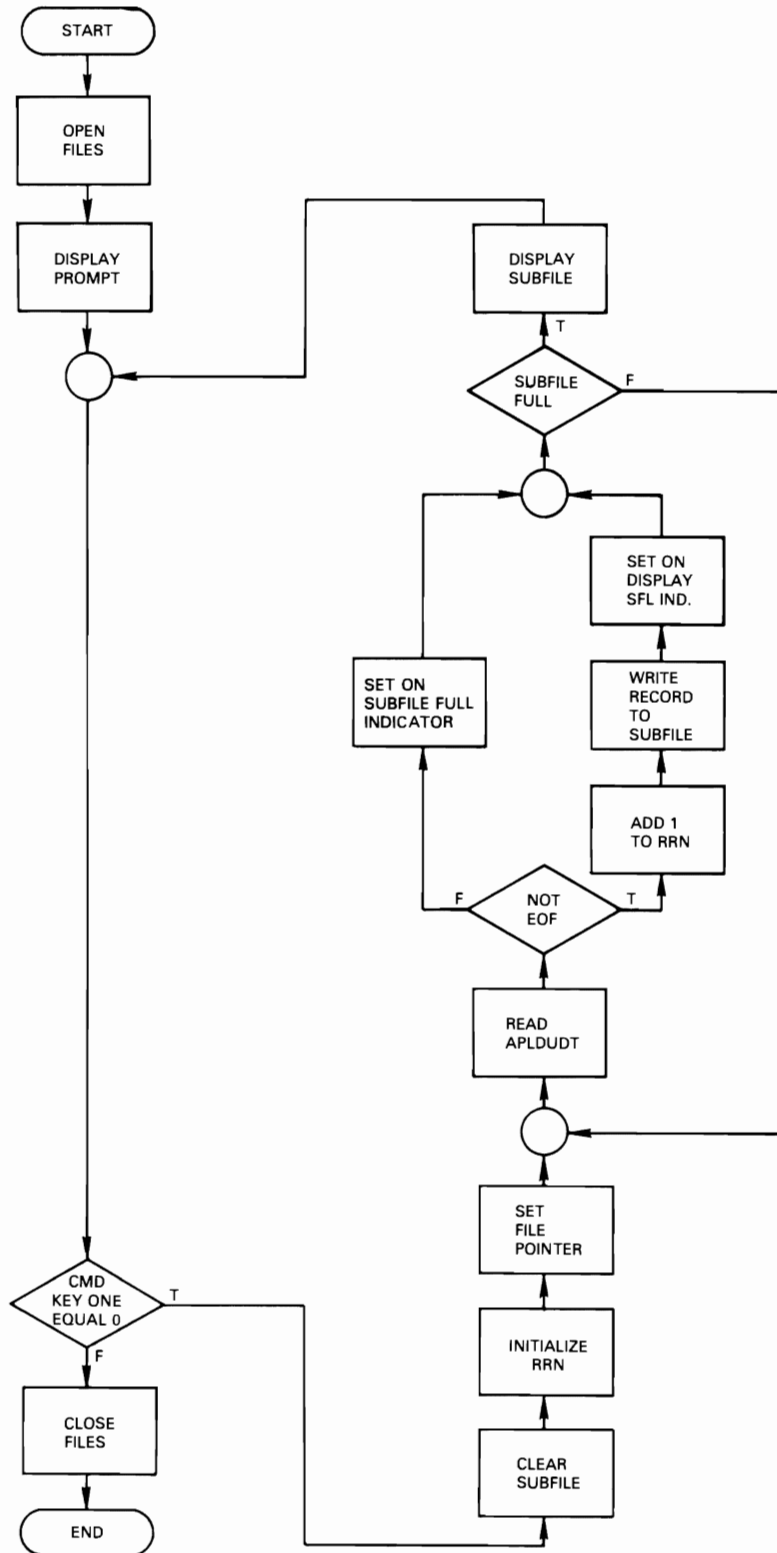


DIAGRAM 3.2.1

Extended Description

Notes		Ref.
Display the subfile control record. The operator will press enter to display all invoices or enter a date and then press enter to display the invoices due from that date forward.		1
The program should allow the operator to end the job anytime command key one is pressed.		2
Build a subfile for all the records in the Open Payables File.		2.A
If a date was entered, a search key must be built and the file pointer set.		2.A.2

VENDOR INVOICE DISPLAY PROGRAM – APR4xx FLOWCHART



**DATA BASE FILES FOR VENDOR INVOICE DISPLAY PROGRAM
APR4xx**

MEMBER: APLDUE

SEQNBR#	1	2	3	4	5	6	7	8
100	A		R PAYBLE					PFILF(APPOPY)
200	A		PORNBR					
300	A		VNDNBR					
400	A		INVNBR					
500	A		DATREC					
600	A		MERCH					
700	A		DCTAVL					
800	A		NET					
900	A		STATUS					
1000	A		DUDATE					CONCAT(DUMODY DUYR)
1100	A		DUMODY					
1200	A		DUYR					
1300	A		K DUYR					
1400	A		K DUMODY					
1500	A		K INVNBR					

MEMBER: APPOPY

SEQNBR#	1	2	3	4	5	6	7	8
100	A							REF(APPFREF)
200	A		R PAYBLE					TEXT(*OPEN PAYABLES RECORD*)
300	A		PORNBR	R				
400	A		VNDNBR	R				
500	A		INVNBR	R				
600	A		DATREC	R				
700	A		MERCH	R				
800	A		DCTAVL	R				
900	A		NET	R				
1000	A		STATUS	R				
1100	A		DTPAID	R				
1200	A		CHECK#	R				
1300	A		DUMODY	R				
1400	A		DUYR	R				

SCREEN LAYOUT

COLUMN

	1-10										11-20										21-30										31-40										41-50										51-60										61-70										71-80									
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
01																																																																																
02	VENDOR INVOICE SELECTION																																																																															
03																																																																																
04	TO DISPLAY ALL INVOICES, PRESS ENTER.																																																																															
05																																																																																
06	TO DISPLAY INVOICES DUE BEYOND A SPECIFIC DATE, KEY THE YEAR LL AND																																																																															
07	MONTH/DAY LLLL, THEN PRESS ENTER.																																																																															
08																																																																																
09	TO END JOB, PRESS CMD KEY 1.																																																																															
10																																																																																
11																																																																																
12																																																																																
13	VENDOR INVOICE DUE GROSS DISCOUNT NET STATUS																																																																															
14	NUMBER NUMBER DATE AMOUNT AVAILABLE AMOUNT																																																																															
15	.XXXX X-----X XX/XX/XX XX,XXX.XX XXX.XX XX,XXX.XX X																																																																															
16																																																																																
17	(VNDNBR) (INVNBR) (DUDATE) (MERCH) (DCTAYL) (NET) (STATUS)																																																																															
18																																																																																
19																																																																																
20																																																																																
21																																																																																
22																																																																																
23																																																																																
24																																																																																

4.

DATA DESCRIPTION SPECIFICATIONS

DISPLAY FILE

		MEMBER APDSLCT							
SEQUENCE		1	2	3	4	5	6	7	8
1.00	A								REF(APPFREF)
2.00	A		R DTARCD						SFL
3.00	A		R CTLRCD						SFLCTL(DTARCD)
4.00	A								02 21'VENDOR INVOICE SELECTION'
5.00	A								04 03'TO DISPLAY ALL INVOICES.'
6.00	A								04 28'PRESS ENTER.'
7.00	A								06 03'TO DISPLAY INVOICES DUE BEYOND A'
8.00	A								06 36'SPECIFIC DATE, KEY THE YEAR'
9.00	A		YR	R	Y	B			06 64DSPATR(CS RI)
10.00	A								REFFLD(DUYR)
11.00	A								06 68'AND'
12.00	A								07 07'MONTH/DAY'
13.00	A		MODY	R	Y	B			07 17DSPATR(CS RI)
14.00	A								REFFLD(DUMODY)
15.00	A								07 23', THEN PRESS ENTER'
16.00	A								09 3'TO END JOB, PRESS CMD KEY 1'
17.00	A								13 3'VENDOR INVOICE DUE'
18.00	A								13 32'GROSS DISCOUNT NET'
19.00	A								13 60'STATUS'
20.00	A								14 3'NUMBER NUMBER DATE'
21.00	A								14 31'AMOUNT AVAILABLE AMOUNT'

Machine Exercise - Subfiles-Inquiry

Sign on by entering the password GLCRPG.

The purpose of this exercise is for you to enter, compile and test your Vendor Invoice Display program.

Please read all instructions before you start to do them.

1. Create a source member named APDSLCLxx for your display file in the RPGSRC file.
2. Copy all the data description specifications from APDSLCT into your member.
3. Enter the additional data description specifications you coded for the subfile.
4. Create your new display file with the name APDSLCLxx and place it in the GLCRPG library.

5. You are to have your own set of data files (APLDUExx and APPOPYxx). To obtain a copy of your physical file, enter the command:

CALL LAB4 PARM(xx)

6. Create a source member named APLDUExx for your logical file in the RPGSRC file.
7. Copy all the data description statements from the member APLDUE into your member. Modify the PFILE parameter by changing APPOPY to your file name APPOPYxx.
8. Create your logical file APLDUExx in the GLCRPG library.
9. Create a source member named APR4xx in the RPGSRC file and enter your program statements.
10. Create your object program APR4xx in the GLCRPG library.
11. Execute your program. Test your program by entering dates between Jan. 1, 1980 and April 30, 1980.
12. Your displays should look similar to the ones on the next page.



When you have completed the Subfile-Inquiry machine exercise, return to Module Text 7 to continue the course.

**SAMPLE SUBFILE DISPLAY FOR YOUR SUBFILE-INQUIRY
EXERCISE - APR4xx**

VENDOR INVOICE SELECTION

TO DISPLAY ALL INVOICES, PRESS ENTER.

TO DISPLAY INVOICES DUE BEYOND A SPECIFIC DATE, KEY THE YEAR 00 AND
MONTH/DAY 0000 , THEN PRESS ENTER

TO END JOB, PRESS CMD KEY 1

VENDOR NUMBER	INVOICE NUMBER	DUE DATE	GROSS AMOUNT	DISCOUNT AVAILABLE	NET AMOUNT	STATUS
10502	872	1/01/80	500.00	50.00	450.00	C
88714	147	1/15/80	600.00	60.00	540.00	C
73013	812	1/30/80	400.00	40.00	360.00	C
56567	923	2/01/80	100.00	10.00	90.00	C
21178	124	2/15/80	120.00	12.00	108.00	C
07733	536	2/28/80	150.00	15.00	135.00	C
00612	138	3/01/80	300.00	30.00	270.00	C
07374	36	3/15/80	600.00	60.00	540.00	C +

VENDOR INVOICE SELECTION

TO DISPLAY ALL INVOICES, PRESS ENTER.

TO DISPLAY INVOICES DUE BEYOND A SPECIFIC DATE, KEY THE YEAR 00 AND
MONTH/DAY 0000 , THEN PRESS ENTER

TO END JOB, PRESS CMD KEY 1

VENDOR NUMBER	INVOICE NUMBER	DUE DATE	GROSS AMOUNT	DISCOUNT AVAILABLE	NET AMOUNT	STATUS
11002	24	3/30/80	500.00	50.00	450.00	C
06242	37	4/01/80	100.00	10.00	90.00	C
84210	116	4/15/80	150.00	15.00	135.00	C
72302	84	4/30/80	50.00	5.00	45.00	C

Coding Exercise - Subfiles-Update

The purpose of this exercise is for you to write an RPG III program that updates a data base file. In this exercise you are to add an update function to the program you wrote for the Subfile-Inquiry exercise of Module 7.

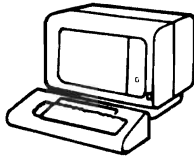
Given the same subfile display requirements as in the previous exercise, add the additional function to your program to allow the operator to select vendor invoices for payment.

You will find documentation (HIPO diagrams, DDS coding for logical and physical files, display layout) on the following pages.

1. Review the HIPO diagram that follows.

Note that the diagram is the same as in the previous exercise except it now includes the update function.
2. See the screen layout that follows. The additional information added to this display is shown blocked.
3. Your program APR5xx is to access the same Vendor Invoice file (APLDUExx) as in the Subfile-Inquiry exercise. Recall the Vendor Invoice file is keyed on invoice number within due date (DUYR, DUMODY, INVNBR). These fields must be defined in your subfile record. See the data description statements for your data base files.
4. Use the following subfile update process flowchart for coding the update function.
5. Examine the complete structured flowchart and note where the new update function should be placed.

6. The additional subfile record and subfile control record information for your display file APDSLCLxx is located in the source member APDUPDT and is listed in the documentation that follows. When you do the machine exercise that follows, you are to copy this new code into your display file source specifications.
7. Code the new update function for your program.
8. Check your new coding with the update flowchart.



When you are done with the coding, ask your Administrator for the display subfile solution and the Subfile-Update program coding solution. Compare the results and resolve any differences.

When you are ready, do the Subfile-Update machine exercise that follows.

