BEFORE THE
POSTAL RATE COMMISSION
WASHINGTON, D.C. 20268–0001

POSTAL RATE AND FEE CHANGES, 1997

Docket No. R97–1

NOTICE OF THE UNITED STATES POSTAL SERVICE CONCERNING FILING OF SUPPLEMENTAL TESTIMONY OF MICHAEL R. MCGRANE (USPS-ST-44)

On October 6, 1997, in its Response to the Newspaper Association of America Motion to Strike Testimony of United States Postal Service Witness Joseph D. Moeller, the Postal Service gave notice of its intent to file the testimony of a Postal Service witness to adopt Library References LR-H-109 and LR-H-182. Consistent with that notice, the Postal Service hereby files the supplemental testimony of Michael R. McGrane, which is being designated as USPS-ST-44. A copy of the testimony is attached to this notice.

The Postal Service also hereby gives notice that witness McGrane is adopting as his own interrogatory responses the following previously filed institutional interrogatory responses, filed on the dates indicated below:

AAPS/USPS-T36-7-11 (filed October 1, 1997)
ABA/USPS-1 (filed September 2, 1997)
ADVO/USPS-26-28 (filed October 1, 1997)
MOAA/USPS-T36-1 (filed October 1, 1997)
NAA/USPS-T36-17-27, 29-31 (filed September 4, 1997)
NAA/USPS-18-19 (filed October 6, 1997)

Respectfully submitted,

UNITED STATES POSTAL SERVICE

By its attorneys:

Daniel J. Foucheaux, Jr.
Chief Counsel, Ratemaking

Anthony F. Alverno

475 L'Enfant Plaza West, S.W.
Washington, D.C. 20260–1137
(202) 268–2997; Fax –5402
October 8, 1997

CERTIFICATE OF SERVICE

I hereby certify that I have this day served the foregoing document upon all participants of record in this proceeding in accordance with section 12 of the Rules of Practice.

Anthony F. Alverno

475 L'Enfant Plaza West, S.W.
Washington, D.C. 20260–1137
October 8, 1997

USPS-ST-44

BEFORE THE
POSTAL RATE COMMISSION
WASHINGTON, D.C. 20268–0001

POSTAL RATE AND FEE CHANGES, 1997	Docket No. R97–1

SUPPLEMENTAL TESTIMONY
OF
MICHAEL R. MCGRANE
ON BEHALF OF
UNITED STATES POSTAL SERVICE

Direct Testimony
of
Michael R. McGrane

AUTOBIOGRAPHICAL SKETCH

My name is Michael R. McGrane. I am a senior economist with Christensen Associates of Madison, Wisconsin. I have been employed by Christensen Associates for eleven years. I previously provided testimony before the Postal Rate Commission on Periodicals costs in Docket MC95-1. In addition, I have performed research and provided support for many other analyses presented in Docket Nos. R94-1, MC95-1, MC96-2, MC97-2, and R97-1. This work has included mail volume estimation using the PERMIT and BRAVIS bulk mail systems, cost estimation using the IOCS and other CRA databases, surveys of mail piece characteristics and makeup practices, field surveys of operational practices, and labor rate forecasting.

I received a B.S. in economics from the University of Wisconsin-Madison in May of 1987. I have also completed further courses in economics and computer science at the University of Wisconsin-Madison.

I. PURPOSE OF TESTIMONY

The purpose of this testimony is to adopt the results and analyses contained in Library Reference H-109, Standard Mail (A) Mail Processing ECR Costs, and Library Reference H-182, Standard Mail (A) Unit Volume Variable Cost by Weight Increment. These documents were filed as library references with the Postal Rate Commission on July 10, 1997. Library Reference H-109 is attached to my testimony and designated as Exhibit USPS-44A, and Library Reference H-182 is also attached to my testimony and designated as Exhibit USPS-44B. This testimony presents a brief summary of the methodology and the results for each of the two studies.

II. ECR MAIL PROCESSING COSTS

A. Methods

This study separates the mail processing costs for Standard Mail (A) Enhanced Carrier Route mail into costs for high-density/saturation mail and costs for other carrier-route mail. The separation is made on the basis of endorsements recorded on IOCS direct tallies, and was performed using FY96 data. Exhibit USPS-44A at pages 2 and 3 describes the methods used to identify these endorsements and produce cost estimates for the two categories. Table 1 of Exhibit USPS-44A shows the detailed calculations.

B. Results

The following tables summarizes the mail processing costs for walk-sequence endorsed and not walk-sequence endorsed mail for both commercial (identified as "Regular Rate" in Exhibit USPS-44A) and nonprofit Enhanced Carrier Route Standard Mail (A).

Mail Processing Costs for Commercial Enhanced Carrier Route Mail (thousands of dollars)

Shape	Not Walk-Sequence Endorsed	Walk-Sequence Endorsed
Letters	183,648	8,734
Non-Letters	193,206	22,488

Mail Processing Costs for Nonprofit Enhanced Carrier Route Mail
(thousands of dollars)

Shape	Not Walk-Sequence Endorsed	Walk-Sequence Endorsed
Letters	31,571	1,017
Non-Letters	11,340	940

III. STANDARD MAIL (A) UNIT VOLUME VARIABLE COST BY WEIGHT INCREMENT

A. Methods

This study estimates unit volume variable cost by weight increment for ECR and Regular Standard (A) Mail in FY96. Pages 2 and 3 of Exhibit USPS-44B briefly describe the methods used to make these estimates. Tables 3 though 6 of Exhibit USPS-44B contain the details of the construction of the cost estimates by weight increment for Standard (A) Commercial Enhanced Carrier Route, Standard (A) Regular, Standard (A) Nonprofit Enhanced Carrier Route, and Standard (A) Nonprofit mail, respectively.

B. Results

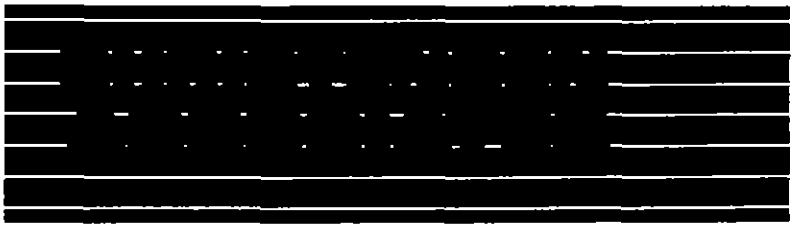
Table 1 of Exhibit USPS-44B summarizes these results for the sum of commercial and nonprofit subclasses and Table 2 summarizes the results for flat-shaped mail only. **(Tony - if we need to beef this section up we could include the following two paragraphs from the library reference)**

Table 1 shows the results for carrier route and other mail separately. These results are also presented graphically in Chart 1 of Exhibit USPS-44B. Unit costs for carrier route mail are in the range of 5 to 7 cents per piece for the first seven ounce increments. Mail weighing more than seven ounces trends upwards to a peak of 18 cents at sixteen ounces. Other bulk mail shows a slightly more sloped path, beginning at 11 cents at one ounce, trending upwards towards 20 cents in the twelve to fourteen ounce range, and sharply rising to 49 cents at sixteen ounces.

For comparison to the all shape tables, Table 2 and Chart 2 of USPS-44B contain the unit cost estimates for flat shaped mail only. The carrier route curve is similar to the curve for all shapes. The other mail curve has a high initial unit cost, then is similar to the all shapes curve in the middle ounce increments. In contrast to the all shapes curve, there is no sharp rise in cost in the heaviest weight increments.

Exhibit USPS-44A

Standard Mail (A) Mail Processing
ECR Costs



Economic Analysis and Consulting

LR-H-109

STANDARD MAIL (A) MAIL PROCESSING ECR COSTS U.S. POSTAL SERVICE

4610 University Avenue
Suite 700
Madison, Wisconsin 53705-2164

608 231 2266

LR-H-109 Standard Mail (A) Mail Processing ECR Costs

Table of Contents

Introduction

This study separates the mail processing costs for Standard Mail (A) enhanced carrier-route mail into costs for high density/saturation mail and costs for other carrier-route mail. This separation is made on the basis of endorsements recorded on IOCS direct tallies for enhanced carrier route mail. This document will refer to high density and saturation mail together as walk-sequenced mail.

The requirement to separately endorse walk-sequenced bulk third-class mail was published in the Postal Bulletin on September 1, 1994, with an effective date of December 10, 1994. The required endorsement is the marking "WS" before either the carrier route endorsement or the carrier route information line. The IOCS began collecting the presence of this marking at the beginning of FY 1995. The "WS" marking was used for both saturation and 125-piece walk-sequence mail.

With the advent of reclassification, the requirements for marking saturation and high-density mail were changed to "ECRWSS" and "ECRWSH" respectively. This change took place on July 1, 1996. The IOCS began collecting the presence of these markings at that same time.

Analysis

To classify Standard Mail (A) IOCS mail processing tallies as walk-sequence the following two rules were used. Any carrier-route tally with the "WS", "ECRWSS", or the "ECRWSH" bullet marked was placed into the walk-sequence group Also, any carrier route letter or flat-shaped tally with the detached address card bullet marked was placed into the walk-sequence group. Only saturation pieces may have detached address cards in Standard Mail (A). This approach is conservative in the sense that it assigns to walk-sequence costs which have the possibility of being caused by walk-sequence mail.

Once the walk-sequence status of each tally was established, the tallies were grouped by mail processing cost pool and activity code.[1] The sum of the tally dollar value for each combination of walk-sequence status, activity code, and cost pool was calculated These dollars were used to form the percentage distribution of cost by walk-sequence status for each cost pool and shape/subclass of ECR mail. These percentages were applied to the variable

mail processing cost including piggybacks for each pool to yield the distribution of mail processing cost by walk-sequence status.

These calculations can be seen in Table 1. There is a separate page in Table 1 for each shape and subclass of ECR mail. Columns 1 and 2 show the direct IOCS tally cost by walk-sequence status. Columns 4 and 5 show the percentage distribution of this cost. Column 6 contains the variable mail processing cost by cost pool from library reference LR-H-106. Columns 7 and 8 contain the distribution of the variable cost to walk-sequence status for cost pools with direct tally cost. Column 9 contains the variable cost for cost pools with no direct tallies.

The costs are summarized in Table 2. The amounts in columns 1 through 3 are taken from the total line of each subclass/shape page of Table 1. Columns 4 and 5 have the cost for cost pools with no direct tallies (column 3) distributed to walk-sequence in proportion to costs for pools with direct tally information.

[1] The following activity codes were used: 1310 (regular ECR letters, 2310 (regular ECR flats), 3310 & 4310 (regular ECR parcels), 1330 (nonprofit ECR letters), 2330 (nonprofit ECR flats), 3330 & 4330 (nonprofit ECR parcels).

