Docket No. R2001-1

USPS-LR-J-117

Development of Delivery Costs by Rate Category for First-Class and Standard Mail

Table of Contents

I. Introduction

This library reference provides the supporting documentation and analyses used to estimate delivery costs by rate category for First-Class Mail and Standard Mail. This is a Category 2 library reference sponsored by witness Schenk (USPS-T-43). This library reference updates the analysis of delivery costs by rate category in USPS-LR-I-95 in Docket No. R2000-1, which was sponsored by witness Daniel (USPS-T-28). In this update, the methodology used is the same as that used by witness Daniel, updated for base year and test year costs and volumes.

Other testimony and library references used to develop the cost estimates in this library reference include:

- USPS-T-12 for test year costs
- USPS-LR-J-53 for test year piggybacks and CRA costs
- USPS-LR-J-57 for CRA worksheets
- USPS-LR-J-112 for volumes by shape and weight increment
- USPS-LR-J-117 for city carrier costs

Witness Miller (USPS-LR-J-60) uses the results of this library reference in developing test year costs for letters. The results from this library reference are also used in developing cost by weight estimates in USPS-LR-J-58.

II. Organization

The final calculations used to develop delivery costs by rate category for First-Class and Standard Mail are presented in spreadsheet 'Table 1' in the Excel workbook LR-J-117.xls. Table 1, which is presented below, is analogous to Table 5 in USPS-LR-I-95/R2000-1. Table 1 contains the major results of this analysis that are referenced by other Postal Service witnesses in this docket. The methodology used in these calculations is the same as that used in USPS-LR-I-95/R2000-1, as described in witness Daniel's testimony (USPS-T-28) in that docket. The only changes made for this library reference are that base year and test year costs and volumes are updated. Data sources are referenced in each spreadsheet in LR-J-117.xls. The underlying City Carrier In-Office cost data is estimated in a similar manner to that described in Section III in USPS LR-I-100/R2000-1. The various programs used are described in Appendix A: Program Documentation. Appendix B contains the Fortran and Sort/Merge program listings. Electronic copies of all Excel workbooks and text files with the Fortran and Sort/Merge programs are provided on the accompanying CD-ROM.

Table 1: Test Year Unit Delivery Costs (Revised 11/20/01)

First-Class Single Piece	Test Year Unit Costs
Single-Piece Letters	6.037
Single-Piece Flats	8.902
Single-Piece Parcels	18.202
Single-Piece Nonletters	9.759

First-Class Presort			
Nonautomation -- Nonmach Mixed ADC	8.408		
Nonautomation -- Nonmach ADC	8.408		
Nonautomation -- Mach Mixed AADC	4.083		
Nonautomation -- Mach AADC	4.083		
Nonautomation -- Nonmach 3-Digit	8.408		
Nonautomation -- Nonmach 5-Digit	8.408		
Nonautomation -- Mach 3-Digit	3.954		
Nonautomation -- Mach 5-Digit	3.954		
Automation Mixed AADC	4.164		
Automation AADC	4.015		
Auto 3-Digit Letters	3.979		
Auto 5-Digit Letters CSBCS/Manual Sites	6.160		
Auto 5-Digit Letters Other Sites	2.893		
Auto 5-Digit Letters All	3.794		
Auto CR Letters	6.059		
		Presort Letters (Avg)	4.169
Presort Flats	9.217		
Presort Parcels	40.874		
Presort Nonletters	9.641		

First-Class Cards			
Single Piece Cards	6.886		
NonAuto Presort Cards	2.674		
Auto Mixed AADC Cards	2.789		
Auto AADC Cards	2.692		
Auto 3-Digit Cards	2.668		
Auto 5-Digit Cards CSBCS/Manual Sites	4.088		
Auto 5-Digit Cards Other Sites	1.962		
Auto 5-Digit Cards All	2.548		
Auto CR Cards	4.022	Presort Cards (Avg)	2.742

Standard Regular	
Nonautomation -- Nonmach Mixed ADC	5.592
Nonautomation -- Nonmach ADC	5.592
Nonautomation -- Mach Mixed AADC	3.854
Nonautomation -- Mach AADC	3.854
Nonautomation -- Nonmach 3-Digit	5.592
Nonautomation -- Nonmach 5-Digit	5.592
Nonautomation -- Mach 3-Digit	3.793
Nonautomation -- Mach 5-Digit	3.793
Automation Mixed AADC	3.887
Automation AADC	3.827
Automation 3-Digit Letters	3.812
Automation 5-Digit Letters	3.738
Regular Flat Subtotal	8.312
Regular Parcel Subtotal	22.974
Regular Nonletter Subtotal	8.945

Standard ECR			
ECR Basic Auto Letters	4.596		
ECR Basic Letters	6.384	ECR Basic Flats	6.033
ECR High Density Letters	4.684	ECR High Density Flats	4.747
ECR Saturation Letters	3.374	ECR Saturation Flats	3.960

Table 1: Test Year Unit Delivery Costs

First-Class Single Piece	Test Year Unit Costs
Single-Piece Letters	6.037
Single-Piece Flats	8.902
Single-Piece Parcels	18.202
Single-Piece Nonletters	9.759

First-Class Presort			
Nonautomation -- Nonmach Mixed ADC	8.408		
Nonautomation -- Nonmach ADC	8.408		
Nonautomation -- Mach Mixed AADC	4.066		
Nonautomation -- Mach AADC	4.066		
Nonautomation -- Nonmach 3-Digit	8.408		
Nonautomation -- Nonmach 5-Digit	8.408		
Nonautomation -- Mach 3-Digit	3.937		
Nonautomation -- Mach 5-Digit	3.937		
Automation Mixed AADC	4.165		
Automation AADC	4.016		
Auto 3-Digit Letters	3.980		
Auto 5-Digit Letters CSBCS/Manual Sites	6.161		
Auto 5-Digit Letters Other Sites	2.894		
Auto 5-Digit Letters All	3.795		
Auto CR Letters	6.060		
		Presort Letters (Avg)	4.169
Presort Flats	9.217		
Presort Parcels	40.874		
Presort Nonletters	9.641		

First-Class Cards			
Single Piece Cards	6.886		
NonAuto Presort Cards	2.674		
Auto Mixed AADC Cards	2.789		
Auto AADC Cards	2.692		
Auto 3-Digit Cards	2.668		
Auto 5-Digit Cards CSBCS/Manual Sites	4.088		
Auto 5-Digit Cards Other Sites	1.962		
Auto 5-Digit Cards All	2.548		
Auto CR Cards	4.022	Presort Cards (Avg)	2.742

Standard Regular	
Nonautomation -- Nonmach Mixed ADC	5.592
Nonautomation -- Nonmach ADC	5.592
Nonautomation -- Mach Mixed AADC	3.847
Nonautomation -- Mach AADC	3.847
Nonautomation -- Nonmach 3-Digit	5.592
Nonautomation -- Nonmach 5-Digit	5.592
Nonautomation -- Mach 3-Digit	3.795
Nonautomation -- Mach 5-Digit	3.795
Automation Mixed AADC	3.887
Automation AADC	3.827
Automation 3-Digit Letters	3.812
Automation 5-Digit Letters	3.738
Regular Flat Subtotal	8.312
Regular Parcel Subtotal	22.974
Regular Nonletter Subtotal	8.945

Standard ECR			
ECR Basic Auto Letters	4.596		
ECR Basic Letters	6.384	ECR Basic Flats	6.033
ECR High Density Letters	4.684	ECR High Density Flats	4.747
ECR Saturation Letters	3.374	ECR Saturation Flats	3.960

Appendix A: Program Documentation

A. Computer Hardware and Software

The IOCS data processing is performed on a Data General AViiON minicomputer with four Pentium Pro microprocessors and one gigabyte of RAM, running the DGUX version of UNIX operating system. Source programs ending with a ".f" are FORTRAN programs and programs ending in a ".sm" are SORT/MERGE programs. The remaining processing is performed on PCs running the Windows NT 4 and Windows 2000 operating systems and Microsoft Office.

B. Preparation of IOCS Data

The following programs are used to extract, code and process the 2000 IOCS data in preparation for LIOCATT distribution of city carrier in-office costs (CRA Cost Segment 6.1)

Program: **encode_wgt.f** - Extracts the necessary data from the IOCS tally data set, encodes specific tally fields into indexes, and writes the indexes to be used by LIOCATT

Input: **FY00 IOCS tally data** (USPS LR-J-10)
iocs2000.h – Declaration of IOCS tally fields
fincag.98 – Tally finance number and CAG combinations
activity00.ecr.all – List of activity codes

Output: **encdata** – Encoded IOCS tallies

Program: **encdata.sm** - Sorts the encoded IOCS data for the LIOCATT process

Input: **encdata**

Output: **encdata.s**

C. LIOCATT Based Cost Distribution Process

The LIOCATT distribution process is run through a main FORTRAN program, which uses subroutines to distribute mixed-mail costs, which are also FORTRAN programs. The declaration file 'liocatt.h' is included in each program and subroutine. This file contains common variables used by all programs and subroutines.