HIPO DIAGRAMS

DIAGRAM ID: 3.2.1

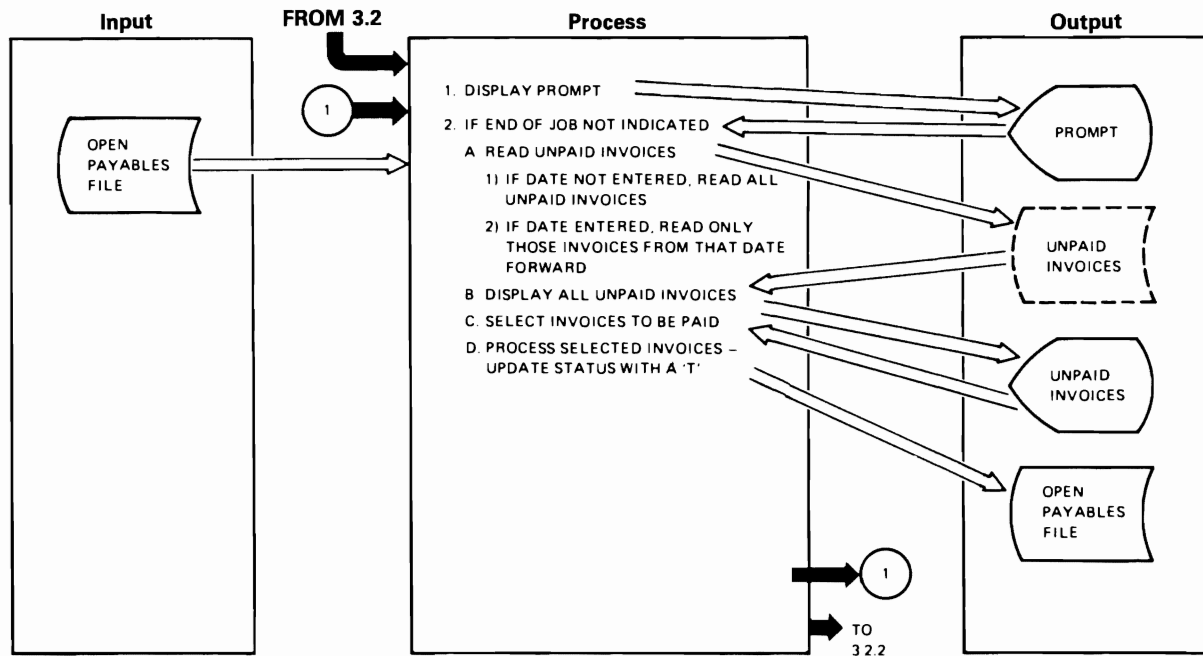


DIAGRAM 3.2.1

Extended Description

Notes		Ref.
Display the subfile control record. The operator will press enter to display all invoices or enter a date and then press enter to display the invoices due from that date forward.		1
The program should allow the operator to end the job anytime command key one is pressed.		2
Build a subfile of all the records in the Open Payables File.		2.A
If a date was entered, a search key must be built and the file pointer set.		2.A.2
Operator should be able to select any invoice by keying an 'X' into the PAY field on the subfile display. The operator will press Command Key 5 to update the invoices.		2.C
For those invoices selected update the record's status code with a 'T'.		2.D

SCREEN LAYOUT

COLUMN

	1-10										11-20										21-30										31-40										41-50										51-60										61-70										71-80									
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
01																																																																																
02	VENDOR INVOICE SELECTION																																																																															
03																																																																																
04	TO DISPLAY ALL INVOICES, PRESS ENTER.																																																																															
05																																																																																
06	TO DISPLAY INVOICES DUE BEYOND A SPECIFIC DATE, KEY THE YEAR LL AND																																																																															
07	MONTH/DAY LLLL, THEN PRESS ENTER.																																																																															
08																																																																																
09	TO END JOB, PRESS CMD KEY 1.																																																																															
10																																																																																
11	TO UPDATE THE INVOICES, PRESS CMD KEY 5																																																																															
12																																																																																
13	VENDOR INVOICE DUE GROSS DISCOUNT NET STATUS PAY																																																																															
14	NUMBER NUMBER DATE AMOUNT AVAILABLE AMOUNT																																																																															
15	.XXXXX X-----X XX/XX/XX XX,XXX.XX XXX.XX XX,XXX.XX X LI																																																																															
16																																																																																
17	(VNDNR) (INVNR) (DUDATE) (MERCH) (DCTAVL) (NET) (STATUS) (PAY)																																																																															
18																																																																																
19																																																																																
20																																																																																
21																																																																																
22																																																																																
23																																																																																
24																																																																																

**DATA BASE FILES FOR VENDOR INVOICE SELECTION
PROGRAM - APR5xx**

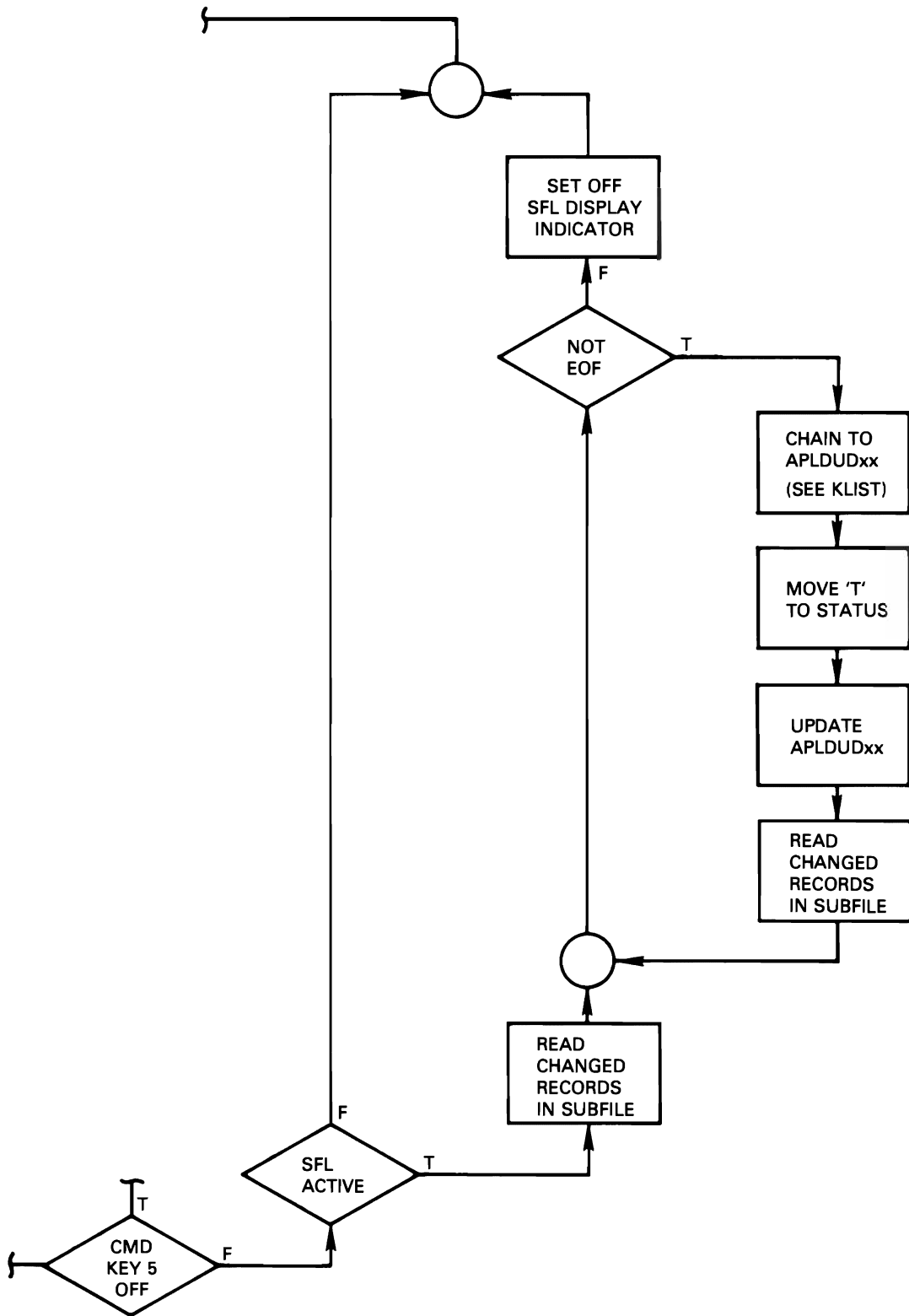
MEMBER: APLDUE

SEQNBR#	1	2	3	4	5	6	7	8
100	A	R PAYBLE						PFILE(APPOPY)
200	A	PORNBR						
300	A	VNDNBR						
400	A	INVNBR						
500	A	DATREC						
600	A	MERCH						
700	A	DCTAVL						
800	A	NET						
900	A	STATUS						
1000	A	DUDATE						CONCAT(DUMODY DUYS)
1100	A	DUMODY						
1200	A	DUYS						
1300	A	K DUYS						
1400	A	K DUMODY						
1500	A	K INVNBR						

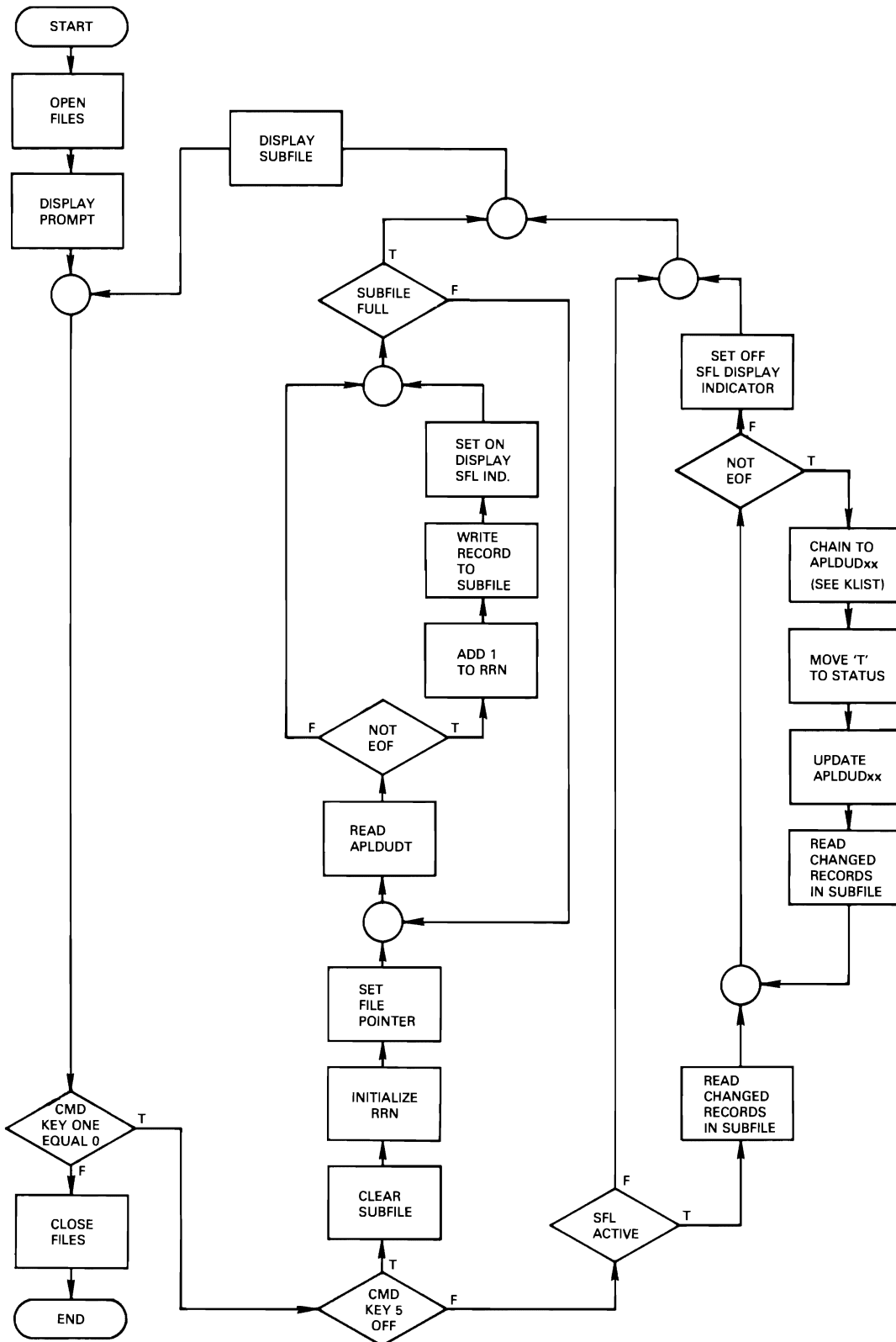
MEMBER: APPOPY

SEQNBR#	1	2	3	4	5	6	7	8
100	A							REF(APPFREF)
200	A	R PAYBLE						TEXT(*OPEN PAYABLES RECORD*)
300	A	PORNBR	R					
400	A	VNDNBR	R					
500	A	INVNBR	R					
600	A	DATREC	R					
700	A	MERCH	R					
800	A	DCTAVL	R					
900	A	NET	R					
1000	A	STATUS	R					
1100	A	DTPAID	R					
1200	A	CHECK#	R					
1300	A	DUMODY	R					
1400	A	DUYS	R					

SUBFILE UPDATE PROCESS FLOWCHART



VENDOR INVOICE SELECTION PROGRAM - APR5xx FLOWCHART



PROVIDED DISPLAY SPECIFICATIONS FOR APR5xx

		MEMBER: APDUPDT							
SEQNBR*		1	2	3	4	5	6	7	8
100	A*								
200	A		FAY		1A	I 15	69DSPATR(CS)		
300	A		DUYR	R		H			
400	A		DUMODY	R		H			
500	*								
600	A						CF05(05 'UPDATE INVOICES')		
700	A*								
800	A					11 03'	TO UPDATE THE INVOICES, PRESS'		
900	A					11 33'	CMD KEY 5.'		
1000	A*								
1100	A					13 68'	FAY'		

Machine Exercise - Subfiles-Update

Sign on by entering the password GLCRPG.

In this exercise you are to enter, compile and test your Vendor Invoice Selection program.

Please read all instructions before starting to do them.

1. The additional subfile data for your display file APDSLCxx is located in the source member APDUPDT. Copy this new code into your display file APDSLCxx at the appropriate locations. The source member is shown on the following page.

Note: Add the VALUES ('X') keyword for the PAY field as shown in the illustration of the APDUPDT source member on the following page.