Table 1

Development of Walk Sequence vs. Non-Walk Sequence Costs - Standard (A) Regular ECR Letters

			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
			Direct Tally IOCS Costs by Presence of Walk-Sequence Endorsement			Percent of Total		Variable Mail	Variable Costs Distributed to WS/Non-WS		
	Group	Pool	Non-WS	WS	Total	=(1)/(3) Non-WS	=(2)/(3) WS	Processing Costs	=(4)*(6) Non-WS	=(5)*(6) WS	No Key
1	mods	bcs/	4,854	127	4,981	0 975	0 025	16,985	16,553	432	
2	mods	express	-	-	-			2	No Key	No Key	2
3	mods	fsm/	63	-	63	1.000	-	169	169	-	
4	mods	lsm/	1,772	133	1,906	0 930	0.070	6,487	6,033	454	
5	mods	manf	129	-	129	1 000	-	249	249	-	
6	mods	manl	7,041	285	7,327	0.961	0 039	11,718	11,262	456	
7	mods	manp	104	-	104	1 000	-	201	201	-	
8	mods	mecparc	-	-	-			32	No Key	No Key	32
9	mods	ocr/	1,450	344	1,793	0.808	0.192	4,838	3,911	927	
10	mods	priority	50	-	50	1.000	-	132	132	-	
11	mods	spbs Oth	1,605	-	1,605	1 000	-	2,584	2,584	-	
12	mods	spbsPrio	171	-	171	1.000	-	574	574	-	
13	mods	BusReply	-	-	-			6	No Key	No Key	6
14	mods	INTL	63	-	63	1.000	-	138	138	-	
15	mods	LD15	208	-	208	1.000	-	6,261	6,281	-	
16	mods	LD41	52	108	160	0 326	0 674	468	153	315	
17	mods	LD42	-	-	-			3	No Key	No Key	3
18	mods	LD43	6,235	155	6,390	0 976	0.024	14,417	14,066	351	
19	mods	LD44	392	-	392	1 000	-	699	699	-	
20	mods	LD48 Exp	152	48	200	0.759	0 241	7	5	2	
21	mods	LD48 Oth	417	67	484	0 862	0 138	496	427	69	
22	mods	LD48_SSv	59	-	59	1 000	-	40	40	-	
23	mods	LD49	732	-	732	1 000	-	1,679	1,679	-	
24	mods	LD79	637	123	759	0 839	0 161	5,018	4,208	810	
25	mods	MAILGRAMS	-	-	-			-			
26	mods	Registry	-	-	-			1	No Key	No Key	1
27	mods	REWRAP	-	-	-			5	No Key	No Key	5
28	mods	1Bulk pr	-	-	-			33	No Key	No Key	33
29	mods	1CancMPP	392	61	453	0 866	0.134	1,245	1,078	167	
30	mods	1EEQMT	-	-	-			834	No Key	No Key	834
31	mods	1MISC	-	-	-			1,027	No Key	No Key	1,027
32	mods	1OPbulk	3,446	129	3,576	0 964	0 036	13,769	13,271	498	
33	mods	1OPpref	2,756	-	2,756	1 000	-	10,915	10,915	-	
34	mods	1Platfrm	1,644	67	1,711	0 961	0 039	18,280	17,565	715	
35	mods	1POUCHNG	949	72	1,021	0 929	0 071	4,449	4,133	316	
36	mods	1SackS_h	442	-	442	1 000	-	2,382	2,382	-	
37	mods	1SackS_m	181	-	181	1 000	-	3,241	3,241	-	
38	mods	1SCAN	66	-	66	1 000	-	659	659	-	
39	mods	1SUPPORT	63	-	63	1.000	-	1,210	1,210	-	
40	BMCs	nmo	48	-	48	1 000	-	13	13	-	
41	BMCs	psm	60	-	60	1.000	-	190	190	-	
42	BMCs	spb	289	-	289	1 000	-	978	978	-	
43	BMCs	ssm	742	-	742	1.000	-	3,956	3,956	-	
44	BMCs	Othr	1,216	104	1,320	0.921	0.079	4,578	4,216	362	
45	BMCs	Pla	896	54	950	0.943	0 057	4,708	4,440	269	
46	Non Mods		23,449	1,329	24,778	0 946	0 054	46,687	44,183	2,504	
	Total		62,826	3,207	66,033	0.951	0 049	192,382	181,792	8,646	1,944
									0 945	0.045	0.010
	Sources:		Appendix A	Appendix A	(1) + (2)	(1) / (3)	(2) / (3)	LR-H-106	(4) * (6)	(5) * (6)	If (3) = 0 then (6)

Table 1

Development of Walk Sequence vs. Non-Walk Sequence Costs - Standard (A) Regular ECR Non-Letters

			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
			Direct Tally IOCS Costs by Presence of Walk-Sequence Endorsement			Percent of Total		Variable Mail	Variable Costs Distributed to WS/Non-WS		
	Group	Pool	Non-WS	WS	Total	=(1)/(3) Non-WS	=(2)/(3) WS	Processing Costs	=(4)*(6) Non-WS	=(5)*(6) WS	No Key
1	mods	bcs/	72	-	72	1 000	-	281	281	-	
2	mods	express	-	-	-			12	No Key	No Key	12
3	mods	fsm/	5,162	327	5,490	0 940	0 060	13,443	12,641	802	
4	mods	lsm/	105	-	105	1 000	-	394	394	-	
5	mods	manf	4,445	145	4,589	0 968	0.032	9,302	9,009	293	
6	mods	manl	642	70	712	0 902	0.098	1,207	1,088	118	
7	mods	manp	493	45	539	0 916	0.084	632	579	53	
8	mods	mecparc	133	-	133	1 000	-	355	355	-	
9	mods	ocr/	-	-	-			14	No Key	No Key	14
10	mods	priority	133	-	133	1.000	-	261	261	-	
11	mods	spbs Oth	4,652	202	4,854	0.958	0 042	8,096	7,759	337	
12	mods	spbsPrio	205	-	205	1.000	-	656	656	-	
13	mods	BusReply	-	-	-			10	No Key	No Key	10
14	mods	INTL	-	-	-			54	No Key	No Key	54
15	mods	LD15	-	-	-			-			
16	mods	LD41	-	-	-			-			
17	mods	LD42	-	80	80	-	1 000	54	-	54	
18	mods	LD43	9,398	1,810	11,208	0.839	0 161	27,255	22,854	4,401	
19	mods	LD44	469	146	614	0 763	0.237	1,158	884	274	
20	mods	LD48 Exp	181	148	329	0 550	0.450	4	2	2	
21	mods	LD48 Oth	953	99	1,052	0 906	0.094	959	868	91	
22	mods	LD48_SSv	370	58	428	0 865	0.135	280	242	38	
23	mods	LD49	659	-	659	1.000	-	1,555	1,555	-	
24	mods	LD79	556	97	653	0 851	0.149	4,239	3,608	630	
25	mods	MAILGRAMS	-	-	-			-			
26	mods	Registry	-	-	-			3	No Key	No Key	3
27	mods	REWRAP	-	-	-			14	No Key	No Key	14
28	mods	1Bulk pr	269	-	269	1 000	-	585	585	-	
29	mods	1CancMPP	107	-	107	1 000	-	337	337	-	
30	mods	1EEQMT	-	-	-			1,093	No Key	No Key	1,093
31	mods	1MISC	137	-	137	1 000	-	1,322	1,322	-	
32	mods	1OPbulk	3,372	189	3,561	0.947	0 053	13,289	12,583	706	
33	mods	1OPpref	2,960	215	3,175	0.932	0.068	11,212	10,453	759	
34	mods	1Platfrm	2,388	117	2,505	0 953	0.047	21,933	20,905	1,028	
35	mods	1POUCHNG	756	70	827	0.915	0.085	3,581	3,276	304	
36	mods	1SackS_h	1,222	111	1,333	0 917	0.083	5,629	5,161	468	
37	mods	1SackS_m	158	-	158	1 000	-	1,826	1,826	-	
38	mods	1SCAN	2	-	2	1.000	-	94	94	-	
39	mods	1SUPPORT	83	-	83	1.000	-	1,255	1,255	-	
40	BMCs	nmo	213	160	373	0.572	0.428	661	378	283	
41	BMCs	psm	409	-	409	1 000	-	1,163	1,163	-	
42	BMCs	spb	638	60	698	0 914	0.086	2,097	1,917	180	
43	BMCs	ssm	421	-	421	1 000	-	2,422	2,422	-	
44	BMCs	Othr	1,285	-	1,285	1 000	-	4,464	4,464	-	
45	BMCs	Pla	1,112	109	1,220	0.911	0 089	6,085	5,542	542	
46	Non Mods		25,864	5,134	30,999	0.834	0.166	66,409	55,410	10,999	
	Total		70,027	9,392	79,418	0 882	0.118	215,694	192,131	22,363	1,200
									0.891	0.104	0.006
	Sources		Appendix A	Appendix A	(1) + (2)	(1) / (3)	(2) / (3)	LR-H-106	(4) * (6)	(5) * (6)	If (3) = 0 then (6)

Table 1

Development of Walk Sequence vs. Non-Walk Sequence Costs - Standard (A) Nonprofit ECR Letters