Program: **liocatt.f** - This program controls the LIOCATT process by running various FORTRAN programs, which replicates the LIOCATT process for mixed-mail cost distribution

Subroutines: **fillmixmap.f** – Produces a map for distributing the mixed mail codes to appropriate direct activity codes

Input: **activity00.ecr.all** – List of activity codes
mmcodes.intl – List of mixed-mail activity codes
mxmail.all.ecr – Maps class specific mixed-mail activity codes to corresponding direct activity codes

loaddata.f - Loads the encoded IOCS data

Input: **encdata.s** – Encoded IOCS data

fungroup.f - Forms function groups for operations

Input: **operrtemap** – Maps operation to function group

sortcost.f - Sort records for level 1 cost distribution

level1.f - Level 1 distribution of mixed-mail/not-handling tallies

level2.f - Level 2 distribution of mixed-mail/not-handling tallies

sortlev2a.f – Sort records for level 3 cost distribution

level3.f - Level 3 distribution of mixed-mail/not-handling tallies

report.f - Write results to file

Output: **level1b** – Level 1 distributed direct costs
level2b – Level 2 distributed direct costs
level3a – Level 3 direct costs
level3b – Level 3 distributed direct costs

D. Standard Mail ECR Rate Categories

Program: **rpt_city_ecr.f** - Summarizes LIOCATT cost distribution estimations by subclass and shape for city carrier in-office costs, Standard Mail ECR costs only

Input: **activity00.ecr.all** - List of activity codes and corresponding subclass codes
classes_ecr.old – List of CRA subclasses
level1b – Level 1 distributed direct costs
level2b - Level 2 distributed direct costs
level3a – Level 3 direct costs
level3b – Level 3 distributed direct costs

Output: **car_ecr_00.txt** - Estimated city carrier in-office Standard ECR costs by rate category and shape

Program: **rpt_cra_rt.f** – Summarizes LIOCATT cost distribution estimations by activity code (First-Class and Standard) and route code (IOCS field F260). Used to replicate LIOCATT system summary schedule K&L (USPS-T-1, WP-A.1)

Input: **activity00.ecr.all** - List of activity codes and corresponding subclass codes
level1b – Level 1 distributed direct costs
level2b - Level 2 distributed direct costs
level3a – Level 3 direct costs
level3b – Level 3 distributed direct costs

Output: **car00cra_rt.csv** – Estimated city carrier in-office costs by activity code and route code

E. Final Summary Spreadsheets

Workbook: **CC Costs ECR BY00.xls** – Summary tables reporting the LIOCATT estimations for city carrier in-office Standard ECR costs. Adjusts the FORTRAN replication of LIOCATT to match the BY00 CRA Cost Segment 6.1 costs

Input: **car_ecr_00.txt** - Estimated city carrier in-office Standard ECR costs by rate category and shape
BY00 CRA Cost Segment 6.1 Costs – CRA City Carrier In-Office costs from CS06&7.xls, worksheet 'oldoutputs' (USPS LR-J-57)
BY00 RPW Volumes – Volumes by subclass, shape, and weight increment from the Microsoft Excel file 'Volumes by Wgt GFY00 update.xls' (USPS LR-J-58)

Workbook: **CC Costs by Route BY00.xls** - Summary tables reporting the LIOCATT estimations for city carrier in-office First Class and Standard ECR costs by activity code and route code. Adjusts the FORTRAN replication of LIOCATT to match the BY00 CRA Cost Segment 6.1 costs. Used to replicate LIOCATT system summary schedule K&L (USPS-T-1, WP-A.1)

Input: **car00cra_rt.csv** – Estimated by city carrier in-office costs by activity code and route code
BY00 CRA Cost Segment 6.1 Costs – CRA City Carrier In-Office costs from I_Forms.xls, worksheet 'CS06.0.2.2' (USPS LR-J-57)

Workbook: **LR-J-117.xls** – City carrier in-office costs for First Class and Standard ECR by activity code and route code are presented in Table 1 of this workbook. Table 1 is analogous to Table 5 in witness Daniel's testimony USPS-T-28 in Docket No. R2000-1.

Input: **CC Costs ECR BY00.xls** for base year city carrier costs
CC Costs by Route BY00.xls for base year city carrier costs by route
USPS-LR-J-112 for volumes
USPS-T-11 for base year costs, city delivery carrier costs, and rural carrier costs
USPS-T-24 for the distribution of DPS volumes by rate element
USPS-T-12 for they FY01 wage rate
USPS-LR-J-98 for FY00 Billing Determinants