**PROVIDED DISPLAY SPECIFICATIONS FOR
SUBFILES-UPDATE EXERCISE**

		MEMBER: APDUPDT														
SEQNBR*	...	1	...	2	...	3	...	4	...	5	...	6	...	7	...	8
100		A*														
200		A		PAY				1A	I	15	69DISPATR(CS) VALUES('X')					
300		A		DUYR		R					H					
400		A		DUMDIY		R					H					
500		*														
600		A														
700		A*														
800		A														
900		A														
1000		A*														
1100		A														

2. Create your display file.
3. Create a source member in the RPGSRC file by the name of APR5xx.
4. Copy your source code from APR4xx into your new member APR5xx.
5. Add your code for the update function to APR5xx.

6. Create your object program APR5xx in the GLCRPG library.
7. Before executing your program, review the "BEFORE" listing below.

FILE LIST BEFORE EXECUTING APR5xx

EXERCISE 5 TEAMWW							
VENDOR NUMBER	INVOICE NUMBER	PO NUMBER	DUE DATE	GROSS AMOUNT	DISCOUNT AVAILABLE	NET AMOUNT	STATUS
10502	872	600050	1/01/80	500.00	50.00	450.00	C
88714	147	600060	1/15/80	600.00	60.00	540.00	C
73013	812	600070	1/30/80	400.00	40.00	360.00	C
56567	923	600080	2/01/80	100.00	10.00	90.00	C
21178	124	600090	2/15/80	120.00	12.00	108.00	C
7733	536	600100	2/28/80	150.00	15.00	135.00	C
612	138	600110	3/01/80	300.00	30.00	270.00	C
7374	36	600120	3/15/80	600.00	60.00	540.00	C
11002	24	600130	3/30/80	500.00	50.00	450.00	C
6242	37	600140	4/01/80	100.00	10.00	90.00	C
84210	116	600150	4/15/80	150.00	15.00	135.00	C
72302	84	600160	4/30/80	50.00	5.00	45.00	C

8. Run your program. Enter the pay date of Jan. 30, 1980.
9. Select invoices 923, 36 and 116 for payment by entering an x for them in the PAY column. This is shown in the sample display that follows.

SAMPLE SUBFILE DISPLAY BEFORE UPDATE

VENDOR INVOICE SELECTION							
TO DISPLAY ALL INVOICES, PRESS ENTER.							
TO DISPLAY INVOICES DUE BEYOND A SPECIFIC DATE, KEY THE YEAR 80 AND MONTH/DAY 0130 , THEN PRESS ENTER							
TO END JOB, PRESS CMD KEY 1							
TO UPDATE THE INVOICES, PRESS CMD KEY 5.							
VENDOR NUMBER	INVOICE NUMBER	DUE DATE	GROSS AMOUNT	DISCOUNT AVAILABLE	NET AMOUNT	STATUS	PAY
56567	923	2/01/80	100.00	10.00	90.00	C	X
21178	124	2/15/80	120.00	12.00	108.00	C	
07733	536	2/28/80	150.00	15.00	135.00	C	
00612	138	3/01/80	300.00	30.00	270.00	C	
07374	36	3/15/80	600.00	60.00	540.00	C	X
11002	24	3/30/80	500.00	50.00	450.00	C	
06242	37	4/01/80	100.00	10.00	90.00	C	
84210	116	4/15/80	150.00	15.00	135.00	C	X

10. After selecting vendor invoices for payment, press command key 5 to update the invoices.

11. To verify that your file APLDUExx was correctly updated, enter the command:

CALL LIST5 PARM(xx)

Your listing should be the same as the "AFTER" listing.

FILE LIST AFTER EXECUTING APR5xx

EXERCISE 5 TEAMWW							
VENDOR NUMBER	INVOICE NUMBER	PO NUMBER	DUE DATE	GROSS AMOUNT	DISCOUNT AVAILABLE	NET AMOUNT	STATUS
10502	872	600050	1/01/80	500.00	50.00	450.00	C
88714	147	600060	1/15/80	600.00	60.00	540.00	C
73013	812	600070	1/30/80	400.00	40.00	360.00	C
56567	923	600080	2/01/80	100.00	10.00	90.00	T ◀
21178	124	600090	2/15/80	120.00	12.00	108.00	C
7733	536	600100	2/28/80	150.00	15.00	135.00	C
612	138	600110	3/01/80	300.00	30.00	270.00	C
7374	36	600120	3/15/80	600.00	60.00	540.00	T ◀
11002	24	600130	3/30/80	500.00	50.00	450.00	C
6242	37	600140	4/01/80	100.00	10.00	90.00	C
84210	116	600150	4/15/80	150.00	15.00	135.00	T ◀
72302	84	600160	4/30/80	50.00	5.00	45.00	C

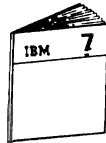
Also, you can verify that the correct records were selected by redisplaying the subfile. Your display should be the same as the "AFTER" display.

SUBFILE DISPLAY AFTER EXECUTING APR5xx

VENDOR INVOICE SELECTION							
TO DISPLAY ALL INVOICES; PRESS ENTER.							
TO DISPLAY INVOICES DUE BEYOND A SPECIFIC DATE, KEY THE YEAR 80 AND MONTH/DAY 0130 , THEN PRESS ENTER							
TO END JOB, PRESS CMD KEY 1							
TO UPDATE THE INVOICES, PRESS CMD KEY 5.							
VENDOR NUMBER	INVOICE NUMBER	DUE DATE	GROSS AMOUNT	DISCOUNT AVAILABLE	NET AMOUNT	STATUS	PAY
56567	923	2/01/80	100.00	10.00	90.00	T	◀
21178	124	2/15/80	120.00	12.00	108.00	C	
07733	536	2/28/80	150.00	15.00	135.00	C	
00612	138	3/01/80	300.00	30.00	270.00	C	
07374	36	3/15/80	600.00	60.00	540.00	T	◀
11002	24	3/30/80	500.00	50.00	450.00	C	
06242	37	4/01/80	100.00	10.00	90.00	C	
84210	116	4/15/80	150.00	15.00	135.00	T	◀

12. If you need a fresh copy of the file APLDUExx, enter the command:

CALL LAB5 PARM(xx)



When you have completed the Subfile-Update machine exercise, return to Module Text 7 to continue the course.



MODULE SUMMARY


You have performed the tasks involved in subfile programming:

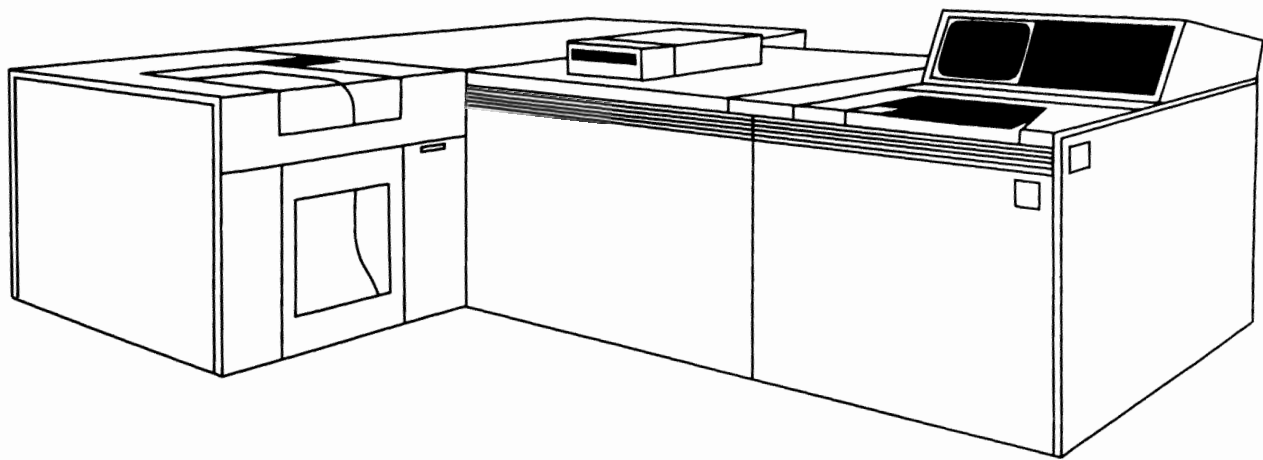
- Activating the subfile.
- Loading the subfile.
- Displaying the subfile.
- Processing the subfile.

You have found that structured programming is ideal for managing those tasks. Once the logic is defined it can be used in succeeding programs.

Subfiles are best suited for inquiry and file maintenance applications as you have experienced with the machine exercises.

You may have a need to manipulate data within a program and to transfer that data between programs as well as between jobs. Data areas are features that provide you this additional capability. Module 8 discusses what a data area is, its function, and the operation codes that control a data area.





Chapter 8. Module 8 - Data Areas

MODULE PURPOSE

Module 8 discusses the concepts, functions, and coding of data areas.

TIME ESTIMATE

1 hour and 30 minutes

MODULE OBJECTIVES

Upon completion of this module, using reference materials, you should be able to code an RPG III program that uses a data area.

TERMS

These terms are described/defined within the module.

- Data Area
- Local Data Area
- DEFN
- IN
- OUT
- UNLCK
- *NAMVAR
- *LOCK







VIDEO INTRODUCTION

Data areas are system objects that provide you with additional capability in satisfying your programming and application requirements. For example, data areas provide another way to manipulate data within your program and/or to transfer data between programs.

The video you are about to see illustrates what a data area is and its functions.

Video Summary

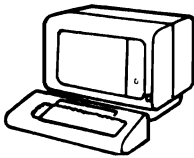
You have been given a brief overview of how data areas may be used to pass data values between programs. In this module, you will learn the RPG III coding for processing data areas.



Coding Exercise - Data Area with IN and OUT

The purpose of this exercise is to access a data area with the IN and OUT operation codes. In this exercise you are to add code to the Open Purchase Order File Maintenance program – POR2xx which you coded and executed in Module 5.

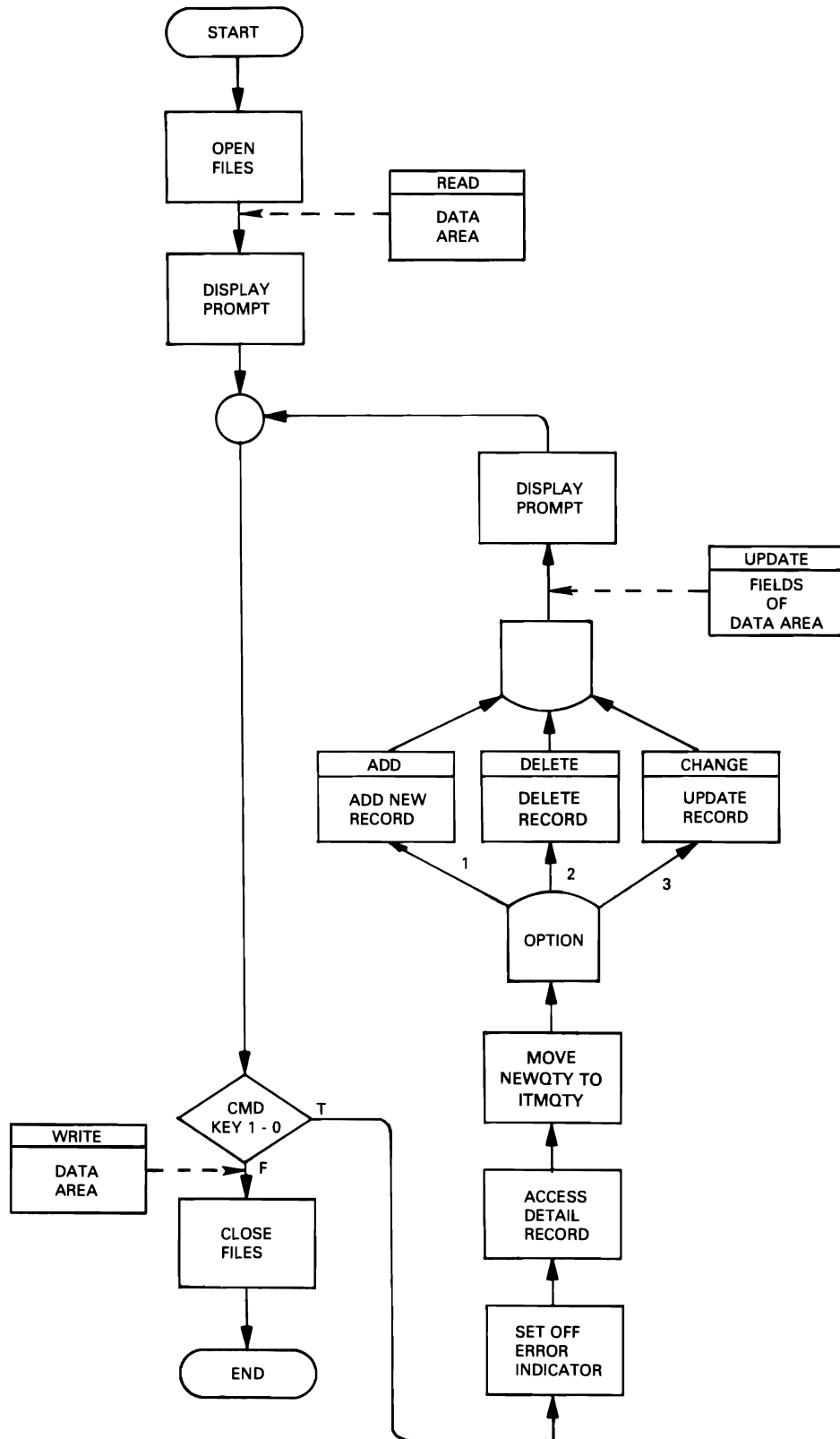
1. Your program (POR6xx) is to retrieve the data area value, update the value, and write it back to the data area.
2. A data area creation program is provided for you.
3. Remember, a data area stores data as a character string. Your program must separate that character string into individual fields for processing. Your data area, named TEAMxx, has a decimal length of six positions; the first two positions are for the transaction count and the last four positions are for the quantity total. Count all transactions except error transactions.
4. Review the flowcharts on the following two pages. Note the new entries for processing your data area with IN and OUT.
5. Code the new function using the IN and OUT operation codes.