			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
			Direct Tally IOCS Costs by Presence of Walk-Sequence Endorsement			Percent of Total		Variable Mail	Variable Costs Distributed to WS/Non-WS		
	Group	Pool	Non-WS	WS	Total	=(1)/(3) Non-WS	=(2)/(3) WS	Processing Costs	=(4)*(6) Non-WS	=(5)*(6) WS	No Key
1	mods	bcs/	1,213	62	1,275	0 951	0 049	4,225	4,019	207	
2	mods	express	-	-	-			0	No Key	No Key	0
3	mods	fsm/	-	-	-			4	No Key	No Key	4
4	mods	lsm/	388	-	388	1 000	-	1,211	1,211	-	
5	mods	manf	-	-	-			1	No Key	No Key	1
6	mods	manl	1,245	72	1,317	0 945	0.055	2,018	1,908	111	
7	mods	manp	-	-	-			1	No Key	No Key	1
8	mods	mecparc	-	-	-			5	No Key	No Key	5
9	mods	ocr/	478	-	478	1.000	-	1,467	1,467	-	
10	mods	priority	-	-	-			11	No Key	No Key	11
11	mods	spbs Oth	333	-	333	1 000	-	668	668	-	
12	mods	spbsPrio	-	-	-			8	No Key	No Key	8
13	mods	BusReply	-	-	-			2	No Key	No Key	2
14	mods	INTL	-	-	-			-			
15	mods	LD15	-	-	-			-			
16	mods	LD41	45	-	45	1.000	-	137	137	-	
17	mods	LD42	-	-	-			2	No Key	No Key	2
18	mods	LD43	1,153	78	1,231	0 936	0 064	2,803	2,625	178	
19	mods	LD44	68	-	68	1 000	-	308	308	-	
20	mods	LD48 Exp	152	-	152	1 000	-	1	1	-	
21	mods	LD48 Oth	208	-	208	1.000	-	155	155	-	
22	mods	LD48_SSv	-	-	-			1	No Key	No Key	1
23	mods	LD49	59	-	59	1 000	-	138	138	-	
24	mods	LD79	280	-	280	1.000	-	1,919	1,919	-	
25	mods	MAILGRAMS	-	-	-			-			
26	mods	Registry	-	-	-			0	No Key	No Key	0
27	mods	REWRAP	-	-	-			1	No Key	No Key	1
28	mods	1Bulk pr	-	-	-			12	No Key	No Key	12
29	mods	1CancMPP	67	-	67	1.000	-	154	154	-	
30	mods	1EEQMT	-	-	-			218	No Key	No Key	218
31	mods	1MISC	54	-	54	1 000	-	261	261	-	
32	mods	1OPbulk	759	-	759	1 000	-	3,444	3,444	-	
33	mods	1OPpref	199	47	246	0.810	0 190	886	717	169	
34	mods	1Platfrm	57	-	57	1.000	-	1,675	1,675	-	
35	mods	1POUCHNG	72	-	72	1 000	-	240	240	-	
36	mods	1SackS_h	133	-	133	1.000	-	534	534	-	
37	mods	1SackS_m	61	-	61	1 000	-	555	555	-	
38	mods	1SCAN	-	-	-			13	No Key	No Key	13
39	mods	1SUPPORT	-	-	-			207	No Key	No Key	207
40	BMCs	nmo	50	-	50	1 000	-	106	106	-	
41	BMCs	psm	-	-	-			-			
42	BMCs	spb	-	-	-			2	No Key	No Key	2
43	BMCs	ssm	45	-	45	1.000	-	294	294	-	
44	BMCs	Othr	185	54	239	0.773	0.227	811	628	184	
45	BMCs	Pla	154	-	154	1 000	-	731	731	-	
46	Non Mods		3,229	69	3,298	0.979	0 021	7,356	7,202	154	
	Total		10,689	383	11,071	0.965	0.035	32,588	31,099	1,002	488
									0.954	0 031	0 015
	Sources		Appendix A	Appendix A	(1) + (2)	(1) / (3)	(2) / (3)	LR-H-106	(4) * (6)	(5) * (6)	If (3) = 0 then (6)

Table 1

Development of Walk Sequence vs. Non-Walk Sequence Costs - Standard (A) Nonprofit ECR Non-Letters

			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
			Direct Tally IOCS Costs by Presence of Walk-Sequence Endorsement			Percent of Total		Variable Mail	Variable Costs Distributed to WS/Non-WS		
	Group	Pool	Non-WS	WS	Total	=(1)/(3) Non-WS	=(2)/(3) WS	Processing Costs	=(4)*(6) Non-WS	=(5)*(6) WS	No Key
1	mods	bcs/	-	-	-			15	No Key	No Key	15
2	mods	express	-	-	-			0	No Key	No Key	0
3	mods	fsm/	411	-	411	1 000	-	972	972	-	
4	mods	lsm/	70	-	70	1.000	-	226	226	-	
5	mods	manf	420	-	420	1.000	-	841	841	-	
6	mods	manl	-	-	-			1	No Key	No Key	1
7	mods	manp	-	-	-			0	No Key	No Key	0
8	mods	mecparc	59	-	59	1 000	-	141	141	-	
9	mods	ocr/	61	-	61	1.000	-	141	141	-	
10	mods	priority	-	-	-			1	No Key	No Key	1
11	mods	spbs Oth	72	-	72	1 000	-	118	118	-	
12	mods	spbsPrio	-	-	-			9	No Key	No Key	9
13	mods	BusReply	-	-	-			0	No Key	No Key	0
14	mods	INTL	-	-	-			1	No Key	No Key	1
15	mods	LD15	-	-	-			-			
16	mods	LD41	-	-	-			-			
17	mods	LD42	70	-	70	1.000	-	43	43	-	
18	mods	LD43	925	-	925	1 000	-	1,986	1,986	-	
19	mods	LD44	48	-	48	1 000	-	86	86	-	
20	mods	LD48 Exp	85	-	85	1 000	-	-	-	-	
21	mods	LD48 Oth	124	-	124	1 000	-	79	79	-	
22	mods	LD48_SSv	-	-	-			0	No Key	No Key	0
23	mods	LD49	45	83	129	0.353	0 647	301	106	195	
24	mods	LD79	-	-	-			0	No Key	No Key	0
25	mods	MAILGRAMS	-	-	-			-			
26	mods	Registry	-	-	-			-			
27	mods	REWRAP	-	-	-			-			
28	mods	1Bulk pr	-	-	-			6	No Key	No Key	6
29	mods	1CancMPP	-	-	-			3	No Key	No Key	3
30	mods	1EEQMT	-	-	-			66	No Key	No Key	66
31	mods	1MISC	-	-	-			57	No Key	No Key	57
32	mods	1OPbulk	265	-	265	1.000	-	770	770	-	
33	mods	1OPpref	63	-	63	1.000	-	178	178	-	
34	mods	1Platfrm	128	-	128	1.000	-	1,089	1,089	-	
35	mods	1POUCHNG	105	-	105	1 000	-	383	383	-	
36	mods	1SackS_h	54	63	117	0.459	0 541	437	201	236	
37	mods	1SackS_m	-	-	-			22	No Key	No Key	22
38	mods	1SCAN	-	-	-			3	No Key	No Key	3
39	mods	1SUPPORT	-	-	-			63	No Key	No Key	63
40	BMCs	nmo	-	-	-			0	No Key	No Key	0
41	BMCs	psm	54	-	54	1.000	-	150	150	-	
42	BMCs	spb	92	-	92	1.000	-	229	229	-	
43	BMCs	ssm	-	-	-			2	No Key	No Key	2
44	BMCs	Othr	102	-	102	1.000	-	283	283	-	
45	BMCs	Pla	-	-	-			52	No Key	No Key	52
46	Non Mods		922	147	1,069	0.862	0 138	3,522	3,036	486	
	Total		4,177	294	4,472	0.934	0.066	12,281	11,060	917	304
									0.901	0.075	0.025
	Sources:		Appendix A	Appendix A	(1) + (2)	(1) / (3)	(2) / (3)	LR-H-106	(4) * (6)	(5) * (6)	If (3) = 0 then (6)

Table 2

Summary of Walk Sequence vs. Non-Walk Sequence Costs
Standard (A) Enhanced Carrier-Route Mail

Regular Rate

				With No Key Distributed		
	(1) Not WS Endorsed	(2) WS Endorsed	(3) No Key	(4) Not WS Endorsed	(5) WS Endorsed	Source
Letters	181,792	8,646	1,944	183,648	8,734	Table 1, pg 1
Non-Letters	192,131	22,363	1,200	193,206	22,488	Table 1, pg 2
Total				376,854	31,222	
Sources:	Table 1	Table 1	Table 1	(1) + (3)* (1)/((1)+(2))	(1) + (3)* (2)/((1)+(2))	

Nonprofit

				With No Key Distributed		
	(1) Not WS Endorsed	(2) WS Endorsed	(3) No Key	(4) Not WS Endorsed	(5) WS Endorsed	Source
Letters	31,099	1,002	488	31,571	1,017	Table 1, pg 3
Non-Letters	11,060	917	304	11,340	940	Table 1, pg 4
Total				42,912	1,957	
Sources:	Table 1	Table 1	Table 1	(1) + (3)* (1)/((1)+(2))	(1) + (3)* (2)/((1)+(2))	

Appendix A

Computer Documentation

The processing for this library reference was performed mostly on a Data General minicomputer running the DGUX version of the UNIX operating system. Source programs ending with a ".f" are FORTRAN programs, and programs ending in a ".sm" are Sort/Merge programs.

Program Documentation

I. Assign IOCS tallies to mail processing cost pools.

- A. Program : cadoc22c_std.f - This program uses various IOCS fields to assign mail processing tallies to cost pool.
- B. Input files:
  1. IOCS file extract - available in LR-H-23.
  2. fins.dat - map from finance number to ROG code. This file is being provided as part of LR-H-146 because of data security issues.
- C. Output files
  1. mods12_mp96_cw.dat2 - Mail processing tallies in MODs 1 & 2 cost pools.
  2. bmcs_mp96_cw.dat2 - Mail processing tallies in BMC cost pools
  3. nonmods_mp96_cw.dat2 - Mail processing tallies in the non-MODs cost pool.

II. Develop direct tally ECR costs for MODs 1 & 2 cost pools.

- A. Program: walkseq.f - Adds up ECR cost by cost pool, activity code, and walk-sequence status.
- B. Input files:
  1. mods12_mp96_cw.dat2
  2. activity.3c - list of Std A activity codes.
- C. Output files:
  1. mod3cws.csv - file to be imported into Excel.

III. Develop direct tally ECR costs for BMC cost pools.

- A. Program: wsbmc.f - Adds up ECR cost by cost pool, activity code, and walk-sequence status.
- B. Input files:
  1. bmcs_mp96_cw.dat2
  2. activity.3c - list of Std A activity codes
- C. Output files:

1. bmc3cws.csv - file to be imported into Excel.

IV. Develop direct tally zone parcel post costs for the non-MODs cost pool.

A. Program: wsnmod.f - Adds up ECR cost by cost pool, activity code, and walk-sequence status.

B. Input files:
   1. nonmods_mp96_cw.dat2
   2. activity.3c - list of Std A activity codes.

C. Output files:
   1. nmod3cws.csv - file to be imported into Excel

V. Summarize results in Excel spreadsheet.

A. File: wscost.xls
   1. Sheet mod3cws: This sheet contains the file produced by walkseq f. A column is added to the right of the text file. This contains an index code used by the sumif() functions in Table 1 to summarize the data in the proper place.
   2. Sheet bmc3cws: Similar to sheet mod3cws, this contains the corresponding data for BMC cost pools.
   3. Sheet nmod3cws: Similar to sheet mod3cws, this contains the corresponding data for the non-MODs cost pool.
   4. Sheet maps: This sheet contain maps from the cost pool numbering used in the Fortran code to the cost pool number in the spreadsheet.
   5. Sheet LRC: This sheet contains Table 1, page 1
   6. Sheet FRC: This sheet contains Table 1, page 2
   7. Sheet LNC: This sheet contains Table 1, page 3
   8. Sheet FNC: This sheet contains Table 1, page 4.
   9. Sheet consolidate: This sheet contains Table 2.