Appendix B: Program Lists

Section I: Preparation of IOCS Data

(Program: encode_wgt.f, encdata.sm)

```
      PROGRAM encode_wgt
C
C     Purpose: Encode tallies with indexes of arrays instead of
C              actual data. Delete leave tallies. Split old group 26
C              into new sub groups, change nonmod groups

C     Modified:  For weight increment analysis for R2001-1
c                Also performs the ECR breakout for delivery study

      IMPLICIT NONE

      integer*4   numcf, numact, numor, nw

      parameter   (numcf = 11)  ! number of cag - finnance number combs.
      parameter   (numor = 17)  ! number of operation/route codes
      parameter   (numact = 501) ! number of activity codes
      parameter   (nw = 22)     ! number of weight increments

      include 'iocs2000.h'

      integer*4   desigind      ! function to assign pay category
      integer*4   orind         ! function to assign operation/route code
      integer*4   bfind         ! function to assign basic function index
      integer*4   weight        ! function to assign weight increment
      integer*4   i2, i3, i4, i5, i6, i8, i7, i9
      integer*4   i, searchc, z, totall, ier, countlv
      integer*4   counto, countg, countsp, countk, countf
      integer*4   if166, if167, ct_nowgt

      real*8    tvalue, cost_clk, cost_mp

      character*7    cagfins(numcf), fincag
      character*4    acodes(numact), activity

      countlv = 0
      counto = 0
      countg = 0
      countsp = 0
      countk = 0
      countf = 0
      totall = 0
      ier = 0
      i2 = 0
      i3 = 0
      i4 = 0
      i5 = 0
      i6 = 0
      i7 = 0
      i8 = 0
      i9 = 0

c     Map of Fin CAG combinations (used for strata)
      open(14,file='fincag.98',iostat=ier)
      if (ier.ne.0) then
         print *,' error opening cagfin.s = ',ier
         stop
      end if

15    format(a7)
      do i = 1,numcf
         read(14,15) cagfins(i)
      end do
      print *,'finished reading cags  '

c     Map of activity codes
      open(16,file='activity00.ecr.all',iostat=ier )
      if (ier.ne.0) then
         print *,' error opening activity code files = ',ier
         stop
      end if

17    format(a4)
      do i = 1,numact
         read(16,17) acodes(i)
      end do
      close (14)
      close (16)
      print *,'finished reading activity codes  '

      open(25,file='iocsdata.2000.new',recl=1200) !  FY00 IOCS Data Set (based on full data set)
```

```
      open(30,file='encdata')
21    format(a)
31    format(i2,i1,i1,i3,i2,i3,i2,f11.2)

      ier = 0
      z=0
      cost_clk = 0.0
      cost_mp = 0.0
      ct_nowgt = 0

      do while (ier.eq.0)
         read(25,21,iostat=ier,end=100) rec

         z=z+1
         fincag = f263//f264     !  Strata

         i2 = searchc(cagfins,numcf,fincag) !  Find fincag (strata)
         i3 = desigind(f257)     ! craft
         i4 = orind(f260)        ! operation or route code
         if (i3.eq.1) i4 = 1     ! supervisors have no route or oper code
         i5 = bfind(f261)        ! basic function
         if (i3.eq.1) i5 = 1     ! supervisors have bf set to 0

         activity = f9806        ! activity code

         if (activity(1:1).ge.'1'.and.activity(1:1).le.'4'.and. ! map x091 to x080
     &      activity(2:4).eq.'091') then
            activity(2:4) = '080'
         end if

         if (i3.eq.2) then
            if (i4.eq.13) i4 = 11 !  Dump Op 88 into other for clk/mh
         end if

c     ECR activity code assignment
         if (activity(2:4).eq.'060') then !  1st Single Piece
            if (f136.eq.'D') then !  Metered
               if (activity.eq.'1060') activity = '1068'
               if (activity.eq.'2060') activity = '2068'
               if (activity.eq.'3060') activity = '3068'
               if (activity.eq.'4060') activity = '4068'
            end if
         end if
         if (activity(2:4).eq.'310') then !  Std A ECR
            if (f9618.eq.'1') then !  WSH
               if (activity.eq.'1310') activity = '1311'
               if (activity.eq.'2310') activity = '2311'
               if (activity.eq.'3310') activity = '3311'
               if (activity.eq.'4310') activity = '4311'
            else if (f9619.eq.'1') then !  WSS
               if (activity.eq.'1310') activity = '1313'
               if (activity.eq.'2310') activity = '2313'
               if (activity.eq.'3310') activity = '3313'
               if (activity.eq.'4310') activity = '4313'
            else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
               if (activity.eq.'1310') activity = '1312'
               if (activity.eq.'2310') activity = '2310'
               if (activity.eq.'3310') activity = '3310'
               if (activity.eq.'4310') activity = '4310'
            else                    !  ECRLOT
               activity = activity
            end if
         end if
         if (activity(2:4).eq.'330') then !  Std A Nonprofit ECR
            if (f9618.eq.'1') then !  WSH
               if (activity.eq.'1330') activity = '1331'
               if (activity.eq.'2330') activity = '2331'
               if (activity.eq.'3330') activity = '3331'
               if (activity.eq.'4330') activity = '4331'
            else if (f9619.eq.'1') then !  WSS
               if (activity.eq.'1330') activity = '1333'
               if (activity.eq.'2330') activity = '2333'
               if (activity.eq.'3330') activity = '3333'
               if (activity.eq.'4330') activity = '4333'
            else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
               if (activity.eq.'1330') activity = '1332'
               if (activity.eq.'2330') activity = '2330'
               if (activity.eq.'3330') activity = '3330'
               if (activity.eq.'4330') activity = '4330'
            else                    !  ECRLOT
```

```
            activity = activity
         end if
      end if

      i6 = searchc(acodes,numact,activity)

      if (i6.eq.0) then
         print*, 'Activity code not found ', activity
      end if

      i9 = 1

      read(f166,'(i2)') if166
      read(f167,'(i2)') if167

c     Weight increment assignment
      if (((activity.ge.'1000').and.(activity.le.'4950')).or.
     &    ((activity.ge.'5300').and.(activity.le.'5480'))) then
         i7 = weight(f165,if166,if167,ct_nowgt,nw) ! weight increments for direct actv codes only
      else
         i7 = nw
      end if

      read(f9250,'(f10.2)') tvalue !  F9250

      if ((i2.gt.0).and.(i3.gt.0).and.
     +    (i4.gt.0).and.(i5.gt.0).and.(i6.gt.0).and.(i7.gt.0)) then
         write (30,31) i2, i3, i5, i6, i9, i7, i4, tvalue
         countg = countg + 1
         if (i3.eq.2) then   ! clk/mh
            cost_clk = cost_clk + tvalue
            if ((i4.le.12).and.(i4.ne.10).and.(i4.ne.11)) then
               cost_mp = cost_mp + tvalue
            end if
         end if
      else
         if (f9806(1:1).eq.'9') then ! first digit of activity code f262
            countlv = countlv + 1
         else if (f257(2:2).eq.'4') then ! second digit of da code f257
            countsp = countsp + 1
         else if (f264.eq.'K') then ! cag k tally  f264
            countk = countk + 1
         else if (i2.eq.0) then
            print *,' invalid cagfin = ', fincag, ' fin = ', f2,
     &           ' roster ', f257, ' op code ', f260
            countf = countf + 1
         else
            print *,'    indexes = ',i2,i3,i4,i5,i6,i8
            print*, ' f260 = ', f260, ' f261 = ', f261
            counto = counto + 1
         end if
       end if
     end do
100  print *,' read exit code = ',ier
     print *,' number of records written to encdata = ',countg
     print *,' number of leave records excluded    = ',countlv
     print *,' number of spec. delivery excluded   = ',countsp
     print *,' number of CAG K records excluded    = ',countk
     print *,' number of invalid finance numbers   = ',countf
     print *,' number of other records excluded    = ',counto

     print*, 'Total valid clk/mh costs = ', cost_clk
     print*, 'Total valid clk/mh mail proc costs = ', cost_mp

     end


C ------------------------------------------------------------------------------
C       desigind
C
C       assigns index of 1-6 based on roster designation

        function    desigind(char)

        integer*4       desigind
        character*2     char

        if ((char.eq.' 9').or.
     +      (char.eq.'09').or.
     +      (char.eq.'19')) then
```

```
            desigind = 1
         else if (char.eq.'11') then
            desigind = 2
         else if ((char.eq.'31').or.
     +           (char.eq.'41').or.
     +           (char.eq.'61').or.
     &           (char.eq.'81')) then
            desigind = 2
         else if ((char.eq.'12').or.
     +           (char.eq.'32').or.
     +           (char.eq.'42').or.
     +           (char.eq.'62').or.
     &           (char.eq.'82')) then
            desigind = 2
         else if (char.eq.'13') then
            desigind = 3
         else if ((char.eq.'33').or.
     +           (char.eq.'43').or.
     +           (char.eq.'63').or.
     &           (char.eq.'83')) then
            desigind = 3
         else
            desigind = -1
         end if

         return
         end
C ------------------------------------------------------------------------------
C    bfind
C
C    returns index for basic function

         function    bfind(char)

         integer*4      bfind
         character*1    char

         if (char.eq.'1') then
            bfind = 1
         else if (char.eq.'2') then
            bfind = 2
         else if (char.eq.'3') then
            bfind = 3
         else if (char.eq.'5') then
            bfind = 4
         else
            bfind = -1
         end if

         return
         end
C ------------------------------------------------------------------------------
C     orind

c     returns index value for operation or route code

         function    orind(char)

         integer*4      orind
         character*2    char

         if (char.eq.'00') then
            orind = 1
         else if (char.eq.'01') then
            orind = 2
         else if (char.eq.'02') then
            orind = 3
         else if (char.eq.'03') then
            orind = 4
         else if (char.eq.'04') then
            orind = 5
         else if ((char.eq.'05').or.
     +           ((char.ge.'11').and.(char.le.'13')).or.
     +           (char.eq.'15').or.
     +           (char.eq.'16').or.
     +           ((char.ge.'19').and.(char.le.'21')).or.
     +           ((char.ge.'27').and.(char.le.'29'))) then
            orind = 6
         else if ((char.eq.'06').or.
     +           (char.eq.'18').or.
```

```
     +           (char.eq.'22').or.
     +           (char.eq.'23')) then
            orind = 7
         else if (char.eq.'07') then
            orind = 8
         else if (char.eq.'08') then
            orind = 9
         else if ((char.eq.'09').or.
     +           (char.eq.'24').or.
     +           (char.eq.'25').or.
     +           (char.eq.'26')) then
            orind = 10
         else if ((char.eq.'10').or.
     +           (char.eq.'17')) then
            orind = 11
         else if (char.eq.'14') then
            orind = 12
         else if (char.eq.'71') then
            orind = 1
         else if (char.eq.'73') then
            orind = 2
         else if (char.eq.'75') then
            orind = 3
         else if (char.eq.'77') then
            orind = 4
         else if (char.eq.'78') then
            orind = 5
         else if (char.eq.'80') then
            orind = 6
         else if (char.eq.'82') then
            orind = 7
         else if (char.eq.'83') then
            orind = 8
         else if (char.eq.'84') then
            orind = 9
         else if (char.eq.'85') then
            orind = 10
         else if (char.eq.'86') then
            orind = 11
         else if (char.eq.'87') then
            orind = 12
         else if (char.eq.'88') then
            orind = 13
         else if (char.eq.'89') then
            orind = 14
         else if (char.eq.'90') then
            orind = 15
         else if (char.eq.'98') then
            orind = 16
         else if (char.eq.'99') then
            orind = 17
         else
            orind = -1
         end if

         return
         end

c ------------------------------------------------------------------------------------------------
c     weight

c     Assigns weight increment

      function weight(f165,if166,if167,ct_nowgt,nw)

      character*1  f165
      integer*4    if166, if167, weight, ct_nowgt, nw

      weight = 0

      if (f165.eq.'A') then
         weight = 1             !  < 1/2 ounce
      else if (f165.eq.'B') then
         weight = 2             !  1 ounces
      else if (f165.eq.'C') then
         weight = 3             !  1 1/2 ounces
      else if (f165.eq.'D') then
         weight = 4             !  2 ounces
      else if (f165.eq.'E') then
         weight = 5             !  2 1/2 ounces
```

```
else if (f165.eq.'F') then
   weight = 6              !  3 ounces
else if (f165.eq.'G') then
   weight = 7              !  3 1/2 ounces
else if (f165.eq.'H') then
   weight = 8              !  4 ounces
else if (f165.eq.'I') then
   if (if166.eq.0) then    !  < 1 lb
      if (if167.gt.0) then
         weight = if167 + 4
      else
         weight = nw
         ct_nowgt = ct_nowgt + 1
      end if
   else if ((if166.eq.1).and.(if167.eq.0)) then
      weight = 20
   else if ((if166.gt.1).or.((if166.eq.1).and.(if167.gt.0))) then
      weight = 21
   else
      weight = nw
      ct_nowgt = ct_nowgt + 1
   end if
else
   weight = nw
   ct_nowgt = ct_nowgt + 1
end if

return
end
```

```
% Name: Encdata.sm
% Sort encoded IOCS tally info
input file is 'encdata', recs are data sensitive upto 30 chars.
output file is 'encdata.s', recs are data sensitive  upto 30 chars.
sort.
end.
```