When you have coded the new function, ask your Administrator for the solution. Compare the results and resolve any differences.

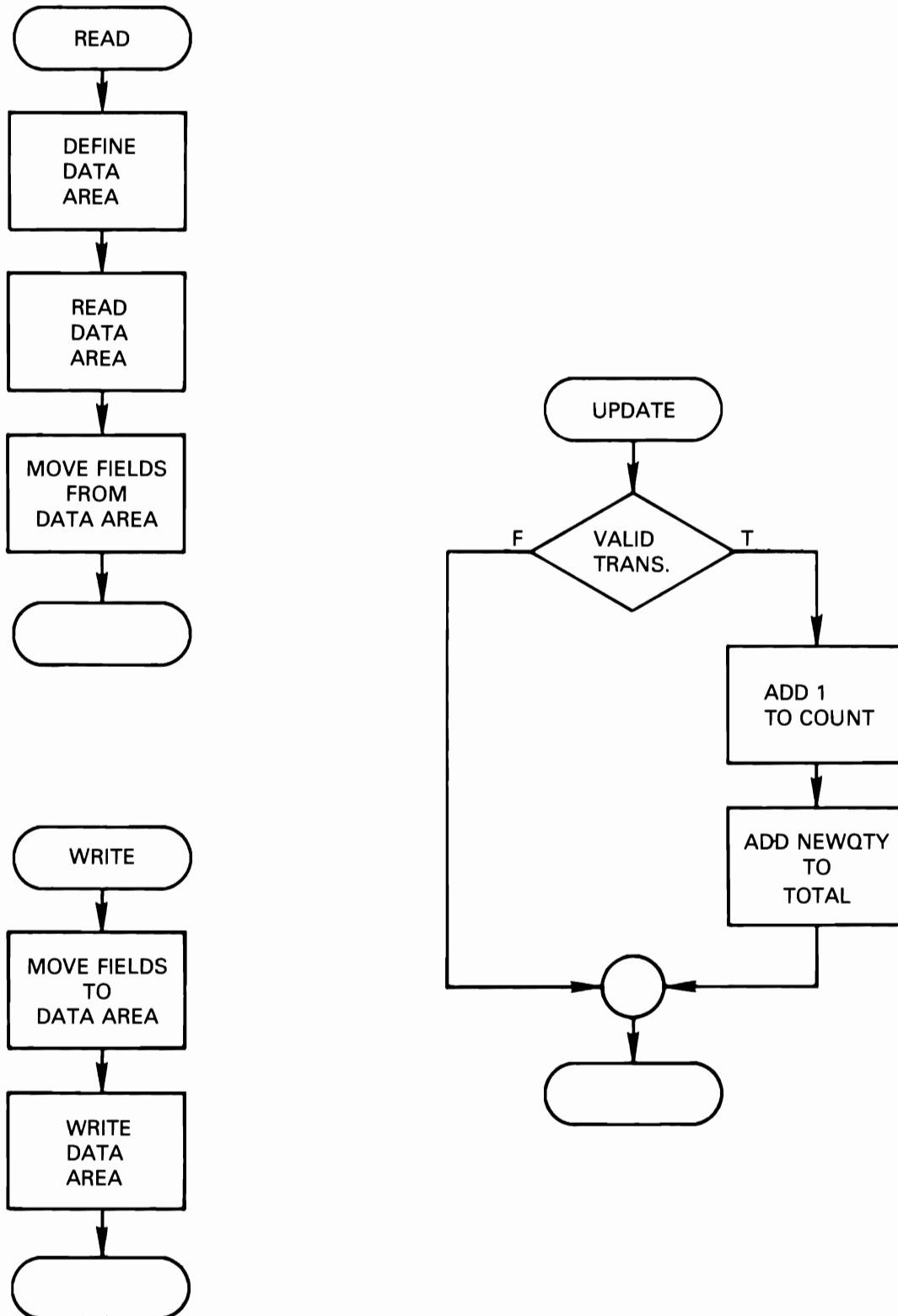
When ready, do the Data Area With IN and OUT machine exercise that follows.

**DATA AREA WITH IN AND OUT PROGRAM – POR6xx
FLOWCHART (Part 1 of 2)**



DATA AREA WITH IN AND OUT PROGRAM – POR6xx
FLOWCHART (Part 2 of 2)

NEW FUNCTION



Machine Exercise - Data Area with IN and OUT

Sign on by entering the password GLCRPG.

In this exercise you are to enter, compile and test your Data Area with the program you wrote using IN and OUT.

Please read all instructions before starting to do them.

1. Create a fresh set of files by entering the command:
`CALL LAB6 PARM(xx)`
2. Create an RPG source member named POR6xx in the RPGSRC file.
3. Copy the code from your source member POR2xx.
4. Add your new code for the data area function to POR6xx.

5. Create your object program POR6xx in the GLCRPG library.
6. Create your data area by entering the command:

CALL DAREA6 PARM(xx)

Your data area is named TEAMxx and is filled with zero characters.

7. Run your program and enter the following transactions:
(NOTE: It is not necessary to correct the error transaction.)

<u>Maintenance Option</u>	<u>Purchase Order No.</u>	<u>Item Number</u>	<u>Quantity</u>
1	400001	50	50
1	400001	450	25
2	400001	25	
3	400001	999	10
3	400001	150	ERROR 200

8. Verify that your program accumulated the correct totals by displaying your data area with the command:

DSPDTAARA TEAMxx

The value displayed should be:

40275

- 9.a) For a fresh set of files:

CALL LAB6 PARM(xx)

- b) For a fresh data area:

CALL DAREA6 PARM(xx)



When you have completed the Data Area With IN and OUT machine exercise, return Module 8 Text to continue the course.

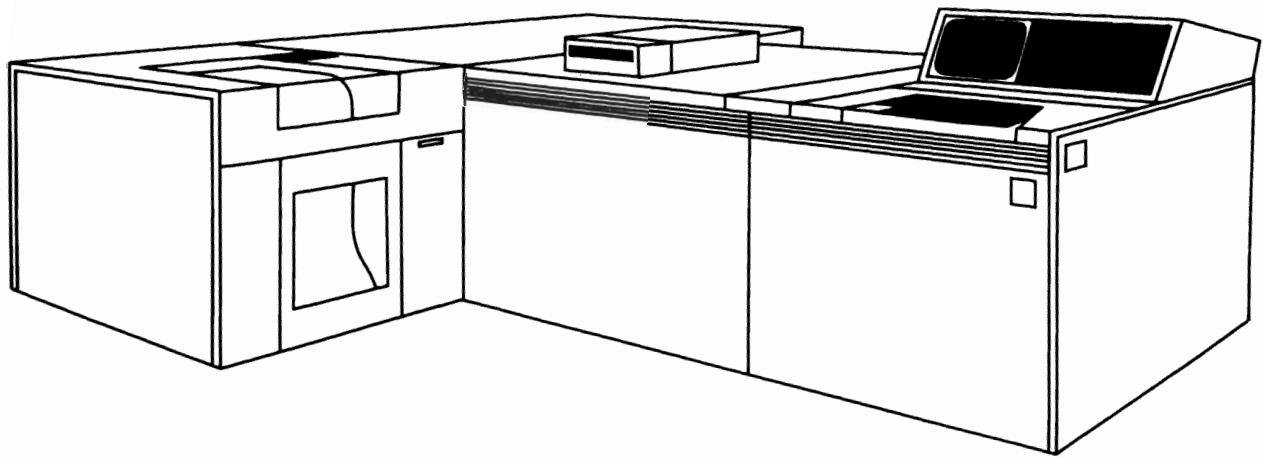
MODULE SUMMARY

You have used a data area with the IN and OUT operation codes. Your program:

- Retrieves the data area.
- Updates the data area.
- Writes the data area back to permanent storage.

You also displayed your data area to verify its contents.

Module 9 discusses how data areas are used with data structures instead of using the IN and OUT operations for retrieval and writing of data areas. Module 9 also discusses the concepts, functions, and coding of data structures.



Module 9 - Data Structures

MODULE PURPOSE

Module 9 discusses the concepts, functions, and coding of data structures.

TIME ESTIMATE

1 hour and 30 minutes

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- Code an RPG III program that uses a data structure.
- Write the two main functions of a basic data structure.
- List two differences between data areas and data structures.

TERMS

These terms are described/defined within the module.

- Data Structure
- Data Area Data Structure
- OCUR
- DS
- UDS

VIDEO INTRODUCTION

A data structure is an RPG III programming feature that allows you to define an area in storage and the layout of the fields called subfields within that area. A data structure be used to group fields or to divide a field into subfields.

The video you are about to see illustrates what a data structure is and its functions.

Video Summary

You have been given a brief overview of how data structures are defined and how they may be used in your program.

Module 9 will discuss additional RPG III coding of data structures.

TYPES OF DATA STRUCTURES

There are five types of data structures. You will find that the type of data structure determines its application.

- **Basic Type** - a single character string that allows field manipulation.
- **Multiple Occurrence** - a multiple occurrence data structure is multiple copies of the basic type. This type forms a series of data structures identical formats.
- **Data Area Data Structure** - RPG III automatically places a previously created data area into the data structure causing the data area to be redefined into specific fields.
- **Program Status and File Information** are special types and are discussed in the next module on "Handling Exception Errors".

- BASIC
- MULTIPLE OCCURRENCE
- DATA AREA
- PROGRAM STATUS
- FILE INFORMATION

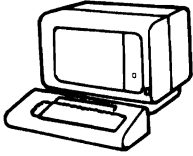
Coding Exercise - Simple Data Structure

The purpose of this exercise is to use a simple data structure to access a part of a field. In this exercise you are to add an edit check to the maintenance program of machine exercise 2 (Open Purchase Order File Maintenance program – POR2xx).

1. Review the flowchart that follows. Note the new entries for using a data structure.
2. Verify that each record to be deleted (option 2) has a date received month in the DATREC field equal to the current month. To simplify calculations, assume a current month of February. The coding requirements therefore are:
 month portion of DATREC = 02 (February)
3. Use a data structure to extract the month from the DATREC field. An appropriate error message for this edit is supplied for you in source member POD060. Note the error message indicator 50.
4. Code the new function for your program.

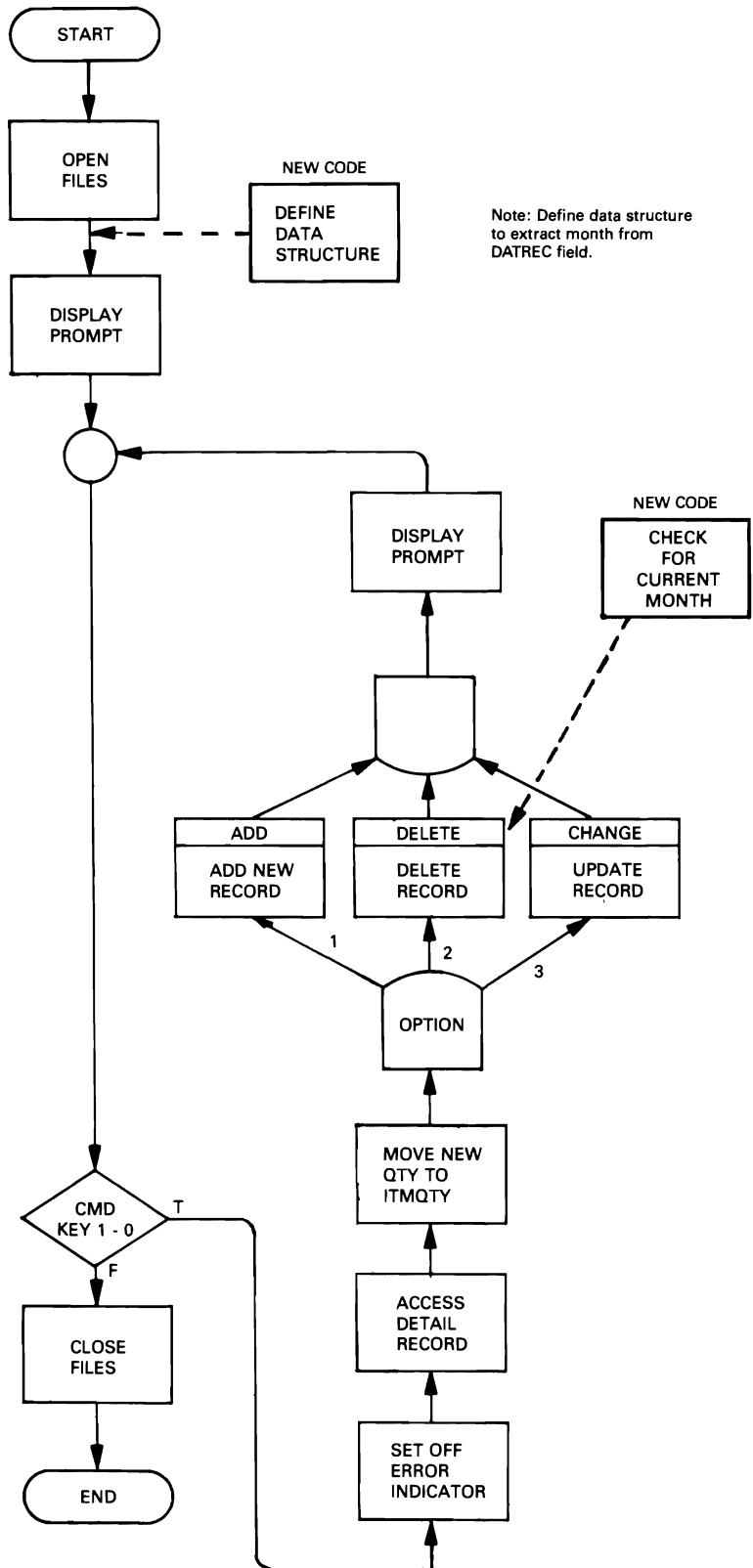
MEMBER * POD060			
SEQUENCE	1	2	3
1.00	A	50	ERRMSG('RECORD DATE IS NOT +
2.00	A	D	CURRENT MONTH')

When you have coded the new function, ask your Administrator for the solution. Compare the results and resolve any differences.