Appendix B

Program Source Codes

```
      Program cadoc22c_std

c      Purpose:  To generate IOCS cost pool groups including admin/windows

      IMPLICIT NONE

      integer*4    keep, hand, type, bmcgrp, ier, ct, i1, i2, i3, i4, i5
      integer*4    fins, costpool, searchc, i,  x, y, ipool, route, ibmc
      integer*4    ld, rd, inonmod, ct0
      parameter    (fins=32685)
      parameter    (ipool=59)
      parameter    (ibmc=41)
      parameter    (inonmod=30)

      integer*4    pool          !  function to assign cost pool group
      integer*4    q19           !  function to use Q19 to assign invalid modes codes
      integer*4    type1         !  function for manual type for Q19
      integer*4    type2         !  funciton for transportation type for Q19
      integer*4    rog(fins)/fins*0/,cpool(ipool)/ipool*0/,group(ibmc)/ibmc*0/
      integer*4    count(ipool)/ipool*0/, countb(ibmc)/ibmc*0/
      integer*4    countn(inonmod)/inonmod*0/,groupn(inonmod)/inonmod*0/
      integer*4    ct1/0/, ct2/0/, ct3/0/, ctaw1/0/, ctmp1/0/, ctaw2/0/
      integer*4    ctmp2/0/, ctaw3/0/, ctmp3/0/, ctkeep/0/, cthand/0/,ctnonh/0/
      integer*4    f262, f260, f257, iw, actv, f244, i6, sort

      real*8       f9250, wgt, dlrs
      real*8       doll(ipool)/ipool*0.0/,dollb(ibmc)/ibmc*0.0/
      real*8       dolln(inonmod)/inonmod*0.0/

      character*263 rec
      character*6  fin(fins)/fins*' '/, f2
      character*1  f1,f7,f116,f118,f119,f121,f122,f9209,f128,f9211,f9212
      character*1  f9214,f9219,f9602,f117,f155,f156,f157,f9606,f9632
      character*3  f114,f246,f247,f248,f249
      character*4  cf244

c     read finance numbers map

      open(10,file='fins.dat',iostat=ier)

11    format(a6,i2)
      do i=1,fins
         read(10,11) fin(i), rog(i)
      end do

      close(10)

21    format(a263)
      open(30,file='mods12_mp96_cw.dat2',recl=300,iointent='output')
      open(35,file='mods12_aw96_cw.dat2',recl=300,iointent='output')
31    format(a263,f15.5,i2,i1,i3,i5)
32    format(a263,f15.5,i2,i1,i1,i3,i5)
      open(40,file='bmcs_mp96_cw.dat2',recl=300,iointent='output')
      open(45,file='bmcs_aw96_cw.dat2',recl=300,iointent='output')
41    format(a263,f15.5,i2,i3,i5)
      open(50,file='nonmods_mp96_cw.dat2',recl=300,iointent='output')
      open(55,file='nonmods_aw96_cw.dat2',recl=300,iointent='output')
      open(80,file='iocs_bad.dat',recl=300,iointent='output')

51    format(a263,f15.5,i1,i3,i5)

      ier=0
      ct=0
      ct0=0
      i1=0
      i2=0
      i3=0
      i4=0
      i5=0
330   format(a1,1x,a6,9x,a1,13x,a3,4a1,4x,2a1,4x,a1,
     +   2x,4a1,2x,a1,4x,a1,73x,a1,33x,4a1,35x,i4,4a3,i2,i2,1x,i4,11x,f10.0)

99    do while (ier.eq.0)
         keep=0
         hand=0
         type=0
         bmcgrp=0
         read(5,21,iostat=ier,end=100) rec
         if (rec(228:229).eq.'$$') then
            write (80,21) rec
```

```
      ct0 = ct + 1
      print *, "$$ in record ",ct0
      rec(228:229) = '  '
      goto 99
    end if
    iw = 1
    read(rec,330) f1, f2, f7, f114, f116, f117, f118, f119, f121, f122,
+      f9209, f128, f9211, f9212, f9602, f9214, f9219, f9606, f9632, f155,
+      f156, f157, f244, f246, f247, f248, f249, f257, f260, f262, f9250

    cf244 = rec(210:213)

    ct=ct+1
    i1=searchc(fin,fins,f2)
    if ((f257.eq.11).or.(f257.eq.12).or.
+      (f257.eq.31).or.(f257.eq.32).or.(f257.eq.41)
+      .or.(f257.eq.42).or.(f257.eq.61).or.
+      (f257.eq.62).or.(f257.eq.81).or.(f257.eq.82))
+      then
      keep=1
    end if

    if (f9250.le.0.0) then
      keep=0
    end if

    if (f7.eq.'K') then
      keep=0
    end if

    if (keep.eq.1) then
      ctkeep=ctkeep+1
    end if

    wgt=f9250/100000

    if (keep.eq.1) then
      if (rec(235:241).eq.'666666A') then !  BMCs
        type=1
        ct1=ct1+1
      else if (((f262.ge.10).and.(f262.le.4950)).or.
+           ((f9219.ge.'A').and.(f9219.le.'J')).or.
+           ((f9214.ge.'A').and.(f9214.le.'P'))) then
        hand=1
        cthand=cthand+1
      else
        hand=2
        ctnonh=ctnonh+1
      end if
      if ((hand.gt.0).and.(i1.gt.0)) then
        if ((rog(i1).eq.1).or.(rog(i1).eq.2)) then
          type=2
          ct2=ct2+1
        else
          type=3
          ct3=ct3+1
        end if
      end if

      if ((hand.gt.0).and.(i1.eq.0)) then
        type=3
        ct3=ct3+1
      end if

      if (type.eq.2) then
        i2=pool(f114)
        i3=q19(f128)
        i4=type1(f9211,f9602)
        i5=type2(f9212)
        i6=sort(f9602)

        if (i2.eq.1) then
          if (f1.eq.'1') then
            if ((i3.eq.1).and.(i4.eq.1)) then
              i2=2
            else if ((i3.eq.1).and.(i4.eq.2)) then
              i2=3
            else if ((i3.eq.1).and.(i4.eq.3)) then
              i2=4
            else if ((i3.eq.1).and.(i4.eq.4)) then
```

```
            i2=36
         else if ((i3.eq.1).and.(i4.eq.5)) then
            i2=35
         else if ((i3.eq.1).and.(i4.eq.6)) then
            i2=30
         else if ((i3.eq.1).and.(i4.eq.7)) then
            i2=32
         else if ((i3.eq.1).and.(i4.eq.8)) then
            i2=29
         else if (i3.eq.2) then
            i2=10
         else if ((i3.ge.3).and.(i3.le.5)) then
            i2=11
         else if (i3.eq.6) then
            i2=8
         else if ((i3.ge.7).and.(i3.le.8)) then
            i2=36
         else if (i3.eq.9) then
            i2=34
         else if (i3.eq.10) then
            i2=5
         else if (i3.eq.11) then
            i2=9
         else if (i3.eq.12) then
            i2=6
         else if (i3.eq.13) then
            i2=34
         else if (i3.eq.14) then
            i2=33
         else if (i3.eq.15) then
            i2=30
         else if (i3.eq.16) then
            i2=37
         else if (i3.eq.17) then
            i2=32
         else if (i3.eq.18) then
            i2=36
         else if (i3.eq.19) then
            i2=17
         else if ((i3.eq.20).and.((i5.ge.1).and.(i5.le.4))) then
            i2=29
         else if (f260.eq.7) then
            i2=26
         else if (f260.eq.8) then
            i2=29
         else if ((f118.eq.'A').or.(f118.eq.'C').or.
     +           (f118.eq.'E').or.(f118.eq.'F').or.
     +           (f118.eq.'I').or.(f118.eq.'K')) then
            i2=36
         else if ((f118.eq.'B').or.(f118.eq.'D').or.
     +           (f118.eq.'H').or.(f118.eq.'J')) then
            i2=30
         else if (f118.eq.'G') then
            i2=19
         else if (((f119.ge.'A').and.(f119.le.'F')).and.
     +           (f122.eq.' ').and.(i6.eq.1)) then
            i2=4
         else if (((f119.ge.'A').and.(f119.le.'F')).and.
     +           (f122.eq.' ').and.(i6.eq.3)) then
            i2=33
         else if (((f119.ge.'A').and.(f119.le.'F')).and.
     +           (f122.eq.' ').and.(i6.eq.4)) then
            i2=4
         else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +           (f122.eq.' ').and.((i3.eq.1).and.(i4.eq.9))) then
            i2=42
         else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +           (f122.eq.' ').and.(i6.eq.2)) then
            i2=32
         else if ((f122.ge.'A').and.(f122.le.'F')) then
            i2=32
         else if ((f122.ge.'I').and.(f122.le.'L')) then
            i2=32
         else if (f122.eq.'G') then
            i2=42
         else if (f122.eq.'M') then
            i2=42
         else if (f122.eq.'H') then
            i2=38
         else if (f260.eq.0) then
```

```
                  i2=18
               else if (f260.eq.6) then
                  i2=40
               else if (f260.eq.9) then
                  i2=39
               else if (f260.eq.10) then
                  i2=58
               else if (f260.eq.14) then
                  i2=25
               else if (f260.eq.17) then
                  i2=45
               else if (f260.eq.18) then
                  i2=16
               else if (f260.eq.19) then
                  i2=20
               else if (f260.eq.20) then
                  i2=40
               else if (f260.eq.21) then
                  i2=24
               else if (f260.eq.22) then
                  i2=15
               else if (f260.eq.23) then
                  i2=24
               else if ((f260.ge.24).and.(f260.le.26)) then
                  i2=39
               else
                  i2=42
               end if
            end if

            if (f1.eq.'4') then
               i2 = 59
               if ((i3.ge.2).and.(i3.le.5)) then
                  i2=12
               else if (i3.eq.6) then
                  i2=13
               else if (i3.eq.11) then
                  i2=13
               else if (f260.eq.0) then
                  i2=23
               else if (f260.eq.6) then
                  i2=23
               else if ((f260.ge.11).and.(f260.le.13).or.
     +                (f260.eq.20)) then
                  i2=27
               else if ((f260.eq.9).or.((f260.ge.24).and.
     +                (f260.le.26))) then
                  i2=39
               else if (f260.eq.10) then
                  i2=59
               else if (f260.eq.14) then
                  i2=25
               else if (f260.eq.17) then
                  i2=45
               else if (f260.eq.18) then
                  i2=23
               else if (f260.eq.19) then
                  i2=20
               else if (f260.eq.21) then
                  i2=23
               else if (f260.eq.22) then
                  i2=21
               else if (f260.eq.23) then
                  i2=23
               else
                  i2=28
               end if
            end if
         end if

c     MODS-based encirclement rule

         if (((f262.ge.10).and.(f262.le.300))
     +      .and.(f9214.eq.' ')) then
            if (f262.eq.60) then
               actv = f262
            else if (f262.eq.190) then
               if ((f155.eq.'1').or.(f156.eq.'1').or.(f9606.eq.'A')
     +            .or.(f9606.eq.'B')) then
                  actv = f262
```

```
               else if ((f262.eq.190).and.(cf244(2:4).eq.'510')) then
                  actv = f262
               else if (((f246.eq.'   ').or.(f247.eq.'   ').or.
     +               (f248.eq.'   ').or.(f249.eq.'   ')).and.

     +               ((i2.eq.26).or.(i2.eq.39).or.(i2.eq.28)

     +               .or.(i2.eq.24).or.(i2.eq.23).or.(i2.eq.22)
     +               .or.(i2.eq.40).or.(i2.eq.41).or.(i2.eq.42)