Section II: LIOCATT Cost Distribution Process

(Programs: liocatt.f, fillmixmap.f, loaddata.f, fungroup.f, sortcost.f, level1.f, level2.f, sortlev2a.f, level3.f, report.f, rpt_city_ecr.f, rpt_cra_rt.f)

```
      PROGRAM liocatt

C     PURPOSE: Allocate costs to raw tallies, and performs LIOCATT mixed mail cost distribution

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   nlev1a, nlev1b, nfun
      integer*4   nlev2a, nlev2b, nlev3a, nlev3b

      character*3 runtype

      logical     dofun

      call  getarg(1,runtype)
      if (runtype.eq.'fun') then
         dofun = .true.
      else
         dofun = .false.
      end if

      call  fillmixmap          ! load mixed mail distribution map and activity codes
      call  loaddata            ! load encdata.s (from encode_wgt.f, encdata.sm)
      call  fungroup(nfun)      ! form function groups from operations
      call  sortcost(nfun)      ! sort records for level1
      call  level1(nfun,nlev1a,nlev1b) ! level 1 indirect cost allocation
      call  level2(nlev1a,nlev2a,nlev2b) ! level 2 indirect cost allocation
      call  sortlev2a(nlev2a)   ! sort records for level 3
      call  level3(nlev2a,nlev3a,nlev3b) ! level 3 indirect cost allocation
      call  report(nlev1b,nlev2b,nlev3a,nlev3b) ! write results to file
      print *,' ************************************************** '
      print *,' Fiscal year 2000 completed '
      print *,' Total number of records = ', nrec
      print *,' Number of records after function creation = ',nfun
      print *,' Number of level 1 records = ',nlev1a,', ',nlev1b
      print *,' Number of level 2 records = ',nlev2a,', ',nlev2b
      print *,' Number of level 3 records = ',nlev3a,', ',nlev3b
      print *,' ************************************************** '

      end

C -------------------------------------------------------------------------------
```

```
      subroutine fillmixmap

C     PURPOSE: Read the mixed mail codes and produce map for distributing
C              mixed mail codes to appropriate direct activity codes

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   i, j, ind
      integer*4   ier
      integer*4   searchc

      character*4 mmcodes(nummix)
      character*4 codes(20)

      logical  flag

      if (debug) print *,' Enter subroutine fillmixmap '

      ier = 0

c     Map of activity codes
      open(16,file='activity00.ecr.all',iostat=ier )
17    format(a4)
      do i = 1,numact
         read(16,17) acodes(i)
      end do

C       initialize count array (number of direct keys indirect code is distributed accross)
      do i =1 , nummix
         count(i) = 0
      end do

c     Map of mixed mail activity codes
      open(18,file='mmcodes.intl')
19    format(a4)
      do i = 1,nummix
         read(18,19) mmcodes(i)
      end do

c     Maps mixed mail codes to direct activity codes
      open(20,file='mxmail.all.ecr')
21    format(20a4)

      do while (ier.eq.0)
         read (20,21,iostat=ier,end=100) codes
         i = searchc(mmcodes,nummix,codes(1))
         if (i.gt.0) then
            flag = .true.
            ind = 1
            do while ((flag).and.(ind.lt.20))
               ind = ind + 1
               if (codes(ind).ne.'   0') then
                  j = searchc(acodes,numact,codes(ind))
                  if (j.gt.0) then
                     count(i) = count(i) + 1
                     mixmap(count(i),i) = j
                  else
                     print *,' Direct mail code did not map ',codes(ind)
                  end if
               else
                  flag = .false.
               end if
            end do
         else
            print *,' Mixed mail code did not map ',codes(1)
         end if
      end do
100   print *,' read exit code = ',ier

C     Fill mix_to_act array

      do i = 1,nummix
         mix_to_act(i) = searchc(acodes,numact,mmcodes(i))
      end do

      if (debug) print *,' exiting fillmixmap '

      close (16)
```

```
close (18)
close (20)

return
end
```

```
      subroutine loaddata

c     Purpose:  To read in coded data set - encode_wgt.f, encdata.sm

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   j
      integer*2   ioff, iact, ibf, iw, ifun, ipig, iocc
      integer*4   ier/0/
      character*14   datum, ldatum
      real*8         cost, lcost, tcost


      open (20,file='encdata.s') !  Sorted data set
21    format(i2,i1,i1,i3,i2,i3,i2,f11.2)

      lcost = 0.0
      tcost = 0.0
      ldatum = ' '
      j = 0
      nrec = 0
      do while (ier.eq.0)
         read (20,21,iostat=ier,end=100) ioff,iocc,ibf,iact,ipig,iw,ifun, cost
         write (datum,'(7a2)') ioff,iocc,ibf,iact,iw,ipig,ifun
         if ((datum.ne.ldatum).and.(j.ne.0)) then
            nrec = nrec + 1
            if (nrec.le.maxcost) then
               write (costbuf2(nrec),'(a14,a8)') ldatum, tcost
            else
               print *,' maxcost exceeded in loaddata '
               stop
            end if
            ldatum = datum
            tcost = cost
         else
            if (j.eq.0) then
               j = 1
               ldatum = datum
            end if
            tcost = tcost + cost
         end if
      end do
100   if (debug) print *,' Read exit of encdata = ',ier,', nrec = ',nrec
      close (20)
      return
      end
```

```
      subroutine fungroup(numout)

C     PURPOSE: Add clerk/mail handler records up into functional groups from operation codes.

      IMPLICIT NONE

      include  'liocatt.h'

      real*8       original_costs(numopr)
      real*8       fun_costs(numfun)
      real*8       cost

      integer*2    opr_to_fun(numfun,numopr)
      integer*2    cofg, cpay, copr, cbf, cact, cw, cpig
      integer*2    ofg, pay, opr, fun, bf, act, w, pig
      integer*4    i, j, numout
      integer*4    ier/0/
      integer*4    indb1, indb2

      logical      same

      if (debug) print *,' entering fungroup.f77 '

C     Fill opr_to_fun array

      open(20,file='operrtemap')
21    format(45i3)
      do i = 1,numfun
         read (20,21) (opr_to_fun(i,j),j=1,numopr)
      end do

C     open input and output file

C     Collect data for a office, pay category cell.
C     1.  Collapse over quarter
C     2.  If pay category is clerk/mailhandler create functional groups
C     3.  Output for regular processing in level1 - level3

31    format(7a2,a8)

C     Read first record of first group

      same = .true.
      indb1 = 1
      indb2 = 0
      do opr = 1, numopr
         original_costs(opr) = 0.0
      end do

      read (costbuf2(indb1),31) cofg, cpay, cbf, cact, cw, cpig, copr, cost
      original_costs(copr) = original_costs(copr) + cost

      do while (ier.eq.0)

C     Read rest of cost group
         do while (same)
            indb1 = indb1 + 1
            if (indb1.gt.nrec) then
               ier = -1
               goto 100
            end if
            read (costbuf2(indb1),31) ofg, pay, bf, act, w, pig, opr, cost
            if ((ofg.eq.cofg).and.(pay.eq.cpay).and.(bf.eq.cbf).and.
     +          (act.eq.cact).and.(w.eq.cw).and.(pig.eq.cpig)) then
               original_costs(opr) = original_costs(opr) + cost
            else
               copr = opr
               same = .false.
            end if
         end do
100      if (ier.ne.0) print *,' Read exit code = ',ier

C     Sum operation codes into function groups if clerk/mailhandler

         if (cpay.eq.2) then
            do fun = 1,numfun
               fun_costs(fun) = 0.
               do i = 2,opr_to_fun(fun,1)
                  fun_costs(fun) = fun_costs(fun) +
     +                 original_costs(opr_to_fun(fun,i))
```

```
            end do
         end do
      end if

C     Output data from original costs if supervisor or carrier,
C     and from fun costs if clerk/mailhandler

      if ((cpay.eq.1).or.(cpay.eq.3)) then
         do opr = 1, numopr
            if (original_costs(opr).gt.0) then
               indb2 = indb2 + 1
               if (indb2.gt.maxcost) then
                  print *,' maxcost exceeded on write to costbuf1 in fungroup '
                  stop
               end if
               write (costbuf1(indb2),31) cofg, cpay, opr, cbf, cact, cw, cpig,
     +            original_costs(opr)
            end if
         end do
      else if (cpay.eq.2) then
         do fun = 1,numfun
            if (fun_costs(fun).gt.0) then
               indb2 = indb2 + 1
               if (indb2.gt.maxcost) then
                  print *,' maxcost exceeded on write to costbuf1 in fungroup '
                  stop
               end if
               write (costbuf1(indb2),31) cofg, cpay, fun, cbf, cact, cw, cpig,
     +            fun_costs(fun)
            end if
         end do
      end if

C     Set up next group from last record read

      same = .true.
      do opr = 1, numopr
         original_costs(opr) = 0.0
      end do
      cofg = ofg
      cpay = pay
      cbf = bf
      cact = act
      cw = w
      cpig = pig
      original_costs(copr) = original_costs(copr) + cost

      end do

      if (debug) print *,' number of records written to costbuf1 = ',indb2
      numout = indb2

      return
      end

C -------------------------------------------------------------------------------
```

```
      subroutine sortcost(n)

c     Purpose:  Sorts records for level 1 cost distribution

      implicit none

      include 'liocatt.h'

      integer*4   i, j, l, n, ir
      character*22   rra

      if (debug) print *,' entering sortcost '

      l=n/2+1
      ir=n
10    continue

      if(l.gt.1)then
         l=l-1
         rra=costbuf1(l)
      else
         rra=costbuf1(ir)
         costbuf1(ir)=costbuf1(1)