When you are ready, do the machine exercise "Simple Data Structure" that follows.

SIMPLE DATA STRUCTURE PROGRAM – POR7xx FLOWCHART



Machine Exercise - Simple Data Structure

Sign on by entering the password GLCRPG.

In this exercise you are to enter, compile and test your Simple Data Structure program.

Please read all instructions before starting to do them.

1. Add the new error message to the source code after the other error messages of your display file PODUPDxx. See the data description specifications below.

DATA DESCRIPTION SPECIFICATIONS

DISPLAY FILE

		PODUPDxx		DATA DESCRIPTION SOURCE					
SEQNBR	#	1	2	3	4	5	6	7	8
100	A								REF(APPREF)
200	A								CA01(01 'END OF JOB')
300	A		R PROMPT						ULINK
400	A					2	22		'OPEN PURCHASE ORDER FILE MAINTENAN
500	A								CE'
600	A					4	10		'MAINTENANCE'
700	A					5	10		'CODE OPTIONS'
800	A					7	15		'1. ADD AN ITEM (ENTER P.O.#, ITEM #,
900	A								QTY)'
1000	A					8	15		'2. DELETE AN ITEM (ENTER P.O.#, +
1100	A								ITEM #)'
1200	A					9	15		'3. MODIFY A QUANTITY ORDERED +
1300	A								(ENTER P.O.#, ITEM #, QTY)'
1400	A					13	13		'OPTION P.O.# ITEM # QTY'
1500	A								DSPATR(HI)
1600	A		OPTION		1	01	14		15'VALUES(1 2 3)
1700	A		PORNBR	R		1	14		21'CHECK(ME)
1800	A	10							ERRMSG('ITEM # OR P.O.# NOT FOUND')
1900	A	20							ERRMSG('CAN NOT ADD - ITEM ALREADY +
2000	A								EXISTS IN PURCHASE ORDER')
2100	A	50							ERRMSG('RECORD DATE IS NOT +
2200	A								CURRENT MONTH')
2300	A		ITMNR	R		1	14		31
2400	A		NEWQTY		5	01	14		41
2500	A								17 15'TO END THE JOB, PRESS CMD KEY 1.'
2600	A								19 15'TO ENTER A TRANSACTION, PRESS +
2700	A								ENTER'

The new code is located in a source member POD060 which is listed below. Because it is just two lines of code you may choose to enter it rather than copy it with the Browse/Copy function of the Services display.

		MEMBER	
		* POD060	
SEQUENCE	1	2	3
1.00	A 50		ERRMSG('RECORD DATE IS NOT +
2.00	A		CURRENT MONTH')

2. Create your display file PODUPDxx.
3. Create a fresh set of files by entering the command:
CALL LAB7 PARM(xx)
4. Create a source member in the RPGSRC file by the name of POR7xx.
5. Copy the code from your source member POR2xx and add the code for the new function.
6. Create your object program POR7xx in the GLCRPG library.

7. Before executing your program, see the "BEFORE" file listing:

FILE LISTING BEFORE EXECUTING POR7xx

MACH. EXPR. 7 TEAMWW									
PO NUMBER	VENDOR NUMBER	VENDOR NAME	ORDER AMOUNT	STATUS	ITEM NUMBER	ITEM DESCRIPTION	QUANTITY ORDERED	QUANTITY RECEIVED	DATE RECEIVED
400001	10	JOHN H. SMITH & SONS	500.00	C	25	LINED PAPER	25		0/00/00
					150	PEAKERS	100		0/00/00
					375	FOGERS	30	30	3/15/80
					475	RIBBONS	100	100	2/26/80
400005	11	FETZNER CFETZNER	110.00	C	50	SPOOLS	10	5	2/02/80
					100	INVITATIONS	15	15	2/12/80
					200	LADDERS	6	6	2/24/80
					400	FILES	8	8	2/16/80
400010	16	BRAND X BEEK	350.00	C	75	BOTTLES	500		0/00/00
					450	CANS	25		0/00/00
400015	431	MARGARET HART INC.	121.00	C	175	CARPETS	3	3	2/22/80
					250	SPINDLES	5	5	2/17/80
					275	MOLDS	2		0/00/00
400020	5075	PETER VAN RUTH CORP.	169.50	C	300	DISPLAYS	6	6	1/23/80
					425	CABINETS	9		0/00/00
400025	7374	PHOTO LABS INC.	45.50	C	500	PICTURES	13	13	2/05/80

8. Execute your program and enter the following transactions:

(NOTE: It is not necessary to correct the error transactions.)

<u>Maintenance Option</u>	<u>Purchase Order No.</u>	<u>Item Number</u>
2	400001	375
		ERROR
2	400001	475
2	400015	275
		ERROR
2	400015	250

9. To verify that the correct records were deleted, execute the list program by entering the command:

CALL LIST7 PARM(xx)

Your list should be the same as the "AFTER" file listing shown below.

FILE LISTING AFTER EXECUTING POR7xx

MACH. EXEC. 7 TEAMMM									
PO NUMBER	VENDOR NUMBER	VENDOR NAME	ORDER AMOUNT	STATUS	ITEM NUMBER	ITEM DESCRIPTION	QUANTITY ORDERED	QUANTITY RECEIVED	DATE RECEIVED
400001	10	JOHN M. SMITH & SONS	500.00	C	25	LINED PAPER	25		0/00/00
					150	BEAKERS	100		0/00/00
					375	EDGERS	30	30	3/15/80
400005	11	FETZNER CFETZNER	110.00	C	50	SPOOLS	10	5	2/02/80
					100	INVITATIONS	15	15	2/12/80
					200	LADDERS	6	6	2/24/80
					400	FILES	8	8	2/16/80
400010	16	BRAND X BEER	350.00	C	75	BOTTLES	500		0/00/00
					450	CANS	25		0/00/00
400015	431	MARGARET HART INC.	121.00	C	175	CARPETS	3	3	2/22/80
					275	MOLDS	2		0/00/00
400020	5075	PETER VAN RUTH CORP.	169.50	C	300	DISPLAYS	6	6	1/23/80
					425	CABINETS	9		0/00/00
400025	7374	PHOTO LABS INC.	45.50	C	500	PICTURES	13	13	2/05/80

10. If you need a fresh set of files, enter the command:

CALL LAB7 PARM(xx)



When you have completed the Simple Data Structure machine exercise, return to Module Text 9 to continue the course.



Progress Check: Unit 2

Record your answers in the space provided.

1. List two differences between data areas and data structures.

see Notes

2. What are the two main functions of a basic data structure?

a. To reference multiple fields as a single field.

b. To reference portions of a field as single fields (subfields)

3. What RPG III function must be used to format a composite key for a) an externally defined file, b) a program defined file?

a. KLIST for an externally described file.

b. Data structure for a program described file.

4. What operation codes must be coded in your program for you to control the reading and writing of data areas?

The operation codes required for you to read and write a data area are: IN, OUT, and DEFN.

5. What is the major advantage of an externally defined data structure?

The major advantage of an externally defined data structure is data integrity.

6. What type of data structure offers you an alternative to array processing?

The multiple occurrence data structure offers you an alternative to array processing.



When you have completed the questions, return to Module Text 9 to review your answers.

NOTES

1. List two differences between data areas and data structures
 - a. Data areas are stored external to the program. Data structures are stored internally within the program.
 - b. Data areas are created and deleted (except for local data areas) by a CH command. Data structures exist only for the duration of the program.
 - c. Data structures are character strings that RPG divides into fields. Data areas are character strings that can be divided into fields by RPG but not by CPF.

DATA STRUCTURE CHARACTERISTICS:

- Data structures like data areas are storage areas. However, they are not objects.
- Data structures are created by the program that uses them and only exist for the duration of that program.
- Data structures storage areas are initialized to blanks when created and are treated as a character string. On the input coding sheets, you are allowed to define sections of that character string as fields.

NOTES

General Coding Rules

A. Data Structures are defined on input specifications and must follow all input specifications for records.

B. Your data structure name is optional. (6 characters maximum) The name cannot be the same as a field name of the data structure.

C. 'DS' is required in positions 19-20 to define your data structure.

D. Data structure fields (subfields) are defined the same as data record fields.

E. Data structure length is determined by the total length of the fields (may be from 1 to 9999 characters in length).

F. The subfield length is specified in positions 48-51.




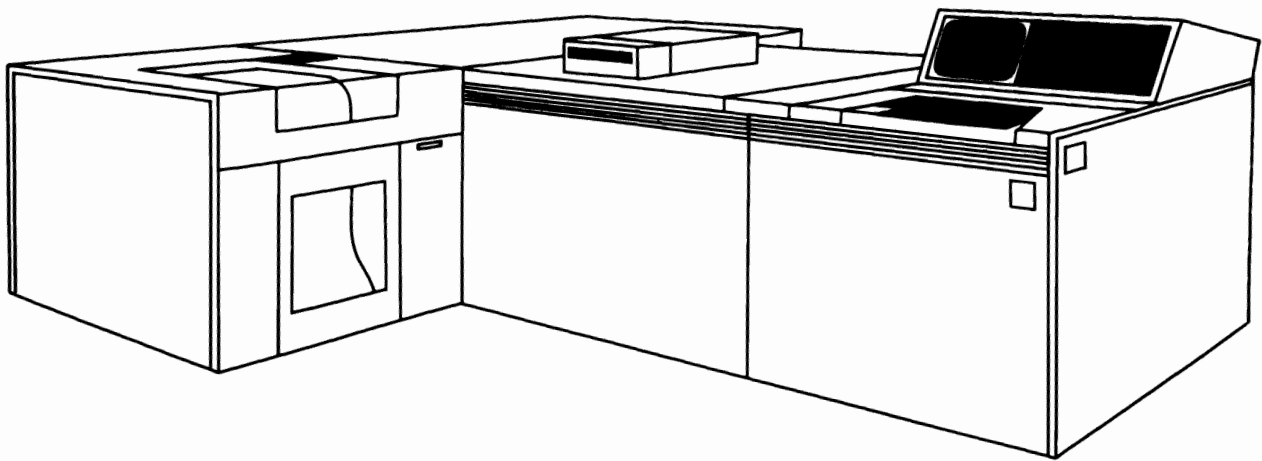
MODULE SUMMARY

RPG III allows you to define an area in program storage and the layout of the fields, called subfields, within that area. This area in storage is called a data structure. A data structure can be used to:

- Redefine a field as an array.
- Define the same field multiple times using different data formats.
- Divide a field into subfields.
- Group fields together.
- Define multiple occurrences of a set of data.

As a programmer, you can choose to let RPG III handle program and file exception errors or you can choose to control the program logic yourself when an exception error occurs. Module 10 discusses handling exception errors.





Chapter 10. Module 10 - Handling Exception Errors

MODULE PURPOSE

Module 10 discusses how your program can manage exception errors by using file information and program status data structures.

TIME ESTIMATE

1 hour and 45 minutes

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- Write an RPG III program that includes the coding for;
 - a file information data structure,
 - a program status data structure,
 - and an exception error subroutine to process information either a file information or a program status data structure.

TERMS

These terms are described/defined within the module.

- Exception Error
- File Information Data Structure
- Program Status Data Structure
- Return Point
- RPG III Default Error Handler
- F* • INFDS *identifies this data structure as a file*
- F* • INFSR *information data structure.*
- *PSSR

INFSR - in order for your subroutine to automatically receive control, it must be coded on the file description continuation line.

EXCEPTION ERRORS

Have you ever had a program terminate because a file was full or a called program was not found? Did the operator know what to do? Did the remote terminal know what to do?

With RPG III, you now have the tools to manage those exception errors that caused your programs to terminate abnormally.

An exception error is an error condition with a status code greater than 99.

Those codes less than 99 are considered normal error conditions and are managed by your normal coding. Some examples of normal code conditions that are not errors (normal completion of an operation): ending the display with a command key, end-of-file, record-not-found, etc.

NORMAL ERROR CODES	
0000	No exception/error
0002	Command key used to terminate display
0011	End of file on a read
0012	No-record-found condition on a CHAIN operation
0013	Subfile is full

Exception Error Codes

- 1011 Undefined record type
- 1211 I/O operation to a closed file
- 0102 divide by zero
- 0211 Program specified on CALL or FREE not found
- 0401 Data area specified on INPUT not found
- 0412 Data area not locked for output.

MANAGING EXCEPTION ERRORS

System/38 RPG III provides you with three ways of managing exception errors:

- By specifying a resulting indicator in position 56-57 of your calculation statement. The indicator will turn on when the operation fails.
- By specifying a file information data structure and a file exception error subroutine, file exception errors can be monitored.
- By specifying a program status data structure and a program exception error subroutine, program exception errors can be managed.

THREE METHODS TO MANAGE EXCEPTION ERRORS
<ul style="list-style-type: none">• INDICATOR IN POSITIONS 56-57 IN CALCULATIONS• FILE INFORMATION SUBROUTINES• PROGRAM STATUS SUBROUTINES

Coding Exercise - Exception Error

The purpose of this exercise is for you to use an exception error subroutine to manage a program exception error.