     +               .or.(i2.ge.44))) then
                  actv = f262
               end if
            else if ((f262.eq.300).and.
     +               ((f157.eq.'1').or.(f9606.eq.'C')
     +               .or.(f9632.eq.'1'))) then
               actv = f262
            else if ((f246.eq.'   ').and.(f247.eq.'   ').and.
     +               (f248.eq.'   ').and.(f249.eq.'   ')) then
               actv = f244
               if (f262.eq.210) then
                  actv = f262
               else if ((f262.eq.90).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.35).or.(i2.eq.37).or.(i2.eq.32)
     +                  .or.(i2.eq.36).or.(i2.eq.30).or.(i2.eq.31)
     +                  .or.(i2.eq.33).or.(i2.eq.42).or.(i2.eq.40)
     +                  .or.(i2.eq.41).or.(i2.eq.28).or.
     +                  (i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if (((f262.eq.30).or.(f262.eq.70).or.
     +                  (f262.eq.80)).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.42).or.(i2.eq.41)
     +                  .or.(i2.eq.40).or.(i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if ((f262.eq.10).and.
     +                  ((i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.21).or.(i2.eq.15).or.(i2.eq.42)
     +                  .or.(i2.eq.40).or.(i2.eq.41))) then
                  actv = f262
               else if ((f262.eq.50).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.42).or.(i2.eq.41)
     +                  .or.(i2.eq.40).or.(i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if ((f262.eq.20).and.
     +                  ((i2.eq.35).or.(i2.eq.37).or.(i2.eq.32)
     +                  .or.(i2.eq.36).or.(i2.eq.30).or.(i2.eq.31)
     +                  .or.(i2.eq.33).or.(i2.eq.26).or.(i2.eq.29)
     +                  .or.(i2.eq.28).or.(i2.eq.39))) then
                  actv = f262
               end if
            else if ((f246.gt.'001').or.(f247.gt.'001')
     +               .or.(f248.gt.'001').or.(f249.gt.'001')) then
               actv = f244
               if (((f262.eq.30).or.(f262.eq.70).or.
     +                  (f262.eq.80)).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.42).or.(i2.eq.41)
     +                  .or.(i2.eq.40).or.(i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if ((f262.eq.10).and.
     +                  ((i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.21).or.(i2.eq.15).or.(i2.eq.42)
     +                  .or.(i2.eq.40).or.(i2.eq.41))) then
                  actv = f262
               else if ((f262.eq.50).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.42).or.(i2.eq.41)
     +                  .or.(i2.eq.40).or.(i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if ((f262.eq.20).and.
     +                  ((i2.eq.35).or.(i2.eq.37).or.(i2.eq.32)
     +                  .or.(i2.eq.36).or.(i2.eq.30).or.(i2.eq.31)
```

```
+                    .or.(i2.eq.33).or.(i2.eq.26).or.(i2.eq.29)
+                    .or.(i2.eq.28).or.(i2.eq.39))) then
                  actv = f262
               end if
            end if
         else
            actv = f262
         end if


         if ((i2.eq.39).or.(i2.ge.44)) then
            if (i2.eq.39) then
               dlrs=wgt*634148939/728038888
            else if (i2.eq.49) then
               dlrs=0.0
            else
               dlrs=wgt*680668580/864658254
            end if
            if (i2.eq.39) then
               costpool=1
            else
               costpool=2
            end if
            write(35,32) rec, dlrs, i2, hand, costpool, iw, actv
            ctaw2=ctaw2+1
         else
            write(30,31) rec, wgt, i2, hand, iw, actv
            ctmp2=ctmp2+1
         end if
         if (i2.le.59) then
            if ((i2.eq.39).or.(i2.ge.44)) then
               doll(i2)=doll(i2)+dlrs
            else
               doll(i2)=doll(i2)+wgt
            end if
            cpool(i2)=i2
            count(i2)=count(i2)+1
         end if
      end if              ! type 2

      if (type.eq.1) then
         if (((f260.ge.0).and.(f260.le.8)).or.
+           ((f260.ge.11).and.(f260.le.16)).or.
+           ((f260.ge.18).and.(f260.le.23)).or.
+           ((f260.ge.27).and.(f260.le.29))) then
            if ((f128.eq.'I').and.(f121.eq.'N')) then
               bmcgrp=31
            else if ((f128.eq.'I').and.(f121.eq.'Y')) then
               bmcgrp=32
            else if ((f128.eq.'J').and.(f121.eq.'N')) then
               bmcgrp=33
            else if ((f128.eq.'J').and.(f121.eq.'Y')) then
               bmcgrp=34
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                (f128.eq.'L')) then
               bmcgrp=35
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                (f9602.eq.'A')) then
               bmcgrp=35
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                (f9602.eq.'B')) then
               bmcgrp=35
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                (f128.eq.'M')) then
               bmcgrp=36
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                (f9211.eq.'C').and.(f9602.eq.'C')) then
               bmcgrp=36
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                (f9211.eq.'C').and.(f9602.eq.'D')) then
               bmcgrp=36
            else if (((f116.ge.'A').and.(f116.le.'H')).and.
+                (f9209.eq.' ')) then
               bmcgrp=37
            else if (((f118.ge.'A').and.(f118.le.'K')).and.
+                (f9209.eq.' ')) then
               bmcgrp=38
            else if (f9209.eq.' ') then
               bmcgrp=39
            else if ((f9209.ge.'A').and.(f9209.le.'F')) then
```

```
                bmcgrp=40
             end if
          else
             bmcgrp=f260
          end if

          if (bmcgrp.eq.0) then
             bmcgrp=41
          end if
          if (bmcgrp.le.30) then
             dlrs=wgt* 1.015935996
             write(45,41) rec, dlrs, bmcgrp, iw, actv
             ctaw1=ctaw1+1
          else
             dlrs=wgt * 1.015935996
             write(40,41) rec, dlrs, bmcgrp, iw, actv
             ctmp1=ctmp1+1
          end if
          dollb(bmcgrp)=dollb(bmcgrp)+dlrs
          group(bmcgrp)=bmcgrp
          countb(bmcgrp)=countb(bmcgrp)+1
       end if

       if (type.eq.3) then
          if (f260.eq.0) then
             f260=30
          end if
          if (((f260.ge.9).and.(f260.le.10)).or.
     +       (f260.eq.17).or.((f260.ge.24).and.
     +       (f260.le.26))) then
             dlrs=wgt * 1.051561404
             write(55,51) rec, dlrs, hand, iw, actv
             ctaw3=ctaw3+1
          else
             dlrs=wgt * 1.051561404
             write(50,51) rec, dlrs, hand, iw, actv
             ctmp3=ctmp3+1
          end if
          if (f260.gt.0) then
             dolln(f260)=dolln(f260)+dlrs
             groupn(f260)=f260
             countn(f260)=countn(f260)+1
          end if
       end if              !type.eq.3
    end if                 !keep

 end do

00 print*, "ier is ", ier
 print*, "Total Records ", ct
 print*, "Number of obs kept ", ctkeep
 print*, "BMC  Total Obs ", ct1, " BMC  Adm/Win ", ctaw1,
+   " BMC  MP ", ctmp1
 print *, "Number of handling ", cthand
 print *, "Number of non-handling ", ctnonh
 print*, "MODS Total Obs ", ct2, " MODS Adm/Win ", ctaw2,
+   " MODS MP ", ctmp2
 print*, "NMOD Total Obs ", ct3, " NMOD Adm/Win ", ctaw3,
+   " NMOD MP ", ctmp3


 open(60,file='mods12.dat',iointent='output')
 open(70,file='bmc.dat',iointent='output')
 open(75,file='nonmod.dat',iointent='output')
41 format(i3,f15.5,i10)
 1 format(i3,f15.5,i10)

 do i=1,ipool
    write(60,61) cpool(i), doll(i), count(i)
 end do

 do i=1, ibmc
    if (countb(i).gt.0) then
       write(70,71) group(i), dollb(i), countb(i)
    end if
 end do

 do i=1, inonmod
    if (countn(i).gt.0) then
       write(75,71) groupn(i), dolln(i), countn(i)
```

```
         end if
       end do

       end

c
c      assigns pool groups to MODS numbers

       function   pool(mod)

       integer*4    pool
       character*3  mod

       pool = 0

c      OCR OPERATIONS
       if (((mod.ge.'830').and.(mod.le.'837')).or.
     +   ((mod.ge.'840').and.(mod.le.'847')).or.
     +   ((mod.ge.'850').and.(mod.le.'857')).or.
     +   ((mod.ge.'880').and.(mod.le.'887'))) then
          pool = 10


c      BCS OPERATIONS
       else if ((mod.eq.'292').or.(mod.eq.'295').or.(mod.eq.'299').or.
     +       ((mod.ge.'860').and.(mod.le.'869')).or.
     +       ((mod.ge.'870').and.(mod.le.'879')).or.
     +       ((mod.ge.'890').and.(mod.le.'899')).or.
     +       ((mod.ge.'910').and.(mod.le.'911')).or.
     +       ((mod.ge.'914').and.(mod.le.'919')).or.
     +       ((mod.ge.'970').and.(mod.le.'979'))) then
          pool=11

c      LSM OPERATIONS
       else if (((mod.ge.'080').and.(mod.le.'089')).or.
     +       (mod.eq.'091').or.
     +       ((mod.ge.'093').and.(mod.le.'099'))) then
          pool=8

c      FSM OPERATIONS
       else if (((mod.ge.'190').and.(mod.le.'191')).or.
     +       ((mod.ge.'194').and.(mod.le.'197')).or.
     +       ((mod.ge.'140').and.(mod.le.'148')).or.
     +       ((mod.ge.'441').and.(mod.le.'444')).or.
     +       (mod.eq.'446').or.(mod.eq.'448').or.
     +       ((mod.ge.'960').and.(mod.le.'967'))) then
          pool=9

c      Mechanized sort-sack outside
       else if ((mod.ge.'238').and.(mod.le.'239')) then
          pool=34

c      MECHANIZED PARCEL SORTER
       else if ((mod.ge.'105').and.(mod.le.'106')) then
          pool=5

c      SMALL PARCEL BUNDLE SORTER
       else if ((mod.ge.'134').and.(mod.le.'137')) then
          pool=6
       else if ((mod.ge.'138').and.(mod.le.'139')) then
          pool=7

c      MANUAL FLAT OPERATIONS
       else if (((mod.ge.'060').and.(mod.le.'061')).or.
     +       ((mod.ge.'064').and.(mod.le.'079')).or.
     +       ((mod.ge.'170').and.(mod.le.'179'))) then
          pool=3

c      MANUAL LETTERS OPERATIONS
       else if (((mod.ge.'029').and.(mod.le.'031')).or.
     +       ((mod.ge.'034').and.(mod.le.'038')).or.
     +       ((mod.ge.'040').and.(mod.le.'049')).or.
     +       ((mod.ge.'150').and.(mod.le.'159')).or.
     +       ((mod.ge.'160').and.(mod.le.'169'))) then
          pool=2

c      MANUAL PARCEL OPERATIONS
       else if (((mod.ge.'100').and.(mod.le.'101')).or.
     +       (mod.eq.'104').or.(mod.eq.'130').or.