         ir=ir-1
         if(ir.eq.1)then
            costbuf1(1)=rra
            if (debug) print *,' sortcost finished '
            return
         end if
      end if
      i=l
      j=l+l
20    if (j.le.ir) then
         if (j.lt.ir) then
            if (costbuf1(j)(1:14).lt.costbuf1(j+1)(1:14)) j=j+1
         end if
         if (rra(1:14).lt.costbuf1(j)(1:14)) then
            costbuf1(i)=costbuf1(j)
            i=j
            j=j+j
         else
            j=ir+1
         end if
         goto 20
      end if
      costbuf1(i)=rra
      goto 10

      end
```

```
      subroutine level1(numin,numouta,numoutb)

C     PURPOSE: Perform level one distribution of mixed mail costs to
C              direct mail codes.

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   maxgrp

      parameter   (maxgrp = 20000)

      integer*4   size1
      parameter   (size1 = numact*nummix)

      integer*2   group(4,maxgrp)
      integer*4   actptr(2,numact,numbf)
      integer*4   bfptr(2,numbf)

      real*8      original_costs(maxgrp)
      real*8      dist_mix_costs(npig,maxgrp)
      real*8      sum, cost, mixsum, chkmix

      integer*4   numin, numouta, numoutb
      integer*4   indin, inda, indb, indx
      integer*2   cofg, cpay, copr, cbf, cact, cw, cpig
      integer*2   ofg, pay, opr, bf, act, mixkey, w, ind, pig
      integer*4   i, j, k
      integer*4   ier/0/

      logical     same

      logical   debug1/.true./

      if (debug) print *,' Entering Level1.f77 '

31    format(7a2,a8)

C       Perform level one allocation
C
C       1. Collect matrix of costs for a office group, pay and cost group
C          category cell
C       2. Within each basic function cell :
C          a. For each mixed mail activity sum over direct mail costs it is
C             to be distributed to.
C          b. If sum is positive use shares to distribute mixed mail costs to
C             to direct mail costs.
C          c. If sum is zero add mixed costs to same cell for basic function 4
C          d. Output records for this bf cell.
C              1) all direct costs and all bf 4 costs to "a" file
C              2) all distributed direct cost to "b" file (bfs 1-3)

C       Set up matrices for first record

      do bf = 1, numbf          ! initialize actptr array
         do act = 1, numact
            actptr(1,act,bf) = 0
         end do
      end do
      do bf = 1, numbf
         bfptr(1,bf) = 0
      end do
      indin = 1
      inda = 0
      indb = 0

      same = .true.
      ind = 1

C       Read first record of first cost group

      read (costbuf1(indin),31) cofg, cpay, copr, cbf, cact, cw, cpig, cost

      original_costs(ind) = cost
      group(1,ind) = cbf
      group(2,ind) = cact
      group(3,ind) = cw
      group(4,ind) = cpig
      actptr(1,cact,cbf) = ind
```

```
      actptr(2,cact,cbf) = 1
      bfptr(1,cbf) = ind
      bfptr(2,cbf) = 1
      ier = 0

      do while (ier.eq.0)

C          Read rest of cost group

         do while (same)
            indin = indin + 1
            if (indin.gt.numin) then
               ier = -1
               goto 100
            end if
            read (costbuf1(indin),31) ofg, pay, opr, bf, act, w, pig, cost
            if ((ofg.eq.cofg).and.(pay.eq.cpay).and.(opr.eq.copr)) then
               ind = ind + 1
               if (debug) then
                  if (ind.gt.maxgrp) then
                     print *,' maxgrp exeeeded , ofg = ',ofg,' pay = ',pay,' opr = ',opr
                     ier = -999
                     goto 100
                  end if
               end if
               original_costs(ind) = cost
               group(1,ind) = bf
               group(2,ind) = act
               group(3,ind) = w
               group(4,ind) = pig
               if (actptr(1,act,bf).eq.0) then
                  actptr(1,act,bf) = ind
                  actptr(2,act,bf) = 1
               else
                  actptr(2,act,bf) = actptr(2,act,bf) + 1
               end if
               if (bfptr(1,bf).eq.0) then
                  bfptr(1,bf) = ind
                  bfptr(2,bf) = 1
               else
                  bfptr(2,bf) = bfptr(2,bf) + 1
               end if
            else
               same = .false.
               cbf = bf
               cact = act
               cw = w
               cpig = pig
            end if
         end do
100      if ((ier.ne.0).and.(debug))  print *,' Read exit code = ',ier
         do i = 1, ind
            do j = 1, npig
               dist_mix_costs(j,i) = 0.0
            end do
         end do
         if (debug1) then
            print *,' bfptr(1,1) = ',bfptr(1,1)
            print *,' bfptr(1,2) = ',bfptr(1,2)
            print *,' bfptr(1,3) = ',bfptr(1,3)
            print *,' bfptr(1,4) = ',bfptr(1,4)
         end if

C          Attempt to distribute mixed dollars into direct costs

         do bf = 1,3            ! do not attempt to distribute other at this level
            if (bfptr(1,bf).ne.0) then
               do i = 1,nummix  ! loop over mixed mail activities
                  if (actptr(1,mix_to_act(i),bf).gt.0) then
                     do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
                        sum = 0
                        mixsum = original_costs(indx)
                        pig = group(4,indx)
                        do j = 1,count(i) ! sum over direct keys for mixed code
                           mixkey = mixmap(j,i)
                           if (actptr(1,mixkey,bf).ne.0) then
                              do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                                 sum = sum + original_costs(k)
                              end do
                           end if
```

```
                  end do
                  chkmix = 0
                  if (sum.gt.0) then ! distribute to direct codes
                     do j = 1,count(i)
                        mixkey = mixmap(j,i)
                        if (actptr(1,mixkey,bf).ne.0) then
                           do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                              dist_mix_costs(pig,k) = dist_mix_costs(pig,k) +
     +                           mixsum*(original_costs(k)/sum)
                              if (debug1) chkmix = chkmix +
     +                           mixsum*(original_costs(k)/sum)
                           end do
                        end if
                     end do
                     original_costs(indx) = 0.0
                     if (dabs(mixsum-chkmix).gt.1.0)
     &                  print *,' allocation failure, mixsum = ',mixsum,', chkmix = ',chkmix
                  end if
               end do
            end if
         end do
         do k = bfptr(1,bf), (bfptr(1,bf)+bfptr(2,bf)-1) ! Output records for this opr,bf  cell
            if (original_costs(k).gt.0.0) then
               inda = inda + 1
               write (costbuf2(inda),31) cofg, cpay, copr,
     +            group(1,k), group(2,k), group(3,k), group(4,k), original_costs(k)
            end if
            do pig = 1, npig
               if (dist_mix_costs(pig,k).gt.0.0) then
                  indb = indb + 1
                  if (indb.le.max11b) then
                     write (level1b(indb),31) cofg, cpay, copr, group(1,k),
     •                  group(2,k), group(3,k), pig, dist_mix_costs(pig,k)
                  else
                     print *,' max11b exceeded, inda = ', inda
                     stop
                  end if
               end if
            end do
         end do
      end if
   end do
   if (bfptr(1,4).gt.0) then
      do k = bfptr(1,4), ind ! Output all records for basic function "other"
         inda = inda + 1
         write (costbuf2(inda),31) cofg, cpay, copr, group(1,k),
     +      group(2,k), group(3,k), group(4,k), original_costs(k)
      end do
   end if

C     Set up next cost group using last record read

   if (ind.gt.(numbf*numact)) then
      do bf = 1, numbf    ! initialize actptr array
         do act = 1, numact
            actptr(1,act,bf) = 0
         end do
      end do
   else
      do k = 1, ind
         bf = group(1,k)
         act = group(2,k)
         actptr(1,act,bf) = 0
      end do
   end if
   do bf = 1, numbf
      bfptr(1,bf) = 0
   end do

   same = .true.
   ind = 1
   cofg = ofg
   copr = opr
   cpay = pay
   cpig = pig

   original_costs(ind) = cost
   group(1,ind) = cbf
   group(2,ind) = cact
   group(3,ind) = cw
```

```
            actptr(1,cact,cbf) = ind
            actptr(2,cact,cbf) = 1
            bfptr(1,cbf) = ind
            bfptr(2,cbf) = 1

         end do

         numouta = inda
         numoutb = indb

         if (debug) print *,' number of records written to costbuf2 = ',numouta
         if (debug) print *,' number of records written to level1b  = ',numoutb
         return
         end

C ------------------------------------------------------------------------------
```

```
      subroutine level2(numin,numouta,numoutb)

C     PURPOSE: Perform level two distribution of mixed mail costs to
C              direct mail codes.

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   maxgrp

      parameter   (maxgrp = 20000)

      integer*4   sizel
      parameter   (sizel = numact*nummix)

      integer*2   group(4,maxgrp)
      integer*4   actptr(2,numact,numbf)
      integer*4   bfptr(2,numbf)

      real*8      original_costs(maxgrp)
      real*8      dist_mix_costs(npig,maxgrp)
      real*8      sum, cost, mixsum, chkmix
      real*8      mixpig(npig)
      real*8      tdirect/0/  , tmixed/0/ , tmixeddist/0/ , tratio/0/

      integer*4   numin, numouta, numoutb
      integer*4   indin, inda, indb, indx