In this exercise you modify a copy of the program POR6xx that processed a data area with IN and OUT operations and that you coded in Module 8.

The program you wrote in Module 8 retrieved and updated a data area. In this exercise your program will not find the data area, causing the program exception/error code 00401 (a data area was not found on an IN or OUT operation) to be placed in the *STATUS subfield of your data structure.

1. When the program exception/error 00401 occurs, your program is to automatically branch to an exception error subroutine.
2. Your subroutine is to test for error code 00401.
3. Call the provided data area creation program POL800, and pass your identification (xx) to it.

Note: Program POL800 creates the data area used by your program. See the source code for POL800 on a following page.

4. Your program should handle the error condition via a subroutine without the operator being alerted that a problem occurred.
5. Also, in the same subroutine monitor for a file exception error on your data base logical file POLITMxx.

6. If an error other than 00401 occurs, display format ERRSRN and end the program. See the following documentation.
7. Code the OPEN operation code for your data base file POLITMxx in your program.
8. Study the program error subroutine that is included in the documentation.
9. Examine your program POR6xx from Module 8 exercise and determine where to add the new code.
10. Code the new function.

When you are done with the coding, ask your Administrator for Exception Error coding solution.



When you are ready, do the Exception Error machine exercise that follows.

PROVIDED DATA AREA CREATION PROGRAM
FOR MODULE EXERCISE 10
(EXCEPTION ERROR)

MEMBER: POL800

```
SEQNBR#... .. 1 ... .. 2 ... .. 3 ... .. 4 ... .. 5 ... .. 6 ... .. 7 ...  
100 START:   PGM          PARM(&T#)  
200          DCL          VAR(&T#) TYPE(*CHAR) LEN(2)  
300          DCL          VAR(&NAME) TYPE(*CHAR) LEN(6)  
400          CHGVAR       VAR(&NAME) VALUE((TEAM #CAT &T#))  
500          CRTDTAARA    DTAARA(&NAME.GLCRPG) TYPE(*DEC) LEN(6 0) +  
600          VALUE(0) TEXT('MACHINE EXERCISE 8 DATA AREA +  
700          FOR EXCEPTIUN ERRORS')  
800          RETURN  
900 END:     ENDPGM
```


**DATA DESCRIPTION SPECIFICATIONS AND DISPLAY
EXAMPLE OF ERRSRN RECORD FORMAT**

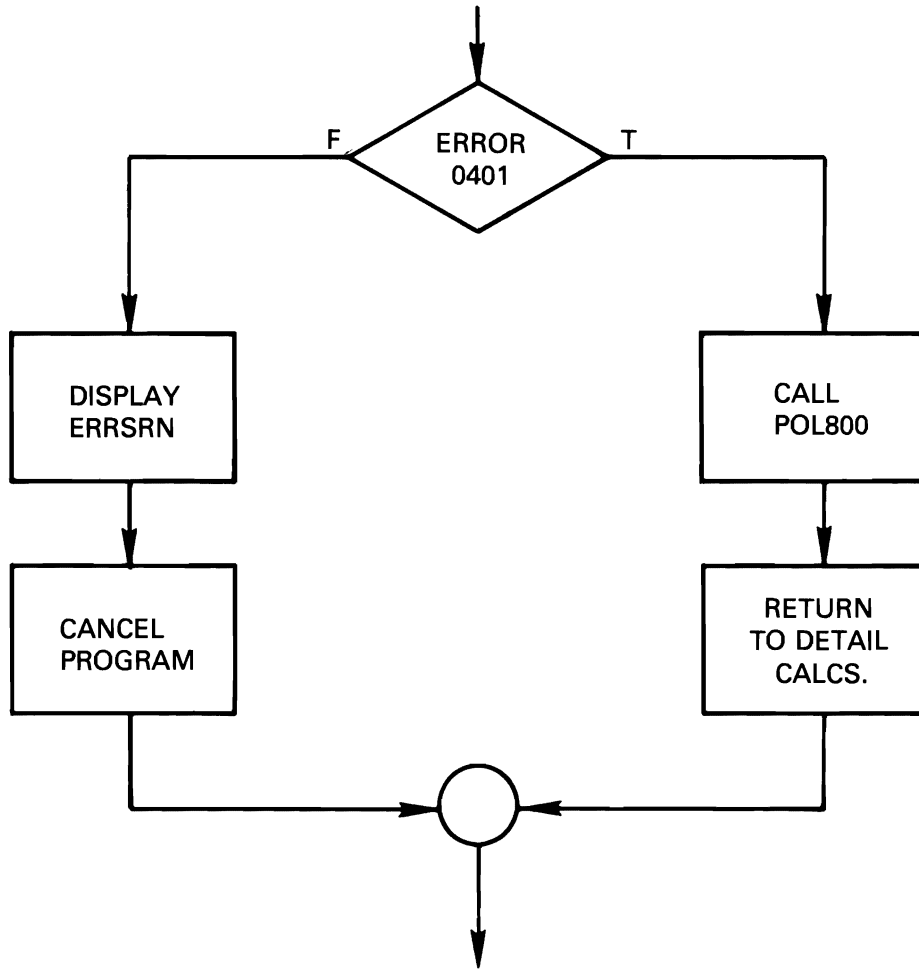
		MEMBER: ERRSRN	
SEQNBR#.....	1	2	3
100	A	R ERRSRN	
200	A		5 30* THERE IS A PROBLEM *
300	A		3 26*CALL THE DP DEPT - (333-2222)*
400	A		10 26*PRESS ENTER TO RETURN TO MENU*

THERE IS A PROBLEM

CALL THE DP DEPT - (333-2222)

PRESS ENTER TO RETURN TO MENU

**PROGRAM EXCEPTION ERROR SUBROUTINE
FLOWCHART**



Machine Exercise - Exception Error

Sign on by entering the password GLCRPG.

In this exercise you are to enter, compile and test your Exception Error program.

Please read all instructions before starting to do them.

1. Obtain a fresh set of files by entering the command:
CALL LAB8 PARM(xx)
2. Delete your data area by executing the command:
DLTDTAARA TEAMxx
3. Copy the source member ERRSRN into your display file source member PODUPDxx.

4. Recreate your display file PODUPDxx in the GLCRPG library.
5. Create a source member for your program and name it POR8xx.
6. Copy your program named POR6xx from Module 8 exercise into POR8xx and add the new code.
7. Create your object program POR8xx in the GLCRPG library.
8. Execute your program and enter the following transaction:

<u>Maintenance Option</u>	<u>Purchase Order No.</u>	<u>Item Number</u>	<u>Quantity</u>
1	400001	50	50

9. If your exception error subroutine successfully managed the error, your data area will have been created.

Display your data area with the command:

DSPDTAARA TEAMxx

It should have a value of:
10050.000000000

10. If you need to retest your program, remember to delete the data area before executing by entering the command:

DLTDTAARA TEAMxx

11. If you need a fresh set of files, enter the command:

CALL LAB8 PARM(xx)

12. Delete your logical file POLITMxx and execute your program again. You should see the error format. When you press the ENTER key, your program should end and return you to the programmer menu.

At this time, you have completed the machine exercises for the System/38 RPG III and Structured Programming course.

To remove your source members, objects, and identification from the system, select option 5 and enter the following command:

CALL RPGEND

At program completion, remove your listing from the printer and give it to your Administrator.



When you are ready, return to Module Text 10 to continue the course.





MODULE SUMMARY

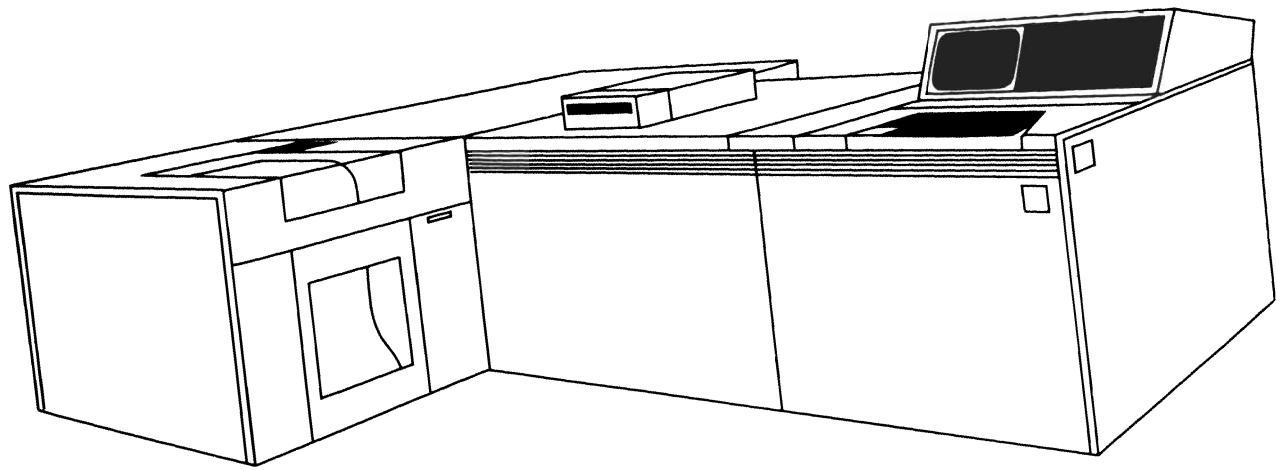
With RPG III, you have the option of controlling file exception and program exception errors.

You have a variety of ways to control those errors through:

- Resulting indicator in positions 56-57 of the calculation specifications
- File information data structure
- Program status data structure

At this point, you should have executed the procedure to remove your source members and objects from the class library. If not, please return to your work station, sign on, and enter the following command:

CALL RPGEND



Chapter 11. Module 11 - Additional RPG III Functions

MODULE PURPOSE

Module 11 discusses additional RPG III functions, such as additional enhancements to RPG, multiple device files, and commitment control.

TIME ESTIMATE

1 hour and 30 minutes

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- Describe how unit record devices would be coded in an RPG III program.
- List three reasons when a 'K' or continuation line is necessary for file descriptions.
- List two occasions when input specifications are required in an RPG III program.
- List one application for a multiple device file.
- Write the procedural steps for using a multiple device file.
- Describe commitment control.
- List and describe the RPG III operation codes for commitment control.

TERMS

These terms are described/defined within the module.

- DO
- READP
- Figurative Constants
- EXCPT
- CABxx
- SORTA
- Indicator Field and Array
- SHTDN
- TESTN
- TIME
- RENAME
- DEL
- PRTCTL
- Multiple Device Files
- Commitment Control



INPUT/OUTPUT OPERATION CODES

The following chart indicates whether an operation code can be used with the program logic cycle or full procedural processing and whether an operation code can be used with program or externally described files.

I/O Operation Codes				
E = COL 19 FILE SPECS				
F = COL 16 FILE SPECS				
OPERATION CODE	CYCLE	FULL PROC	EXT DESCR	PGM DESCR
— CHAIN		X	X	X
— READ		X	X	X
— READC		(2)	X	
— READE		X	X	X
— READP		X	X	X
— SETLL		X	X	X
— SETGT		X	X	X
— UPDAT	X	X	X	X
— WRITE	X	X	X	X
— DELETE	X	X	X	X
— OPEN	X	X	X	(1)
— CLOSE	X	X	X	X
— FEOD	X	X	X	X
— EXCPT	X	X	X	X
— EXFMT		(3)	X	

(1) OUTPUT FILE NOT USING 1P INDICATOR, ONLY
 (2) ONLY ALLOWED WITH SUBFILES
 (3) ONLY ALLOWED WITH DISPLAY FILE

UNIT 2 SUMMARY

Your task is to match the programming tools to meet your program requirements. The minimum executable RPG III program consists of calculation specifications only. This would probably be a subprogram.

AN RPG III PROGRAM

- ANY VALID GROUP OF RPG STATEMENTS CAN BE A PROGRAM
- CONTROL SPECIFICATION IS OPTIONAL — PROGRAM NAME IS GIVEN BY CRTRPGPGM COMMAND
- FILE DESCRIPTION SPECIFICATIONS ARE REQUIRED ONLY IF I/O IS TO BE PERFORMED
- INPUT AND OUTPUT SPECIFICATIONS ARE NOT NEEDED WITH EXTERNALLY DESCRIBED DB FILES

Progress Check: Unit 2

Record your answers in the space provided.

1. List three RPG III enhancements for output specifications.

- a. DEL
- b. PRTCTL
- c. Relative End Position
- d. EXCPT Group Name

2. When are input specifications required in a program?

- a. Define data structures
- b. Use record identifying field relations, →
- c. Use unit record devices
- d. Rename a field.