     +       ((mod.ge.'200').and.(mod.le.'207'))) then
```

```
          pool=4

c       MANUAL PRIORITY
        else if ((mod.ge.'050').and.(mod.le.'059')) then
          pool=14

c       LDC15
        else if ((mod.eq.'771').or.((mod.ge.'774').and.(mod.le.'776')).or.
     +       (mod.eq.'779')) then
          pool=17

c       ALLIED OPERATIONS

c       ACDCS
        else if ((mod.ge.'118').and.(mod.le.'119')) then
          pool=37

c       Bulk presort
        else if ((mod.ge.'002').and.(mod.le.'009')) then
          pool=35

c       Cancellation/mail prep
        else if (((mod.ge.'010').and.(mod.le.'019')).or.
     +    ((mod.ge.'020').and.(mod.le.'028'))) then
          pool=36

c       manual sack sort
        else if ((mod.ge.'235').and.(mod.le.'237')) then
          pool=33

c       opening unit - pref
        else if (((mod.ge.'110').and.(mod.le.'114')).or.
     +    ((mod.ge.'180').and.(mod.le.'184'))) then
          pool=30

c       opening unit - bbm
        else if (((mod.ge.'115').and.(mod.le.'117')).or.
     +    ((mod.ge.'185').and.(mod.le.'189'))) then
          pool=31

c       platform
        else if ((mod.ge.'210').and.(mod.le.'234')) then
          pool=29

c       pouching
        else if (((mod.ge.'120').and.(mod.le.'129')).or.
     +    ((mod.ge.'208').and.(mod.le.'209'))) then
          pool=32

c       BUSINESS REPLY / POSTAGE DUE
        else if (mod.eq.'930') then
          pool=18

c       DAMAGED PARCEL REWRAP
        else if (mod.eq.'109') then
          pool=19

c       empty equipment
        else if (mod.eq.'549') then
          pool=38

c       EXPRESS
        else if ((mod.eq.'131').or.(mod.eq.'669').or.(mod.eq.'793')) then
          pool=15

c       MAILGRAM
        else if (mod.eq.'584') then
          pool=20

c       MAIL PROCESSING SUPPORT
        else if (((mod.ge.'340').and.(mod.le.'341')).or.
     +       ((mod.ge.'554').and.(mod.le.'555')).or.
     +       (mod.eq.'547').or.(mod.eq.'548').or.(mod.eq.'607').or.
     +       (mod.eq.'612').or.(mod.eq.'620').or.(mod.eq.'625').or.
     +       (mod.eq.'630').or.(mod.eq.'677').or.(mod.eq.'755').or.
     +       (mod.eq.'798')) then
          pool=40

c       MISCELLANEOUS
        else if ((mod.ge.'560').and.(mod.le.'564')) then
```

```
          pool=42

         REGISTRY
        else if ((mod.ge.'585').and.(mod.le.'590')) then
          pool=16

c        LDC41 AND LDC42
        else if (((mod.ge.'821').and.(mod.le.'829')).or.
     +       ((mod.ge.'905').and.(mod.le.'906')).or.
     +       ((mod.ge.'912').and.(mod.le.'913'))) then
          pool=12
        else if ((mod.ge.'801').and.(mod.le.'819')) then
          pool=13

c        MANUAL DISTRIBUTION - STATION/BRANCH (LDC43)
        else if ((mod.ge.'240').and.(mod.le.'339')) then
          pool=28

c        STATION/BRANCH - BOX SECTION (LDC44)
        else if (mod.eq.'769') then
          pool=27

c        WINDOW Service
        else if (((mod.ge.'355').and.(mod.le.'453')).or.(mod.eq.'568')) then
          pool=39

c        LDC48
        else if (mod.eq.'583') then
          pool=21
        else if ((mod.eq.'353').or.(mod.eq.'558').or.(mod.eq.'559').or.
     +       (mod.eq.'608').or.(mod.eq.'621').or.(mod.eq.'626').or.
     +       (mod.eq.'631').or.(mod.eq.'678')) then
          pool=22
        else if ((mod.ge.'542').and.(mod.le.'544')) then
          pool=23
        else if ((mod.eq.'741').or.(mod.eq.'742').or.(mod.eq.'794')) then
          pool=24

c        ADDRESS INFO SYSTEM & CENTRAL MAIL MARK-UP
        else if ((mod.eq.'539').or.((mod.ge.'795').and.(mod.le.'797'))) then
          pool=25

c        MAILING REQUIREMENTS & BUSINESS MAIL ENTRY
        else if ((mod.eq.'001').or.(mod.eq.'550').or.(mod.eq.'660').or.
     +    (mod.eq.'697')) then
          pool=26

c        invalid mods code for mail processing
        else
          pool = 1
        end if

        if (pool.eq.1) then

c        INTERNATIONAL
          if ((mod.eq.'032').or.(mod.eq.'033')) pool=43
          if ((mod.eq.'062').or.(mod.eq.'063')) pool=43
          if ((mod.eq.'090').or.(mod.eq.'092')) pool=43
          if ((mod.eq.'192').or.(mod.eq.'193')) pool=43
          if ((mod.eq.'102').or.(mod.eq.'103')) pool=43
          if ((mod.eq.'107').or.(mod.eq.'108')) pool=43
          if (mod.eq.'132') pool=43
          if ((mod.ge.'346').and.(mod.le.'347')) pool=43
          if (mod.eq.'349') pool=43
          if (((mod.ge.'343').and.(mod.le.'345')).or.
     +       (mod.eq.'348').or.((mod.ge.'350').and.(mod.le.'352'))) pool=43
          if (mod.eq.'454') pool=43
          if ((mod.eq.'545').or.(mod.eq.'546').or.((mod.ge.'573').and.
     +       (mod.le.'578')).or.(mod.eq.'580').or.(mod.eq.'681')) pool=43

c        ADMINISTRATION

c        2adm_out
          if (mod.eq.'597') pool=49
          if ((mod.eq.'920').or.(mod.eq.'922').or.(mod.eq.'924')) pool=49
          if ((mod.eq.'342').or.(mod.eq.'598').or.(mod.eq.'698').or.
     +       (mod.eq.'699').or.((mod.ge.'700').and.(mod.le.'702')).or.
     +       (mod.eq.'770').or.((mod.ge.'927').and.(mod.le.'928')).or.
     +       (mod.eq.'932')) pool=49
          if ((mod.eq.'354').or.(mod.eq.'613').or.(mod.eq.'614').or.
```

```
     +      (mod.eq.'622').or.(mod.eq.'627').or.(mod.eq.'632').or.
     +      (mod.eq.'705').or.((mod.ge.'707').and.(mod.le.'711')).or.
     +      ((mod.ge.'713').and.(mod.le.'740')).or.
     +      (mod.eq.'743').or.(mod.eq.'744').or.(mod.eq.'757').or.
     +      (mod.eq.'768')) pool=49
            if ((mod.eq.'758').or.(mod.eq.'759').or.(mod.eq.'760')) pool=49
            if ((mod.eq.'676').or.(mod.eq.'933').or.((mod.ge.'951').and.
     +      (mod.le.'955'))) pool=49
            if ((mod.eq.'706').or.(mod.eq.'929')) pool=49
            if ((mod.eq.'599').or.(mod.eq.'635').or.(mod.eq.'703').or.
     +      (mod.eq.'923').or.((mod.ge.'935').and.(mod.le.'939'))) pool=49
            if ((mod.eq.'641').or.(mod.eq.'704').or.((mod.eq.'940').and.
     +      (mod.eq.'945'))) pool=49
            if ((mod.eq.'601').or.(mod.eq.'655').or.((mod.eq.'946').and.
     +      (mod.eq.'950'))) pool=49
            if (mod.eq.'671') pool=49
            if (mod.eq.'664') pool=49
            if (((mod.ge.'455').and.(mod.le.'462')).or.((mod.ge.'471').and.
     +      (mod.le.'504'))) pool=49

             function 0 admin
            if (mod.eq.'582') pool=50
            if ((mod.eq.'581').or.(mod.eq.'573')) pool=50
            if (((mod.ge.'594').and.(mod.le.'596')).or.(mod.eq.'674')) pool=50
            if ((mod.eq.'645').or.(mod.eq.'672')) pool=50
            if ((mod.eq.'668').or.(mod.eq.'900').or.(mod.eq.'905')) pool=50
            if ((mod.eq.'646').or.(mod.eq.'675')) pool=50

             function 3 admin
            if ((mod.eq.'615').or.(mod.eq.'617').or.(mod.eq.'679').or.
     +      (mod.eq.'763').or.(mod.eq.'764').or.(mod.eq.'901').or.
     +      (mod.eq.'906')) pool=53
            if ((mod.eq.'761').or.(mod.eq.'762')) pool=53
            if (mod.eq.'647') pool=50
            if ((mod.eq.'765').or.(mod.eq.'766').or.(mod.eq.'772').or.
     +      (mod.eq.'773')) pool=53
            if ((mod.ge.'750').and.(mod.le.'752')) pool=53
            if ((mod.ge.'753').and.(mod.le.'754')) pool=50
            if ((mod.ge.'747').and.(mod.le.'749')) pool=53
            if ((mod.eq.'616').or.(mod.eq.'624').or.(mod.eq.'629').or.
     +      (mod.eq.'634').or.(mod.eq.'680').or.(mod.eq.'745').or.
     +      (mod.eq.'746')) pool=53

              function 4 admin
            if ((mod.ge.'980').and.(mod.le.'987')) pool=51

              function 5 admin
            if ((mod.eq.'540').or.(mod.eq.'556').or.(mod.eq.'569').or.
     +      (mod.eq.'579').or.(mod.eq.'591').or.(mod.eq.'592').or.
     +      (mod.eq.'610').or.(mod.eq.'623').or.(mod.eq.'628').or.
     +      (mod.eq.'633').or.((mod.ge.'636').and.(mod.le.'640')).or.
     +      ((mod.ge.'648').and.(mod.le.'651')).or.
     +      ((mod.ge.'682').and.(mod.le.'685')).or.
     +      ((mod.ge.'968').and.(mod.le.'969'))) pool=52

              function 6 admin
            if (mod.eq.'958') pool=46
            if (mod.eq.'959') pool=46
            if ((mod.eq.'541').or.(mod.eq.'557').or.(mod.eq.'566').or.
     +      (mod.eq.'572').or.(mod.eq.'611').or.
     +      ((mod.ge.'642').and.(mod.le.'644')).or.
     +      ((mod.ge.'652').and.(mod.le.'654')).or.
     +      ((mod.ge.'686').and.(mod.le.'692')).or.
     +      (mod.eq.'902').or.(mod.eq.'907')) pool=54

              function 7 admin
            if (mod.eq.'656') pool=55
            if ((mod.eq.'657').or.(mod.eq.'693').or.(mod.eq.'695')) pool=55
            if ((mod.eq.'658').or.(mod.eq.'694')) pool=55
            if ((mod.eq.'659').or.(mod.eq.'696')) pool=55
            if ((mod.ge.'551').and.(mod.le.'552')) pool=45
            if (mod.eq.'661') pool=55
            if (mod.eq.'662') pool=55
            if ((mod.eq.'663').or.(mod.eq.'903')) pool=55

              function 8 admin
            if (((mod.ge.'463').and.(mod.le.'470')).or.((mod.ge.'505').and.
     +      (mod.le.'538')).or.(mod.eq.'570').or.(mod.eq.'571').or.
     +      (mod.eq.'648').or.(mod.eq.'665').or.(mod.eq.'666').or.
     +      (mod.eq.'670').or.(mod.eq.'682').or.(mod.eq.'904')) pool=44
```

```
c         function 9 admin
          if ((mod.ge.'780').and.(mod.le.'789')) pool=47

c         2adm_spc
          if (((mod.ge.'777').and.(mod.le.'778')).or.(mod.eq.'888').or.
     +      ((mod.ge.'988').and.(mod.le.'999'))) pool=57
          end if

          return
          end

c ------------------------------------------------------------------
c     Uses Q19 to assign invalid tallies to appropriate MODS group

          function   q19(mm)

          integer*4    q19
          character*1  mm

          q19=0
          if (mm.eq.'A') q19=1   ! Manual
          if (mm.eq.'B') q19=2   ! OCR
          if (mm.eq.'C') q19=3   ! BCR/BCS
          if (mm.eq.'D') q19=4   ! BCR/S
          if (mm.eq.'E') q19=5   ! Carrier BCS
          if (mm.eq.'F') q19=6   ! LSM
          if (mm.eq.'G') q19=7   ! Letter Facer
          if (mm.eq.'H') q19=8   ! Flat Facer
          if (mm.eq.'I') q19=9   ! Sack Sorter
          if (mm.eq.'J') q19=10  ! Parcel Sorter
          if (mm.eq.'K') q19=11  ! FSM
          if (mm.eq.'L') q19=12  ! SPBS
          if (mm.eq.'M') q19=13  ! NHO
          if (mm.eq.'N') q19=14  ! Multi-Slide
          if (mm.eq.'O') q19=15  ! Conveyor