      integer*2   cofg, cpay, copr, cbf, cact, cw, cpig
      integer*2   ofg, pay, opr, bf, act, mixkey, w, ind, pig
      integer*2   bf4/4/
      integer*4   i, j, k
      integer*4   ier/0/

      logical     same , dist
      logical   debug1/.true./

      if (debug) print *,' Entering Level2.f77 '

C       open input and output file

31    format(7a2,a8)
45    format(7a2,a8)

C       Perform level two mixed cost allocation
C
C       1. Collect matrix of costs for a office group,pay category, and
C          operation/route code cell
C       2. Over all records in cell
C          a. For each mixed mail activity sum over basic functions to get
C             mixed mail costs.
C          c. For each mixed mail activity sum over direct mail costs it is
C             to be distributed to (over all basic functions).
C          b. If sum is positive use shares to distribute mixed mail costs to
C             to direct mail costs (all distributed mixed costs get basic
C             function 4.
C          d. Output records for this opr cell.
C               1) all direct costs costs to "a" file
C               2) all distributed direct cost to "b" file (bf 4)

C       Set up matrices for first record

      do bf = 1, numbf          ! initialize actptr array
         do act = 1, numact
            actptr(1,act,bf) = 0
         end do
      end do
      do bf = 1, numbf
         bfptr(1,bf) = 0
      end do
      indin = 1
      inda = 0
      indb = 0

      same = .true.
      ind = 1

C       Read first record of first cost group

      read (costbuf2(indin),31) cofg, cpay, copr, cbf, cact, cw, cpig, cost
```

```
      original_costs(ind) = cost
      group(1,ind) = cbf
      group(2,ind) = cact
      group(3,ind) = cw
      group(4,ind) = cpig
      actptr(1,cact,cbf) = ind
      actptr(2,cact,cbf) = 1
      bfptr(1,cbf) = ind
      bfptr(2,cbf) = 1
      ier = 0

      do while (ier.eq.0)

C          Read rest of cost group

         do while (same)
            indin = indin + 1
            if (indin.gt.numin) then
               ier = -1
               goto 100
            end if
            read (costbuf2(indin),31) ofg, pay, opr, bf, act, w, pig, cost
            if ((ofg.eq.cofg).and.(pay.eq.cpay).and.(opr.eq.copr)) then
               ind = ind + 1
               if (debug) then
                  if (ind.gt.maxgrp) then
                     print *,' maxgrp exceeded , ofg = ',ofg,' pay = ',pay,' opr = ',opr
                     ier = -999
                     goto 100
                  end if
               end if
               original_costs(ind) = cost
               group(1,ind) = bf
               group(2,ind) = act
               group(3,ind) = w
               group(4,ind) = pig
               if (actptr(1,act,bf).eq.0) then
                  actptr(1,act,bf) = ind
                  actptr(2,act,bf) = 1
               else
                  actptr(2,act,bf) = actptr(2,act,bf) + 1
               end if
               if (bfptr(1,bf).eq.0) then
                  bfptr(1,bf) = ind
                  bfptr(2,bf) = 1
               else
                  bfptr(2,bf) = bfptr(2,bf) + 1
               end if
            else
               same = .false.
               cbf = bf
               cact = act
               cw = w
               cpig = pig
            end if
         end do
100      if ((ier.ne.0).and.(debug)) print *,' Read exit code = ',ier
         do i = 1, ind
            do j = 1, npig
               dist_mix_costs(j,i) = 0.0
            end do
         end do
         if (debug1) then
            print *,' bfptr(1,1) = ',bfptr(1,1)
            print *,' bfptr(1,2) = ',bfptr(1,2)
            print *,' bfptr(1,3) = ',bfptr(1,3)
            print *,' bfptr(1,4) = ',bfptr(1,4)
         end if

C          Attempt to distribute mixed dollars into direct costs

         do i = 1,nummix        ! loop over mixed mail activities
            do pig = 1, npig
               mixpig(pig) = 0.0
            end do
            do bf = 1, numbf
               if (actptr(1,mix_to_act(i),bf).gt.0) then
                  do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
                     pig = group(4,indx)
```

```
                  mixpig(pig) = mixpig(pig) + original_costs(indx)
               end do
            end if
         end do
         dist = .false.
         do pig = 1, npig
            if (mixpig(pig).gt.0.0) then
               sum = 0.0
               do bf = 1, numbf
                  do j = 1,count(i) ! sum over direct keys for mixed code
                     mixkey = mixmap(j,i)
                     if (actptr(1,mixkey,bf).ne.0) then
                        do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                           sum = sum + original_costs(k)
                        end do
                     end if
                  end do
               end do
               chkmix = 0
               if (sum.gt.0) then ! distribute to direct codes
                  do bf = 1, numbf
                     do j = 1,count(i)
                        mixkey = mixmap(j,i)
                        if (actptr(1,mixkey,bf).ne.0) then
                           do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                              dist_mix_costs(pig,k) = dist_mix_costs(pig,k) +
     +                           mixpig(pig)*(original_costs(k)/sum)
                              if (debug1) chkmix = chkmix +
     +                           mixpig(pig)*(original_costs(k)/sum)
                           end do
                        end if
                     end do
                  end do
                  dist = .true.
                  if (dabs(mixpig(pig)-chkmix).gt.1.0)
     &                print *,' level2 allocation failure, mixsum = ',mixsum,', chkmix = ',chkmix
               end if
            end if
         end do
         if (dist) then
            do bf = 1, numbf
               if (actptr(1,mix_to_act(i),bf).gt.0) then
                  do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
                     original_costs(indx) = 0.0
                  end do
               end if
            end do
         end if
      end do
      do k = 1, ind
         if (original_costs(k).gt.0.0) then
            inda = inda + 1
            write (costbuf1(inda),45) cofg, cpay, copr, group(1,k),
     +         group(2,k), group(3,k), group(4,k), original_costs(k)
         end if
         do pig = 1, npig
            if (dist_mix_costs(pig,k).gt.0.0) then
               indb = indb + 1
               if (indb.le.maxl2b) then
                  write (level2b(indb),45) cofg, cpay, copr, bf4, group(2,k),
     +               group(3,k), pig, dist_mix_costs(pig,k)
               else
                  print *,' maxl2b exceeded, inda = ',inda
                  stop ' fatal error '
               end if
            end if
         end do
      end do

C     Set up next cost group using last record read

      if (ind.gt.(numbf*numact)) then
         do bf = 1, numbf    ! initialize actptr array
            do act = 1, numact
               actptr(1,act,bf) = 0
            end do
         end do
      else
         do k = 1, ind
            bf = group(1,k)
```

```
            act = group(2,k)
            actptr(1,act,bf) = 0
         end do
      end if
      do bf = 1, numbf
         bfptr(1,bf) = 0
      end do

      same = .true.
      ind = 1
      cofg = ofg
      copr = opr
      cpay = pay

      original_costs(ind) = cost
      group(1,ind) = cbf
      group(2,ind) = cact
      group(3,ind) = cw
      group(4,ind) = cpig
      actptr(1,cact,cbf) = ind
      actptr(2,cact,cbf) = 1
      bfptr(1,cbf) = ind
      bfptr(2,cbf) = 1

   end do

   numouta = inda
   numoutb = indb

   if (debug) print *,' number of records written to costbuf1 = ',numouta
   if (debug) print *,' number of records written to level2b  = ',numoutb
   return
   end

C ------------------------------------------------------------------------------
```

```
      subroutine sortlev2a(n)

C     Purpose:  To sort records for level 3 cost distribution

      implicit none

      include 'liocatt.h'

      integer*4   i, j, l, n, ir
      character*22   rra

      if (debug) print *,' entering sortlev2a.f77 '

      l=n/2+1
      ir=n
10    continue

      if(l.gt.1)then
         l=l-1
         rra=costbuf1(l)
      else
         rra=costbuf1(ir)
         costbuf1(ir)=costbuf1(1)
         ir=ir-1
         if(ir.eq.1)then
            costbuf1(1)=rra
            if (debug) print *,' exiting sortlev2a '
            return
         end if
      end if
      i=l
      j=l+l
20    if (j.le.ir) then
         if (j.lt.ir) then
            if (costbuf1(j)(3:12).lt.costbuf1(j+1)(3:12)) j=j+1
         end if
         if (rra(3:12).lt.costbuf1(j)(3:12)) then
            costbuf1(i)=costbuf1(j)
            i=j
            j=j+j
         else
            j=ir+1
         end if
         goto 20
      end if
      costbuf1(i)=rra
      goto 10
      end
```

```
      subroutine level3(numin,numouta,numoutb)

C     PURPOSE: Perform level three distribution of mixed mail costs to direct mail codes.

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   maxgrp

      parameter   (maxgrp = 200000)

      integer*4   size1
      parameter   (size1 = numact*nummix)

      integer*2   group(4,maxgrp)
      integer*4   actptr(2,numact,numbf)
      integer*4   bfptr(2,numbf)

      real*8      original_costs(maxgrp)
      real*8      dist_mix_costs(npig,maxgrp)
      real*8      sum, cost, mixsum, chkmix
      real*8      mixed(npig)

      integer*4   numin, numouta, numoutb