3. Which RPG III indicator cannot be referenced with the field *INxx?

IP

4. List three reasons why a continuation line ('K' line) would be specified in an RPG III program.

- a. continuation line upon subfiles
- b. Exception error handling
- c. RRV processing for output files
- d. RENAME
- e. PRTCTL
- f. Special files



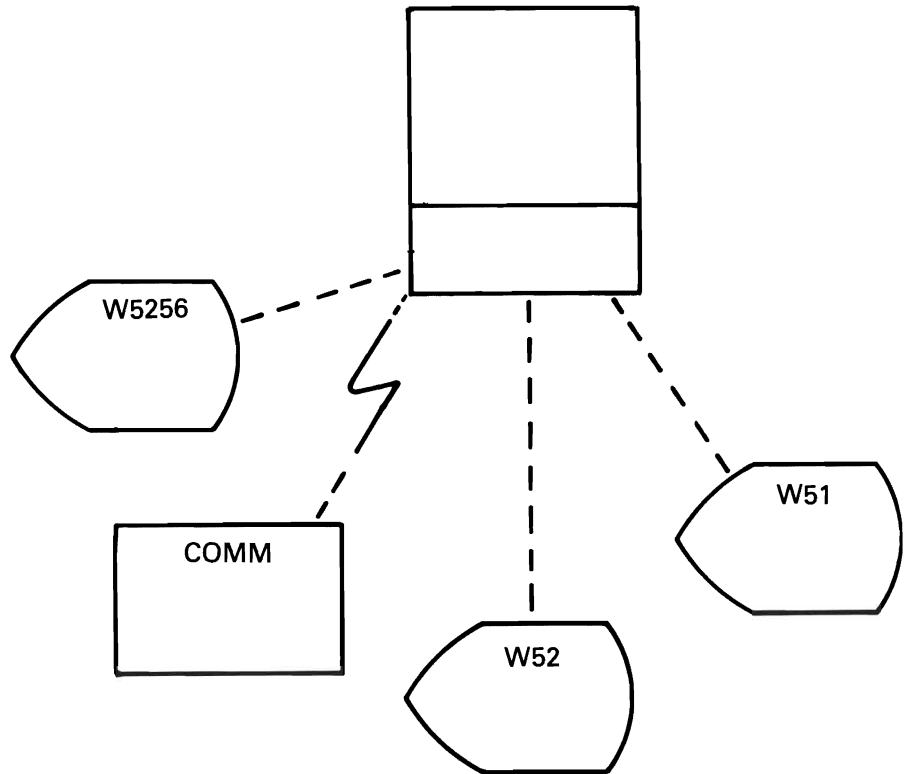
When you have completed the questions, return to Module Text 11 to review your answers.

→ control levels or matching record indicators.

APPLICATIONS FOR MULTIPLE DEVICE FILES

- Some programs must communicate with more than one device through one file. Using a multiple device file allows your program to wait for input from any of the devices at one point in the program.

The devices supported on a multiple device file are any combination of display devices and communication devices.



Another application would be for a program that requires only one device but needs some of the multiple device file functions. For example, you might want control returned to your program when no activity has occurred on a work station during a specified length of time.

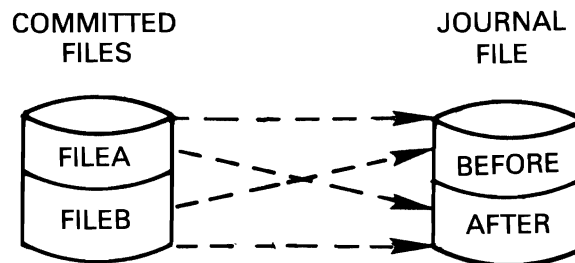
USING COMMITMENT CONTROL

Overview

Using commitment control you can define and process a number of changes to data base files as a single unit of work (or transaction). Commitment control requires journaling. The basic concept of journaling is to automatically record the changes to data base files as they occur. Commitment control extends journaling. Journaling is discussed in more detail in the System/38 Application Design Class.

Files under commitment control must be journaled to the same journal file, both before-images and after-images. You need not specify both images. Commit will put both images in the journal regardless of how the journal was specified, for the duration of commit processing. In other words, you don't have to redo all the journals to use commit.

When changes are made to these data base files, entries are placed in the journal file for each change generated by the transaction. When the transaction is completed, your program executes an RPG III COMMIT operation that places a separate entry on the journal to identify that the changes generated by the transaction are committed, that is, the changes are made permanently in the file.



MODULE SUMMARY

Module 11 discussed RPG III operations that you had not used in the machine exercises. However, your applications may require these additional functions of the RPG III language.

Module 11 also introduced you to multiple device files and commitment control with references for more detailed information.

Module 12 may be skipped. Please read the following before continuing the course.



Module 12 is written for those who may have the requirement to write RPG III programs using program described work station files or the requirement to maintain these programs. If you feel you need to know about program described work station files, return the text to your Administrator and ask for Module Text 12.



Module 13 illustrates how printer device files are coded and created as well as their probable uses as an alternative to program described printer files.

Should you choose to skip Module 12, return the text to your Administrator and ask for Module Text 13.

Chapter 12. Module 12 - Program Described Work Station Files

MODULE PURPOSE

Module 12 presents how program described work station files are coded and their possible uses as an alternative to externally described work station files.

TIME ESTIMATE

1 hour

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- List 3 reasons why a program described work station file should be used.
- List the considerations in coding the data description specifications for a program described work station file.
- Describe the functions of programs without a format name and those with a format name that use a program described work station file.

TERMS

These terms are described/defined within the module.

- Input/Output Buffers
- Record Identification
- Pass
- *NOID
- K1-K8 End Positions
- KA-KN, KP-KY Command Key Indicators

GENERAL CONCEPTS

General concepts of program described work station files are as follows.

Input and output specifications are used in the RPG III program to define the display file.

Because input and output specifications are used, you can code programs similar to some System/3 programs and System/34 work station programs. However, System/38 RPG III programs require less coding.

You also have the added flexibility of using structured programming techniques or the familiar RPG III program logic cycle.

GENERAL CONCEPTS
<ul style="list-style-type: none">• USE INPUT/OUTPUT SPECIFICATIONS TO DEFINE THE DISPLAY FILE• CAN BE CODED SIMILAR TO SOME<ul style="list-style-type: none">— SYSTEM/3 PROGRAMS— SYSTEM/34 WORK STATION PROGRAMS• CAN USE STRUCTURED PROGRAMMING• CAN USE THE RPG LOGIC CYCLE

Progress Check: Unit 2

Record your answers in the space provided.

1. List two reasons why program described work station files would be used.

2. List at least two coding considerations when specifying the data description specifications for a program described work station file.

3. When must KPASS *NOIND be specified on the file description specification for a program described work station file?

4. Are command key indicators KA-KN and KP-KY unique to RPG III programs using program described work station files?

5. How is a screen format name referenced on the output specification for a program described work station file?

MODULE SUMMARY

A program described work station file can be used with a format name or without a format name specified on your output specifications.

If you use a format name:

- Indicators may be passed to and from your program.
- Command keys may be defined.
- Input/output fields are listed in the same sequence as they are coded in DDS. However, the fields names do not have to be the same.

If you don't use a format name:

- No indicators are passed to your program.
- No command key indicators are defined.
- The input record is treated as a single field. The output record is treated as a string of characters. Each record displayed overlays the previous record displayed.

Module 13 discusses how to create a printer device file and the reasons why a printer device file would be used.

Module 13 - Printer Device Files

MODULE PURPOSE

Module 13 illustrates how printer device files are coded and created as well as their probable uses as an alternative to program described printer files.

TIME ESTIMATE

1 hour

MODULE OBJECTIVES

Upon completion of this module, you should be able to:

- List the specification entries for a printer device file.
- Code the control language command to create a printer device file.
- List 3 reasons why a printer device file would be used.
- Write how to control overflow for an external printer device file.

TERMS

These terms are described/defined within the module.

- UNDERLINE
- BLKFOLD
- INDTXT
- DFT
- SKIPA
- SKIPB
- SPACEA
- SPACEB
- PAGNBR

Overview

External printer device files offer an alternative to program described files.

Printer files are defined and created in the same manner as display, physical, and logical files. Refer to the chart.

PRINTER FILE DESCRIPTIONS
<ul style="list-style-type: none">• EXTERNAL TO ANY PROGRAM• DEFINED AS PRINTER FORMATS• ARE CODED ON DATA DESCRIPTION SPECIFICATIONS• MAY BE DEFINED AT THREE LEVELS<ul style="list-style-type: none">— FILE— RECORD— FIELD• ONE PRINTER FILE MAY CONTAIN ALL THE FORMATS FOR ONE PROGRAM OR ONE APPLICATION

Progress Check: Unit 2

Record your answers in the space provided.

1. List at least two reasons why a printer device file would be used instead of using the RPG III output specifications.

2. List two ways the overflow line for a printer device file can be monitored.

3. How can a report be retained in the spool file after it has printed?

4. When using full functional file processing, what RPG III operation permits controlling the printing logic within calculations without using an externally described file?



When you have completed the questions, return to Module Text 13 to review your answers.

MODULE SUMMARY

Printer device files offer additional tools to support your programming requirements such as:

- Use of special keywords - UNDERLINE, BLKFOLD.
- The same record can be printed under different formats by issuing multiple writes.
- Formats from different printer files can be printed on a single report.
- Printing logic is external to the program.
- Ease of use. For example, if there's a change in the forms for a particular application. Modify the printer data description specifications and recompile. In most instances, your program will not have to be changed.

You have now come to the end of System/38 RPG III and Structured Programming. At this time, go to the next section and read the "Course Summary".

Chapter 14. Course Summary

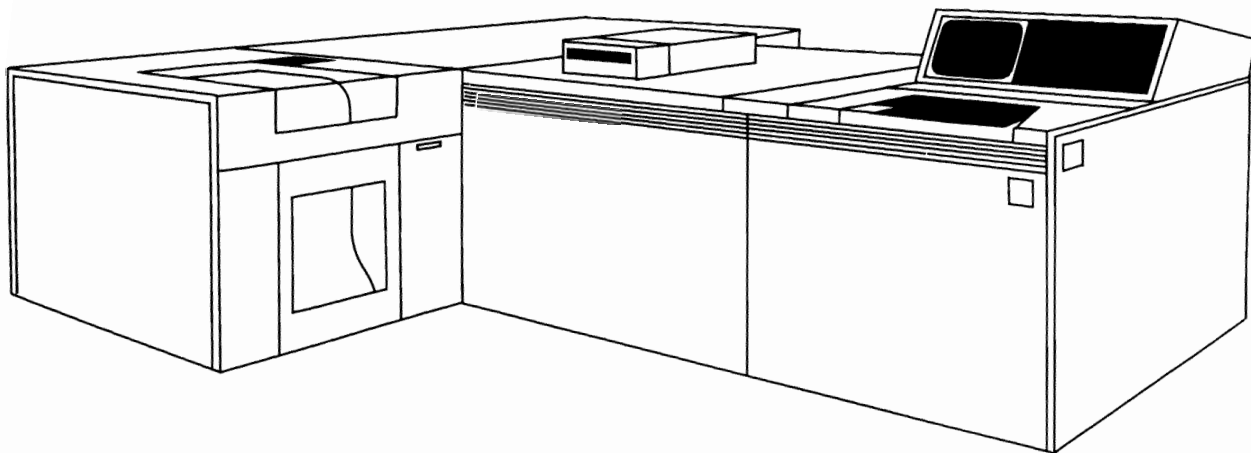
Now that you have completed the System/38 RPG III and Structured Programming course, you will want to continue to code and test programs on your own system.

You will find in the Appendix of the Student Materials Book a number of references (field reference file, job description, profile) that you may wish to use with your applications.

As a result of taking this course, you should be able to code, compile, and test RPG III structured programs for:

- Normal inquiry
- Inquiry with update
- File maintenance
- Data areas
- Data structures
- Subfiles
- Passing/receiving parameters
- Handling exception errors

Return your Module Texts to your Guided Learning Center Administrator. Remember, the Student Materials Book, flowcharts, program listings, and debugging templates are yours to keep. See your Administrator for final directions. Best wishes and good programming.



Appendix A. References

The appendix consists of your field reference file, job description, and user profile.

FIELD REFERENCE FILE

MEMBER
APPREF

PAGE
1

-----1-----2-----3-----4-----5-----6-----7-----8

```

A*
A** FIELD REFERENCE FILE FOR ACCOUNTS PAYABLE
A** STUDENT EXERCISES
A*
A          R REFFMT                      TEXT('FIELD REFERENCE FILE')
A*
A** FIELDS USED IN MULTIPLE RECORDS
A*
A          VNDNBR          5 0          TEXT('VENDOR ID NUMBER')
A          DEPTNO          3 0          COLHDG('VENDOR' 'NUMBER')
A          ACTCD           1           TEXT('DEPARTMENT CHARGED')
A          ACTCD           1           COLHDG('DEPARTMENT' 'NUMBER')
A          ACTCD           1           TEXT('A=ACTIVE D=DELETE S=SUSPEND')
A          ACTCD           1           COLHDG('ACTIVE' 'RECORD' 'CODE')
A*
A** FIELDS IN VENDOR MASTER RECORD
A*
A          VNDNAM          25          TEXT('VENDOR NAME')
A          VNDAD1          25          COLHDG('VENDOR' 'NAME')
A          VNDAD1          25          TEXT('ADDRESS LINE 1')
A          VNDAD2          25          COLHDG('ADDRESS LINE 1')
A          VNDAD2          25          TEXT('ADDRESS LINE 2')
A          VNDAD3          25          COLHDG('ADDRESS LINE 2')
A          VNDAD3          25          TEXT('ADDRESS LINE 3')
A          VNDAD3          25          COLHDG('ADDRESS LINE 3')
A          VNDZIP          5 0          TEXT('ZIP CODE')
A          VNDZIP          5 0          COLHDG('ZIP' 'CODE')
A          VNDACD          3 0          TEXT('AREA CODE')
A          VNDACD          3 0          COLHDG('AREA' 'CODE')
A          VNDPHN          7 0          TEXT('PHONE NUMBER')
A          VNDPHN          7 0          COLHDG('PHONE')
A          VTRMPC          3 3          TEXT('NORMAL PROMPT PAY DISCOUNT %')
A          VTRMPC          3 3          COLHDG('TERMS' '%')
A          VTRMDA          2 0          TEXT('NORMAL NO. OF DAYS FOR TERMS')
A          VTRMDA          2 0          COLHDG('TERMS' 'DAYS')
A          VNDCLS          2 0          TEXT('VENDOR CLASS')
A          VNDCLS          2 0          COLHDG('CLASS')
A          VNDCLS          2 0          TEXT('VENDOR SALESMAN')
A          VNDCLS          2 0          COLHDG('SALESPERSON')
A          DCTMTD          7 2          TEXT('DISCOUNT TAKEN THIS MONTH')
A          DCTMTD          7 2          COLHDG('DISCOUNT' 'TAKEN' 'MTD')
A          DCTYTD          9 2          TEXT('DISCOUNT TAKEN THIS YEAR')
A          DCTYTD          9 2          COLHDG('DISCOUNT' 'TAKEN' 'YTD')
A          PCHMTD          9 2          TEXT('PURCHASE THIS MONTH')
A          PCHMTD          9 2          COLHDG('PURCHASES' 'YTD')
A          PCHYTD          11 2         TEXT('PURCHASES THIS YEAR')
A          PCHYTD          11 2         COLHDG('PURCHASES' 'YTD')
A          BALOWE          9 2          TEXT('BALANCE OWED')
A          BALOWE          9 2          COLHDG('BALANCE' 'OWED')
A          SRVRTG          1           TEXT('SERVICE RATING')
A          SRVRTG          1           COLHDG('SERVICE' 'RATING')
A          DELRTG          1           TEXT('DELIVERY RATING')
A          DELRTG          1           COLHDG('DELIVERY' 'RATING')
A          SCHCOD          10A         TEXT('SEARCH CODE')
A          SCHCOD          10A         COLHDG('SEARCH' 'CODE')
    
```