          if (mm.eq.'P') q19=16  ! ACDCS
          if (mm.eq.'Q') q19=17  ! Banding
          if (mm.eq.'R') q19=18  ! Cutting
          if (mm.eq.'S') q19=19  ! RBCS
          if (mm.eq.'T') q19=20  ! Transportation
          if (mm.eq.'U') q19=21  ! Other

          return
          end

c ------------------------------------------------------------------
c     Assigns manual labor type for Q19 answer A

          function type1(man)

          integer*4    type1
          character*1  man

          type1=0
          if (man.eq.'A') type1=1   ! Letter
          if (man.eq.'B') type1=2   ! Flat
          if (man.eq.'C') type1=3   ! Parcel
          if (man.eq.'D') type1=4   ! Coll/Cancel/Mtr
          if (man.eq.'E') type1=5   ! Prsrt Units
          if (man.eq.'F') type1=6   ! Open Units
          if (man.eq.'G') type1=7   ! Pouch/Rack
          if (man.eq.'H') type1=8   ! Platf Units
          if (man.eq.'I') type1=9   ! Other

          return
          end

c ------------------------------------------------------------------

          function sort(f9602)

          integer*4    sort
          character*1  f9602

          sort=0

          if (f9602.eq.'A') sort=1   ! Sort Sacks
          if (f9602.eq.'B') sort=2   ! Sort Trays
          if (f9602.eq.'C') sort=3   ! Sort Pallets
```

```
      if (f9602.eq.'D') sort=4   !  Sort Rolling Containers

      return
      end
c
c      Assigns transportation type for Q19 answer T

      function type2(trp)

      integer*4    type2
      character*1  trp

      type2=0
      if (trp.eq.'A') type2=1
      if (trp.eq.'B') type2=2
      if (trp.eq.'C') type2=3
      if (trp.eq.'D') type2=4
      if (trp.eq.'E') type2=5
      return
      end
```

```
      Program walkseq

C     BY:   mike mcgrane
C     Date: 7/16/96
C
C     Purpose: program separates walk-sequence tallies in Std A as
C              separate activity code

      implicit none

      integer*4     nact, n3c, nzpp, nwalk, nbas, nop7, npool

      parameter     (n3c = 18)
      parameter     (nzpp = 5)
      parameter     (nact = 278)
      parameter     (nwalk = 2)
      parameter     (nbas = 5)
      parameter     (nop7 = 2)
      parameter     (npool = 43)


      real*8        dols3c(n3c,nwalk,npool)
      real*8        dols


      integer*4     ier, count, searchc, iwalk, iop7, ibf, iact
      integer*4     ipool, ihand, iw, iactv, i, j

      character*259 rec
      character*1   shape , rcshape
      character*4   act, cact
      character*4   acts(nact)
      character*4   acts3c(n3c), actszpp(nzpp)



      count=0
      ier=0

      open(15,file='activity.3c',iointent='input')
      read(15,'(a4)') acts3c


      do ipool = 1, npool
         do iwalk = 1, nwalk
            do iact = 1, n3c
               dols3c(iact,iwalk,ipool) = 0.
            end do
         end do
      end do


      open(20,file='../maps/activity.s',iointent='input')
21    format(a4)
      read(20,21) acts

11    format(a259,f15.5,i2,i1,i3,i5)

      do while (ier.eq.0)
         read(5,11,iostat=ier,end=100) rec, dols, ipool, ihand, iw, iactv
C         act = rec(231:234)     ! f262
         write (act,'(i4.4)') iactv
         if (rec(231:234).ne.act) then
            print *,' F262 ',rec(231:234),' ne Actv field ',act
         end if
         shape = rec(134:134)   ! f133
         rcshape = rec(136:136) ! f9635
         iwalk = 1              ! set to one for non-walk sequence

         cact = act

C        Edit Walk-Sequence Tallies to Have New Activity Codes

         if (act(1:1).ge.'1'.and.act(1:1).le.'4'.and.
     &       act(2:2).eq.'3') then ! 3c tally

            if (rec(150:150).eq.'1'.or. ! WS Marked - f9501
     &          rec(160:160).eq.'1'.or. ! ECRWSH marked - f9618
     &          rec(161:162).eq.'1'.or. ! ECRWSS marked - f9619
     &          (rec(139:139).eq.'Y'.and.act(1:1).le.'2')) then ! Detachd cd on ltr or flt
```

```
            iwalk = 2

         end if
         iact = searchc(acts3c,n3c,act)
         if (iact.gt.0) then
            dols3c(iact,iwalk,ipool) = dols3c(iact,iwalk,ipool) + dols
         else
            if (act(3:4).ne.'60') print *,' 3c activity not found ',act
         end if

      end if


   end do

00 print *, ' Read exit code = ',ier
   print *, ' Number of records processed ',count



   open(40,file='mod3cws.csv',iointent='output')
41 format(i2,',',i1,',',a4,',',f15.5)

   do ipool = 1, npool
      do iwalk = 1, nwalk
         do iact = 1, n3c
            if (dols3c(iact,iwalk,ipool).gt.0) then
               write (40,41) ipool, iwalk, acts3c(iact), dols3c(iact,iwalk,ipool)
            end if
         end do
      end do
   end do



   call exit
   end
```

```
      Program wsbmc

C     BY:   mike mcgrane
C     Date: 7/16/96
C
C     Purpose: program separates walk-sequence tallies in Std A as
C              separate activity code

      implicit none

      integer*4     nact, n3c, nzpp, nwalk, nbas, nop7, npool

      parameter     (n3c = 18)
      parameter     (nzpp = 5)
      parameter     (nact = 278)
      parameter     (nwalk = 2)
      parameter     (nbas = 5)
      parameter     (nop7 = 2)
      parameter     (npool = 11)


      real*8        dols3c(n3c,nwalk,npool)
      real*8        dols


      integer*4     ier, count, searchc, iwalk, iop7, ibf, iact
      integer*4     ipool, ihand, iw, iactv, i, j

      character*259 rec
      character*1   shape , rcshape
      character*4   act, cact
      character*4   acts(nact)
      character*4   acts3c(n3c), actszpp(nzpp)



      count=0
      ier=0

      open(15,file='activity.3c',iointent='input')
      read(15,'(a4)') acts3c


      do ipool = 1, npool
         do iwalk = 1, nwalk
            do iact = 1, n3c
               dols3c(iact,iwalk,ipool) = 0.
            end do
         end do
      end do


      open(20,file='../maps/activity.s',iointent='input')
21    format(a4)
      read(20,21) acts

11    format(a259,f15.5,i2,i3)

      do while (ier.eq.0)
         read(5,11,iostat=ier,end=100) rec, dols, ipool, iw
         count = count + 1

         ipool = ipool - 30  ! map onto 1 to 11

         act = rec(231:234)     ! f262
         shape = rec(134:134)   ! f133
         rcshape = rec(136:136) ! f9635
         iwalk = 1              ! set to one for non-walk sequence

         cact = act

C       Edit Walk-Sequence Tallies to Have New Activity Codes

         if (act(1:1).ge.'1'.and.act(1:1).le.'4'.and.
     &      act(2:2).eq.'3') then ! 3c tally

            if (rec(150:150).eq.'1'.or. ! WS Marked - f9501
     &         rec(160:160).eq.'1'.or. ! ECRWSH marked - f9618
     &         rec(161:162).eq.'1'.or. ! ECRWSS marked - f9619
     &         (rec(139:139).eq.'Y'.and.act(1:1).le.'2')) then ! Detachd cd on ltr or flt
```

```
            iwalk = 2

         end if
         iact = searchc(acts3c,n3c,act)
         if (iact.gt.0) then
            dols3c(iact,iwalk,ipool) = dols3c(iact,iwalk,ipool) + dols
         else
            if (act(3:4).ne.'60') print *,' 3c activity not found ',act
         end if

      end if


   end do

100 print *, ' Read exit code = ',ier
   print *, ' Number of records processed ',count



   open(40,file='bmc3cws.csv',iointent='output')
41 format(i2,',',i1,',',e4,',',f15.5)

   do ipool = 1, npool
      do iwalk = 1, nwalk
         do iact = 1, n3c
            if (dols3c(iact,iwalk,ipool).gt.0) then
               write (40,41) ipool+30, iwalk, acts3c(iact), dols3c(iact,iwalk,ipool)
            end if
         end do
      end do
   end do



   call exit
   end
```

```
      Program wsnmod

C     BY:   mike mcgrane
C     Date: 7/16/96
C
C     Purpose: program separates walk-sequence tallies in Std A as
C              separate activity code

      implicit none

      integer*4     nact, n3c, nzpp, nwalk, nbas, nop7, npool

      parameter     (n3c = 18)
      parameter     (nzpp = 5)
      parameter     (nact = 278)
      parameter     (nwalk = 2)
      parameter     (nbas = 5)
      parameter     (nop7 = 2)
      parameter     (npool = 1)


      real*8        dols3c(n3c,nwalk,npool)
      real*8        dols


      integer*4     ier, count, searchc, iwalk, iop7, ibf, iact
      integer*4     ipool, ihand, iw, iactv, i, j

      character*259 rec
      character*1   shape , rcshape
      character*4   act, cact
      character*4   acts(nact)
      character*4   acts3c(n3c), actszpp(nzpp)



      count=0
      ier=0

      open(15,file='activity.3c',iointent='input')
      read(15,'(a4)') acts3c


      do ipool = 1, npool
         do iwalk = 1, nwalk
            do iact = 1, n3c
               dols3c(iact,iwalk,ipool) = 0.
            end do
         end do
      end do


      open(20,file='../maps/activity.s',iointent='input')
21    format(a4)
      read(20,21) acts

11    format(a259,f15.5,i1,i3)

      do while (ier.eq.0)
         read(5,11,iostat=ier,end=100) rec, dols, ihand, iw
         ipool = 1
         count = count + 1


         act = rec(231:234)     ! f262
         shape = rec(134:134)   ! f133
         rcshape = rec(136:136) ! f9635
         iwalk = 1              ! set to one for non-walk sequence

         cact = act

C       Edit Walk-Sequence Tallies to Have New Activity Codes

         if (act(1:1).ge.'1'.and.act(1:1).le.'4'.and.
     &      act(2:2).eq.'3') then ! 3c tally

            if (rec(150:150).eq.'1'.or. ! WS Marked - f9501
     &         rec(160:160).eq.'1'.or. ! ECRWSH marked - f9618
     &         rec(161:162).eq.'1'.or. ! ECRWSS marked - f9619
     &         (rec(139:139).eq.'Y'.and.act(1:1).le.'2')) then ! Detachd cd on ltr or flt
```

```
            iwalk = 2

         end if
         iact = searchc(acts3c,n3c,act)
         if (iact.gt.0) then
            dols3c(iact,iwalk,ipool) = dols3c(iact,iwalk,ipool) + dols
         else
            if (act(3:4).ne.'60') print *,' 3c activity not found ',act
         end if

      end if


   end do

100 print *, ' Read exit code = ',ier
   print *, ' Number of records processed ',count



   open(40,file='nmod3cws.csv',iointent='output')
41 format(i1,',',a4,',',f15.5)

   do ipool = 1, npool
      do iwalk = 1, nwalk
         do iact = 1, n3c
            if (dols3c(iact,iwalk,ipool).gt.0) then
               write (40,41)  iwalk, acts3c(iact), dols3c(iact,iwalk,ipool)
            end if
         end do
      end do
   end do



   call exit
   end
```