      integer*4   indin, inda, indb, indx
      integer*2   cpay, copr, cbf, cact, cw, cpig
      integer*2   pay, opr, bf, act, mixkey, w, ind, pig
      integer*2   bf4/4/
      integer*4   i, j, k
      integer*4   ier/0/

      logical     same , dist
      logical   debug1/.true./

      if (debug) print *,' entering level3.f77 '

C       open input and output file

31    format(2x,6a2,a8)
45    format(6a2,a8)

C       Perform level three mixed cost allocation
C
C       1. Collect matrix of costs for a pay category,operation/route code cell
C       2. Over all records in cell
C          a. For each mixed mail activity sum over basic functions to get
C             mixed mail costs.
C          c. For each mixed mail activity sum over direct mail costs it is
C             to be distributed to (over all basic functions).
C          b. If sum is positive use shares to distribute mixed mail costs to
C             to direct mail costs (all distributed mixed costs get basic
C             function 4.
C          d. Output records for this opr cell.
C              1) all direct costs costs to "a" file
C              2) all distributed direct cost to "b" file (bf 4)

C       Set up matrices for first record

      do bf = 1, numbf          ! initialize actptr array
         do act = 1, numact
            actptr(1,act,bf) = 0
         end do
      end do
      do bf = 1, numbf
         bfptr(1,bf) = 0
      end do
      indin = 1
      inda = 0
      indb = 0

      same = .true.
      ind = 1

C       Read first record of first cost group

      read (costbuf1(indin),31) cpay, copr, cbf, cact, cw, cpig, cost

      original_costs(ind) = cost
      group(1,ind) = cbf
```

```
      group(2,ind) = cact
      group(3,ind) = cw
      group(4,ind) = cpig
      actptr(1,cact,cbf) = ind
      actptr(2,cact,cbf) = 1
      bfptr(1,cbf) = ind
      bfptr(2,cbf) = 1
      ier = 0

      do while (ier.eq.0)

C          Read rest of cost group

         do while (same)
            indin = indin + 1
            if (indin.gt.numin) then
               ier = -1
               goto 100
            end if
            read (costbuf1(indin),31) pay, opr, bf, act, w, pig, cost
            if ((pay.eq.cpay).and.(opr.eq.copr)) then
               if ((bf.eq.group(1,ind)).and.
     +             (act.eq.group(2,ind)).and.
     +             (w.eq.group(3,ind)).and.
     +             (pig.eq.group(4,ind))) then
                  original_costs(ind) = original_costs(ind) + cost
               else
                  ind = ind + 1
                  if (debug) then
                     if (ind.gt.maxgrp)
     +                  print *,' maxgroup exceeded, pay = ',pay,', opr = ',opr
                  end if
                  original_costs(ind) = cost
                  group(1,ind) = bf
                  group(2,ind) = act
                  group(3,ind) = w
                  group(4,ind) = pig
                  if (actptr(1,act,bf).eq.0) then
                     actptr(1,act,bf) = ind
                     actptr(2,act,bf) = 1
                  else
                     actptr(2,act,bf) = actptr(2,act,bf) + 1
                  end if
                  if (bfptr(1,bf).eq.0) then
                     bfptr(1,bf) = ind
                     bfptr(2,bf) = 1
                  else
                     bfptr(2,bf) = bfptr(2,bf) + 1
                  end if
               end if
            else
               same = .false.
               cbf = bf
               cact = act
               cw = w
               cpig = pig
            end if
         end do
100      if ((ier.ne.0).and.debug) print *,' Read exit code = ',ier
         do i = 1, ind
            do pig = 1, npig
               dist_mix_costs(pig,i) = 0.0
            end do
         end do
         if (debug1) then
            print *,' bfptr(1,1) = ',bfptr(1,1)
            print *,' bfptr(1,2) = ',bfptr(1,2)
            print *,' bfptr(1,3) = ',bfptr(1,3)
            print *,' bfptr(1,4) = ',bfptr(1,4)
         end if

C          Attempt to distribute mixed dollars into direct costs

         do i = 1,nummix        ! loop over mixed mail activities
            do pig = 1, npig
               mixed(pig) = 0.0
            end do
            do bf = 1, numbf
               if (actptr(1,mix_to_act(i),bf).gt.0) then
                  do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
```

```
                  pig = group(4,indx)
                  mixed(pig) = mixed(pig) + original_costs(indx)
               end do
            end if
         end do
         dist = .false.
         do pig = 1, npig
            if (mixed(pig).gt.0.0) then
               sum = 0.0
               do bf = 1, numbf
                  do j = 1,count(i) ! sum over direct keys for mixed code
                     mixkey = mixmap(j,i)
                     if (actptr(1,mixkey,bf).ne.0) then
                        do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                           sum = sum + original_costs(k)
                        end do
                     end if
                  end do
               end do
               chkmix = 0
               if (sum.gt.0) then ! distribute to direct codes
                  do bf = 1, numbf
                     do j = 1,count(i)
                        mixkey = mixmap(j,i)
                        if (actptr(1,mixkey,bf).ne.0) then
                           do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                              dist_mix_costs(pig,k) = dist_mix_costs(pig,k) +
     +                           mixed(pig)*(original_costs(k)/sum)
                              if (debug1) chkmix = chkmix +
     +                           mixed(pig)*(original_costs(k)/sum)
                           end do
                        end if
                     end do
                     if (actptr(1,mix_to_act(i),bf).ne.0)
     +                  original_costs(actptr(1,mix_to_act(i),bf)) = 0.0
                  end do
                  dist = .true.
               end if
               if (dabs(mixed(pig)-chkmix).gt.1.0)
     &               print *,' level 3 allocation failure, mixsum = ',mixsum,', chkmix = ',chkmix
            end if
         end do
         if (dist) then
            do bf = 1, numbf
               if (actptr(1,mix_to_act(i),bf).gt.0) then
                  do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
                     original_costs(indx) = 0.0
                  end do
               end if
            end do
         end if
      end do
      do k = 1, ind
         if (original_costs(k).gt.0.0) then
            inda = inda + 1
            write (costbuf2(inda),45) cpay, copr, group(1,k),
     +           group(2,k), group(3,k), group(4,k), original_costs(k)
         end if
         do pig = 1, npig
            if (dist_mix_costs(pig,k).gt.0.0) then
               indb = indb + 1
               if (indb.le.maxl3b) then
                  write (level3b(indb),45) cpay, copr, bf4, group(2,k),
     +                 group(3,k), pig, dist_mix_costs(pig,k)
               else
                  print *,' maxl3b exceeded inda = ',inda
                  stop
               end if
            end if
         end do
      end do

C     Set up next cost group using last record read

      if (ind.gt.(numbf*numact)) then
         do bf = 1, numbf    ! initialize actptr array
            do act = 1, numact
               actptr(1,act,bf) = 0
            end do
         end do
```

```
         else
            do k = 1, ind
               bf = group(1,k)
               act = group(2,k)
               actptr(1,act,bf) = 0
            end do
         end if
         do bf = 1, numbf
            bfptr(1,bf) = 0
         end do

         same = .true.
         ind = 1
         cpay = pay
         copr = opr

         original_costs(ind) = cost
         group(1,ind) = cbf
         group(2,ind) = cact
         group(3,ind) = cw
         group(4,ind) = cpig
         actptr(1,cact,cbf) = ind
         actptr(2,cact,cbf) = 1
         bfptr(1,cbf) = ind
         bfptr(2,cbf) = 1

      end do

      numouta = inda
      numoutb = indb

      if (debug) print *,' number of records written to level3a = ',numouta
      if (debug) print *,' number of records written to level3b = ',numoutb
      return
      end

C -----------------------------------------------------------------------------
```

```
      subroutine report(nlev1b,nlev2b,nlev3a,nlev3b)

C     PURPOSE: Produce report on results of Liocatt by activity for city carriers by
C              activity code and operation/route code

      IMPLICIT NONE

      include 'liocatt.h'

      real*8       data(numw,numact)
      real*8       indata

      integer*4    nlev1b, nlev2b, nlev3a, nlev3b, irun
      integer*2    ofg, opr, act, pay, bf, w, pig
      integer*4    i, j
      integer*4    ier/0/

      character*3 buffer

      logical flag/.false./

      if (debug) then
         print *,' in subroutine report '
         print *, ' nlev1b = ',nlev1b,' nlev2b = ',nlev2b
         print *, ' nlev3a = ',nlev3a,' nlev3b = ',nlev3b
      end if
      do i = 1, numact
         do j = 1, numw
            data(j,i) = 0.0
         end do
      end do

C       Open input files

25    format(7a2,a8)
35    format(6a2,a8)

      open(20,file='level1b')
      open(21,file='level2b')
      open(22,file='level3a')
      open(23,file='level3b')
45    format(i2.2,i1,i2.2,i1,i3.3,i3.3,i2,f13.1)
46    format(i1,i2.2,i1,i3.3,i3.3,i2,f13.1)

C       Assemble data for report

      do i = 1, nlev1b
         read (level1b(i),25) ofg,pay,opr,bf,act,w,pig,indata
         write (20,45) ofg,pay,opr,bf,act,w,pig,indata