(1 of 3)

-----1-----2-----3-----4-----5-----6-----7-----8

```

A          CHECK(ME)
A          COMNIS          25          TEXT('COMMENTS ABOUT THIS VENDOR')
A          COLHDG('COMMENTS ABOUT ' 'VENDOR')
A*
A** FIELDS USED IN TRANSACTION RECORD
A*
A          INVNBR          8          TEXT('VENDORS INVOICE AMOUNT')
A          COLHDG('VENDOR' 'INVOICE' 'NUMBER')
A          DATREC          6 0          TEXT('DATE RECEIVED')
A          COLHDG('DATE' 'RECEIVED')
A          MERCH          7 2          TEXT(MERCHANDISE AMOUNT')
A          EDTCDE(3)
A          COLHDG('MERCHANDISE' 'AMOUNT')
A          NET          7 2          TEXT('NET AMOUNT PAID')
A          EDTCDE(3)
A          COLHDG('NET' 'AMOUNT' 'PAID')
A          STATUS          1          TEXT('E=ENTERED T=TO PAY D=DELETE +
A          P=PAID')
A          COLHDG('STATUS' 'OF' 'RECORD')
A          DTPAID          6 0          TEXT('DATE PAID')
A          COLHDG('DATE' 'PAID')
A          CHECK#          6 0          TEXT('CHECK NUMBER')
A          COLHDG('CHECK' 'NUMBER')
A*
A** FIELDS USED IN DEPARTMENT MASTER
A*
A          DNAME          15          TEXT('DEPARTMENT NAME')
A          COLHDG('DEPARTMENT' 'NAME')
A          BUDMTD          9 2          TEXT('AMOUNT BUDGETED THIS MONTH')
A          COLHDG('BUDGET' 'AMOUNT' 'MTD')
A          BUDYTD          11 2          TEXT('AMOUNT BUDGETED THIS YEAR')
A          COLHDG('BUDGET' 'AMOUNT' 'YTD')
A          DISMTD          9 2          TEXT('AMOUNT DISPERSED THIS MONTH')
A          COLHDG('DISPERSED' 'MTD')
A          DISYTD          11 2          TEXT('AMOUNT DISPERSED THIS YEAR')
A          COLHDG('DISPERSED' 'YTD')
A          GLACT          6 0          TEXT('GENERAL LEDGER ACCOUNT NBR')
A          COLHDG('G/L' 'ACCOUNT' 'NUMBER')
A*
A* FIELDS USED IN ITEM MASTER FILE
A*
A          ITMNR          5 0          TEXT('ITEM NUMBER')
A          COLHDG('ITEM' 'NUMBER')
A          ITMDSC          25          TEXT('ITEM DESCRIPTION')
A          COLHDG('ITEM' 'DESCRIPTION')
A          ITMOOH          7 0          TEXT('QUANTITY ON HAND')
A          COLHDG('QUANTITY' 'ON HAND')
A          ITMOOO          7 0          TEXT('QUANTITY ON ORDER')
A          COLHDG('QUANTITY' 'ON ORDER')
A          ITMCST          5 2          TEXT('ITEM UNIT COST')
A          COLHDG('ITEM' 'UNIT' 'COST')
A          ITMPRC          5 2          TEXT('ITEM UNIT PRICE')
A          COLHDG('ITEM' 'UNIT' 'PRICE')
A          CATNBR          7          TEXT('VENDOR CATALOG NUMBER')
A          COLHDG('VENDOR' 'CATALOG' 'NUMBER')

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8-----

```

A*
A** FIELDS USED FOR PURCHASE ORDERS
A*
A          PORNBR          6  0          TEXT('PURCHASE ORDER NUMBER')
A          COLHDG('PURCHASE' 'ORDER' 'NUMBER')
A          DATORD          6  0          TEXT('DATE ORDERED')
A          COLHDG('DATE' 'ORDERED')
A          ITMAMT          7  2          TEXT('ITEM EXTENDED AMOUNT')
A          COLHDG('ITEM' 'EXTENDED' 'AMOUNT')
A          QTYORD          5  0          TEXT('ITEM ORDERED QUANTITY')
A          COLHDG('ITEM' 'ORDERED' 'QUANTITY')
A          QTYREC          5  0          TEXT('ITEM QUANTITY RECEIVED')
A          COLHDG('ITEM' 'QUANTITY' 'RECEIVED')
A          DATDUE          6  0          TEXT('INVOICE DUE DATE')
A          COLHDG('INVOICE' 'DUE' 'DATE')
A*
A** ADDITIONAL FIELDS
A*
A          DUVR            2  0          TEXT('VENDOR INVOICE DUE DATE +
A          - YEAR')
A          COLHDG('INVOICE' 'DUE' 'YEAR')
A          DUMODY          4  0          TEXT('VENDOR INVOICE DUE DATE +
A          - MONTH DAY')
A          COLHDG('INVOICE' 'DUE' 'MONTH-DAY')
A          DCTAVL          5  2          TEXT('VENDOR INVOICE DISCOUNT +
A          AVAILABLE')
A          EDTCDE(3)
A          COLHDG('INVOICE' 'DISCOUNT' +
A          'AVAILABLE')
A          DUDATE          6  0          TEXT('VENDOR INVOICE DUE DATE: +
A          MONTH-DAY-YEAR')
A          EDTCDE(Y)
A          COLHDG('DISPLAY' 'DUE-DATE' 'MO-DAY +
A          -YR')

```

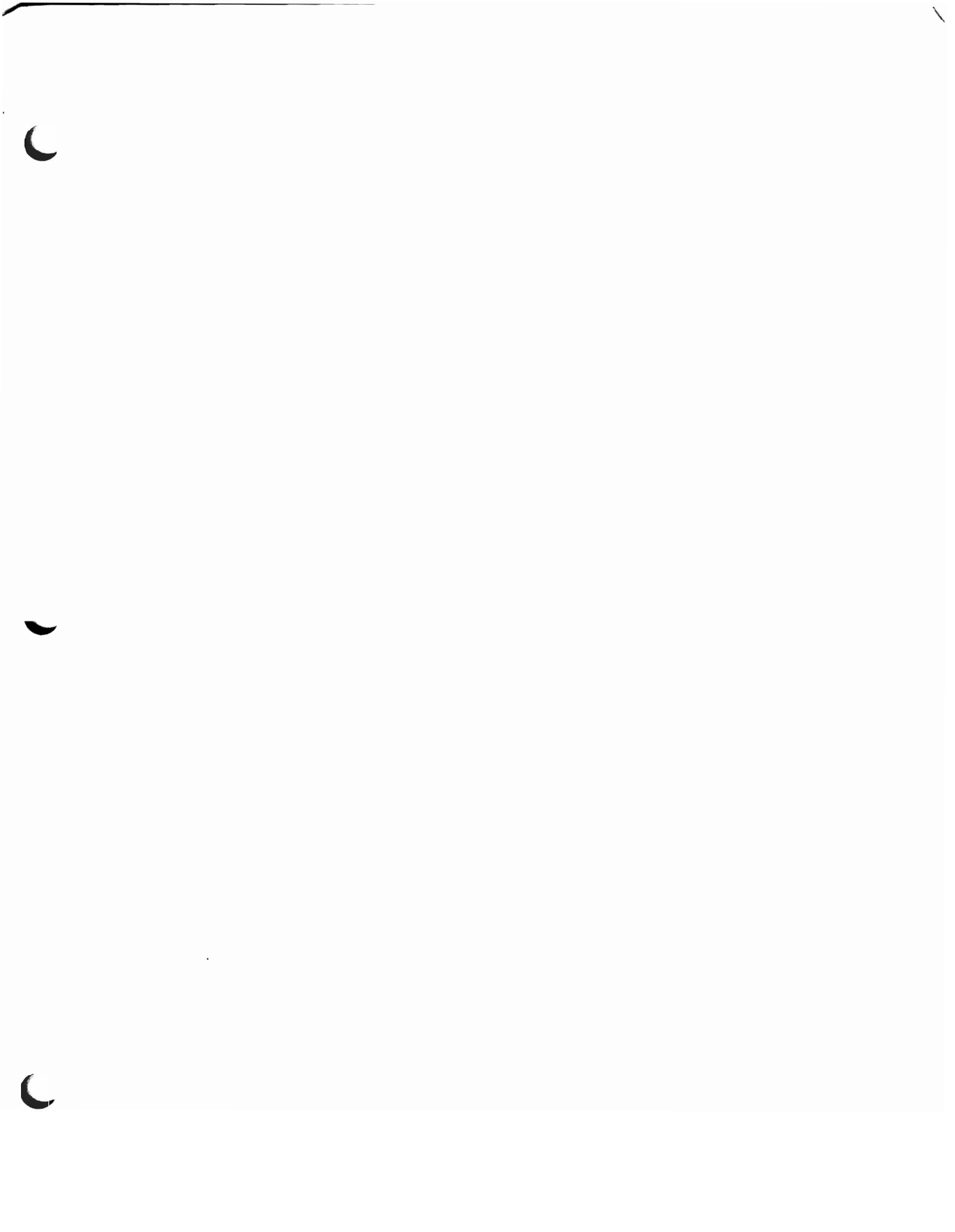
(3 of 3)

JOB DESCRIPTION

2/10/83 19:12:13	JOB DESCRIPTION
JOB DESCRIPTION: <u>GLCRPG</u>	JOB PRIORITY: 5
LIBRARY NAME: GLCRPG	JOB QUEUE NAME: QBATCH
USER PROFILE NAME: GLCRPG	LIBRARY NAME: QGPL
CL SYNTAX CHECK: 30	OUTPUT PRIORITY: 5
HOLD ON JOB QUEUE: *NO	OUTPUT QUEUE NAME: <u>GLCOUTQ</u>
CANCEL SEVERITY: 30	LIBRARY NAME: QGPL
JOB DATE: *SYSVAL	JOB LOGGING
JOB SWITCHES: 00000000	MESSAGE LEVEL: 3
	MESSAGE SEVERITY: 10
	TEXT LEVEL: *MSG
<hr/>	
2/10/83 19:12:46	JOB DESCRIPTION GLCRPG
ROUTING DATA: QCMDB	
REQUEST DATE: *NONE	
<hr/>	
2/10/83 19:13:02	JOB DESCRIPTION GLCRPG
INITIAL LIBRARY LIST: (READ BY COLUMNS)	
GLCRPG	
QGPL	
QTEMP	
QIDU	
QRPG	

PROFILE

10/13/86 13:56:06		USER PROFILE - GLCRPG		+++
BASIC INFORMATION				
Special authority:	SFCAUT	*JOBCTL		
Maximum storage allowed:	MAXSTG	*NOMAX		
Used:		00000000000		
Highest scheduling priority:	PTYLMT	5		
Initial program to call:	INLPGM	RGLSET		
Library name:		GLCRPG		
Job description:	JOB	*NONE		
Library name:				
Group profile:	GRPPRF	GLCGRP		
Owner (*USRPRF or *GRPPRF):	OWNER	*GRPPRF		
Group authority:	GRPAUT	*NONE		
Accounting code:	ACGCDE			
Message queue name:	MSGQ	*NONE		
Library:				
Output queue name:	OUTQ	*NONE		
Library name:				
Date of last password change:	PWDCHGDAT	08/09/86		
Text:	TEXT			
user profile for RPG III course				
10/13/86 13:55:29		USER PROFILE - GLCRPG		+++
AUTHORIZED COMMANDS				
CRTCLPGM				
CRTDSPF				
CRTDTAARA				
CRTLF				
CRTPF				
DSPPGMMNU				
ENTDBG				
OVRPRTF				
10/13/86 13:51:39		USER PROFILE - GLCRPG		+++
00013 AUTHORIZED OBJECTS				
			OBJ RIGHTS	DATA RIGHTS
OBJECT	LIBRARY	TYPE	OPER MGT EXIST	READ ADD UPD DLT
CRTCLPGM	QSYS	*CMD	X	X X X X
CRTDSPF	QSYS	*CMD	X	X X X X
CRTDTAARA	QSYS	*CMD	X	X X X X
CRTLF	QSYS	*CMD	X	X X X X
CRTPF	QSYS	*CMD	X	X X X X
DSPPGMMNU	QSYS	*CMD	X	X X X X
ENTDBG	QSYS	*CMD	X	X X X X
OVRPRTF	QSYS	*CMD	X	X X X X
GLCGRP	QSYS	*USRPRF		X
GLCRPG	QSYS	*USRPRF	X X	X X X X
GLCRPG	QSYS	*LIB	X	X X X X
RPGSRC	GLCRPG	*FILE	X X	X X X X
QPGMMENU	QSYS	*PGM	X	X X X X











ZR30-0870-04



Printed in U.S.A.