      end do
      do i = 1, nlev2b
         read (level2b(i),25) ofg,pay,opr,bf,act,w,pig,indata
         write (21,45) ofg,pay,opr,bf,act,w,pig,indata
      end do
      do i = 1, nlev3a
         read (costbuf2(i),35) pay,opr,bf,act,w,pig,indata
         write (22,46) pay,opr,bf,act,w,pig,indata
      end do
      do i = 1, nlev3b
         read (level3b(i),35) pay,opr,bf,act,w,pig,indata
         write (23,46) pay,opr,bf,act,w,pig,indata
      end do

      return
      end

C -------------------------------------------------------------------------------
```

```
      PROGRAM rpt_city_ecr

C     PURPOSE: Summarizes city carrier in-office costs for Standard (A) ECR by ECR category and shape

      IMPLICIT NONE

      integer*4   numact, w, nopr, nshp, ncl

      parameter   (numact = 501) ! number of activity codes
      parameter   (nopr = 12)    !  number of operations
      parameter   (ncl = 243)    !  number of subclasses
      parameter   (nshp = 4)     !  number of shapes

      integer*4   is, shape, ishp, shp(numact), icl

      real*8      carrier(ncl,nshp)
      real*8      indata

      integer*4   pay, opr, bf, act
      integer*4   unit, i, ier
      integer*4   class(numact)

      character*4 acodes(numact)
      character*9 classes(ncl)
      character*5 shapetype(nshp)

      ier = 0

c     Map of activity codes and codes to corresponding subclass
      open(16,file='activity00.ecr.all')
17    format(a4,i6)

      do i = 1,numact
         read(16,17) acodes(i), class(i)
         is = shape(acodes(i))
         shp(i) = is
      end do
      close(16)

c     Map of subclasses
      open(16,file='classes_ecr.old')
18    format(a9)
      do i = 1, ncl
         read(16,18) classes(i)
      end do

      close(16)

c     Initialize matrices
      do icl = 1, ncl
         do ishp = 1, nshp
            carrier(icl,ishp) = 0.0
         end do
      end do

c     Open files of LIOCATT results
      open(20,file='level1b')
      open(21,file='level2b')
25    format(2x,i1,i2.2,i1,2i3.3,2x,f13.1)
      open(30,file='level3a')
      open(31,file='level3b')
35    format(i1,i2.2,i1,2i3.3,2x,f13.1)

      do unit = 20,21
         do while (ier.eq.0)
            read (unit,25,iostat=ier,end=100) pay,opr,bf,act,w,indata
            icl = class(act)     !  Assign subclass
            ishp = shp(act)      !  Assign shape
            if (pay.eq.3) then   !  City Carriers only
               if (icl.gt.0) then
                  if (ishp.gt.0) then
                     carrier(icl,ishp) = carrier(icl,ishp) + indata/1000.
                  else
                     print*, 'Invalid shape ', acodes(act), shp(act)
                  end if
               else
                  print*, 'Invalid class assignment ', acodes(act), ' ', class(act)
               end if
            end if
         end do
```

```
100      print *,' Read exit of unit ',unit,' = ',ier
         ier = 0
      end do

      do unit = 30,31
         do while (ier.eq.0)
            read (unit,35,iostat=ier,end=101) pay,opr,bf,act,w,indata
            icl = class(act)     !  Assign subclass
            ishp = shp(act)      !  Assign shape
            if (pay.eq.3) then   !  City Carriers only
               if (icl.gt.0) then
                  if (ishp.gt.0) then
                     carrier(icl,ishp) = carrier(icl,ishp) + indata/1000.
                  else
                     print*, 'Invalid shape ', acodes(act), shp(act)
                  end if
               else
                  print*, 'Invalid class assignment ', acodes(act), ' ', class(act)
               end if
            end if
         end do
101      print *,' Read exit of unit ',unit,' = ',ier
         ier = 0
      end do

      shapetype(1) = '1Ltr '
      shapetype(2) = '2Flt '
      shapetype(3) = '3Pcl '
      shapetype(4) = '4None'

c     Write out results
      open(45,file='car_ecr_00.txt')
41    format(a9,1x,i3,1x,a5,1x,f15.5)
      do icl = 1, ncl
         do ishp = 1, nshp
            if (((icl.ge.18).and.(icl.le.21)).or.((icl.ge.24).and.(icl.le.27))) then !  Std ECR only
               write (45,41) classes(icl), icl, shapetype(ishp), carrier(icl,ishp)
            end if
         end do
      end do

      end

c-------------------------------------------------------------------------------
c     Assign shape

      function shape(act)

      integer*4    shape
      character*4  act

      if (act(1:1).eq.'1') then
         shape = 1   !  Letters
      else if (act(1:1).eq.'2') then
         shape = 2   !  Flats
      else if ((act(1:1).ge.'3').and.(act(1:1).le.'4')) then
         shape = 3   !  IPPs/Parcels
      else
         shape = 4   !  None
      end if

      return
      end
```

```
      PROGRAM rpt_cra_rt

C     PURPOSE: Summarizes city carrier in-office costs for Standard (A) ECR by ECR category
C              and route code category

      IMPLICIT NONE

      integer*4   numact, nopr, nroute

      parameter   (numact = 501) ! number of activity codes
      parameter   (nopr = 12)    !  number of operations
      parameter   (nroute = 17) !  number of route codes

      integer*4   ishp, icl
      integer*4   pay, opr, bf, act, w
      integer*4   unit, i, ier, iop
      integer*4   class(numact)

      real*8      carrier(numact,nroute)
      real*8      indata

      character*4 acodes(numact)
      character*2 opcode(nroute)/'71','73','75','77','78','80','82','83','84','85',
     &   '86','87','88','89','90','98','99'/

      ier = 0

c     Map of activity codes
      open(16,file='activity00.ecr.all')
17    format(a4,i6)

      do i = 1,numact
         read(16,17) acodes(i), class(i)
      end do
      close(16)

c     Initialize matrices
      do icl = 1, numact
         do ishp = 1, nroute
            carrier(icl,ishp) = 0.0
         end do
      end do

c     Open files of LIOCATT results
      open(20,file='level1b')
      open(21,file='level2b')
25    format(2x,i1,i2.2,i1,2i3.3,2x,f13.1)
      open(30,file='level3a')
      open(31,file='level3b')
35    format(i1,i2.2,i1,2i3.3,2x,f13.1)

      do unit = 20,21
         do while (ier.eq.0)
            read (unit,25,iostat=ier,end=100) pay,opr,bf,act,w,indata
            if (act.eq.22) act = 21 !  1SP
            if ((act.ge.36).and.(act.le.38)) act = 35 !  ECR
            if ((act.ge.40).and.(act.le.42)) act = 39 !  NPECR
            if (act.eq.100) act = 99 !  1SP
            if ((act.ge.112).and.(act.le.114)) act = 111 !  ECR
            if ((act.ge.116).and.(act.le.118)) act = 115 !  NPECR
            if (act.eq.182) act = 181 !  !SP
            if ((act.ge.192).and.(act.le.194)) act = 191 !  ECR
            if ((act.ge.196).and.(act.le.198)) act = 195 !  NPECR
            if (act.eq.256) act = 255 !  1SP
            if ((act.ge.266).and.(act.le.268)) act = 265 !  ECR
            if ((act.ge.270).and.(act.le.272)) act = 269 !  NPECR
            if ((act.ge.50).and.(act.le.95)) act = 83 !  Intl
            if ((act.ge.130).and.(act.le.177)) act = 164 !  Intl
            if ((act.ge.206).and.(act.le.251)) act = 238 !  Intl
            if ((act.ge.286).and.(act.le.337)) act = 321 !  Intl
            if (pay.eq.3) then  !  City Carriers
               if (act.gt.0) then
                  if (opr.gt.0) then
                     carrier(act,opr) = carrier(act,opr) + indata/1000.
                  else
                     print*, 'Route code ', opr
                  end if
               else
                  print*, 'Invalid class assignment ', acodes(act), class(act)
               end if
```

```
            end if
         end do
100      print *,' Read exit of unit ',unit,' = ',ier
         ier = 0
      end do

      ier = 0

      do unit = 30,31
         do while (ier.eq.0)
            read (unit,35,iostat=ier,end=101) pay,opr,bf,act,w,indata
            if (act.eq.22) act = 21 !  1SP
            if ((act.ge.36).and.(act.le.38)) act = 35 !  ECR
            if ((act.ge.40).and.(act.le.42)) act = 39 !  NPECR
            if (act.eq.100) act = 99 !  1SP
            if ((act.ge.112).and.(act.le.114)) act = 111 !  ECR
            if ((act.ge.116).and.(act.le.118)) act = 115 !  NPECR
            if (act.eq.182) act = 181 !  !SP
            if ((act.ge.192).and.(act.le.194)) act = 191 !  ECR
            if ((act.ge.196).and.(act.le.198)) act = 195 !  NPECR
            if (act.eq.256) act = 255 !  1SP
            if ((act.ge.266).and.(act.le.268)) act = 265 !  ECR
            if ((act.ge.270).and.(act.le.272)) act = 269 !  NPECR
            if ((act.ge.50).and.(act.le.95)) act = 83 !  Intl
            if ((act.ge.130).and.(act.le.177)) act = 164 !  Intl
            if ((act.ge.206).and.(act.le.251)) act = 238 !  Intl
            if ((act.ge.286).and.(act.le.337)) act = 321 !  Intl
            if (pay.eq.3) then  !  City Carriers
               if (act.gt.0) then
                  if (opr.gt.0) then
                     carrier(act,opr) = carrier(act,opr) + indata/1000.
                  else
                     print*, 'Route code ', opr
                  end if
               else
                  print*, 'Invalid class assignment ', acodes(act), class(act)
               end if
            end if
         end do
101      print *,' Read exit of unit ',unit,' = ',ier
         ier = 0
      end do

c     Write out results
      open(45,file='car00cra_rt.csv')
41    format(a4,17(',',f15.3))
      do icl = 1, numact
         if ((class(icl).eq.1).or.((class(icl).ge.3).and.(class(icl).le.5)).or.
     &       ((class(icl).ge.18).and.(class(icl).le.29))) then
            write (45,41) acodes(icl), (carrier(icl,iop), iop = 1, nroute)
         end if
      end do

      end
```