Appendix C

Input and Output Files

File: activity.3c begins after this line
1310
330
340
1345
1350
2310
2330
2340
2345
2350
3310
3330
3340
3350
4310
4330
4340
4350

Table D-1: Format of FY96 IOCS Extract Used for Cadoc22c_Std

Field	First	Last	Length	Description	Source
1	1	2	2	Region and Division	PDC File
2	3	8	6	Finance Number	PDC File
6	9	17	9	Social Security Number	PDC File
7	18	18	1	CAG	PDC File
16	19	24	6	Begin Date	PDC File
9601	25	26	2	Pay Period	01 thru 27
27	27	27	1	Night Differential	Codes Q5A
28	28	28	1	Sunday Premium	Codes Q5A
112	29	29	1	In Office Activity	Codes Q16F
9206	30	30	1	Other Selections For 16F	Codes
9207	31	31	1	Work Center Type	Codes
114	32	34	3	FONS PSDS BMC Number	Codes Q18A
116	35	35	1	Platform Operations	Codes Q18B
117	36	36	1	Type Of Other Platform	
118	37	37	1	Collection And Prep Of Mail	Codes Q18C
119	38	38	1	Mail Proc And Distribution	Codes Q18D
120	39	39	1	Type Of INC SEC/CASE Dist	
9208	40	40	1	Type of Dist To Carriers	
9411	41	41	1	Type Sector/Segment Pass	
9412	42	42	1	Type DPS/Walk Sequence Pass	
121	43	43	1	Allied Labor	Codes 18D
122	44	44	1	Type Of Allied Labor	
123	45	45	1	Miscellaneous Operations	Codes Q18E
124	46	46	1	Window Service Part 1	Codes Q18F
125	47	47	1	Window Service Part 2	
126	48	48	1	Admin & Other Activities	Codes Q18G
9209	49	49	1	Employee On Break From	
9210	50	50	1	Other Selections For 18G	
9419	51	51	1	Type Distribution on Break From	
128	52	52	1	Manual Or Mech Operation	Codes Q19
9211	53	53	1	Type Of Manual Operation	
9212	54	54	1	Type of Transport Equipment	
9602	55	55	1	Sorting to	Codes
129	56	56	1	Directional Statement	Codes Q20
9213	57	57	1	Emp Hndl PC Item, Container	Codes 21A
9214	58	58	1	Single Item Type	Codes 21B
9215	59	59	1	Is The Item Empty	
9216	60	60	1	Identical Mail In Item	
9217	61	61	1	Top Piece Rule Apply	
9218	62	62	1	Contents Countable	
9219	63	63	1	Container Type	Codes 21C
9220	64	64	1	Is The Container Empty	
9221	65	65	1	Identical Mail In Container	
9901	66	69	4	Loose Card	Count
9902	70	73	4	Loose Letters	Count
9903	74	77	4	Loose Flats	Count
9904	78	81	4	Loose IPPS	Count
9905	82	85	4	Loose Parcels	Count
9906	86	88	3	Bundles	Count
9907	89	91	3	Con Cons	Count
9908	92	94	3	Flat Trays	Count

Table D-1 Continued

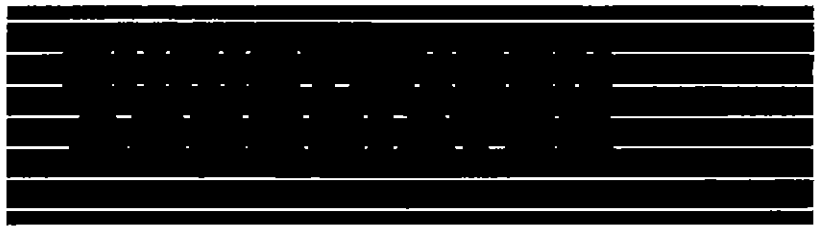
Field	First	Last	Length	Description	Source
9909	95	97	3	Letter Trays	Count
9910	98	100	3	SM Parcel Trays	Count
9911	101	103	3	Pallets	Count
9912	104	106	3	Other Items	Count
9913	107	109	3	Blue And Orange	SK Pouch CT
9914	110	112	3	Green	SK Pouch CT
9915	113	115	3	Orange Or Yellow	SK Pouch CT
9916	116	118	3	Brown	SK Pouch CT
9917	119	121	3	White	SK Pouch CT
9420	122	124	3	White # 2	SK Pouch CT
9421	125	127	3	White #3	SK Pouch CT
9918	128	130	3	Other	SK Pouch CT
9919	131	133	3	International	SK Pouch CT
133	134	134	1	Shape	Codes Q22A
134	135	135	1	Automation Compatible	Codes Q22B
9635	136	136	1	Shape Single Piece	Codes after June 31
9606	137	137	1	USPS Form	Codes After June 31
9222	138	138	1	RBCS Label	
135	139	139	1	Detached Address Card	Codes Q22A
911	140	140	1	Address Block	Codes Q22B
9608	141	141	1	Automation Rate Barcode	Yes or No after June 3
136	142	142	1	Indicia	Codes Q23A
137	143	143	1	Class Of Mail	Codes Q23B
9224	144	144	1	Carrier Route	
9611	145	145	1	Mail Markings - Class of Mail	Codes after June 31
9223	146	146	1	Dedicated Space	
138	147	147	1	Attachment Enclosure	
139	148	148	1	Secondary Markings-Zip+4	Codes Q23C
140	149	149	1	-Zip+4 Barcoded	
9501	150	150	1	-Walk Sequence	
9612	151	151	1	Auto First Class	
9613	152	152	1	Auto	
9614	153	153	1	Auto CR	
141	154	154	1	-Presorted	
9615	155	155	1	First Class Presorted	
9616	156	156	1	Single Piece	
142	157	157	1	-Carr Rte	
143	158	158	1	-Bulk Rate	
9617	159	159	1	ECRLOT	
9618	160	160	1	ECRWSH	
9619	161	161	1	ECRWSS	
144	162	162	1	-Nonprofit	
145	163	163	1	-Printed Matter	
9620	164	164	1	Special Standard Mail	
9621	165	165	1	Markings - Library Rate	
9463	166	166	1	-DMBC	
9464	167	167	1	-BSPS	
9225	168	168	1	-FIM	
146	169	169	1	-Business Reply	
9226	170	170	1	-Courtesy Mail	
9632	171	171	1	Additional Services Being Provided	
155	172	172	1	-Ancillary Mrkgs-Form 3811	Codes Q23D
156	173	173	1	-Form 3811A	

Table D-1 Continued

Field	First	Last	Length	Description	Source
157	174	174	1	-Form 3547/3579	
165	175	175	1	Weight Of Piece-LS 1/2 1 Oz	Codes Q23G
166	176	177	2	-Pounds	Codes Q23G
167	178	179	2	-Ounces	Codes Q23G
168	180	182	3	Revenue On Piece-Dollars	Codes Q23H
169	183	184	2	-Cents	Codes Q23H
170	185	185	1	-Mills	Codes Q23H
171	186	186	1	Postage Due Or Overpaid	Codes Q23I
172	187	189	3	Amount-Dollars	
173	190	191	2	-Cents	
174	192	192	1	-Mills	
176	193	200	8	ISSN Number	Codes Q23K
178	201	206	6	Publication Number	Codes Q23L
179	207	207	1	Application Pending	Codes Q23L
9227	208	209	2	Number Of Records Counted	Codes Q24
244	210	213	4	Activity Code	ALB040
246	214	216	3	-2	
247	217	219	3	-3	
248	220	222	3	-4	
249	223	225	3	-5	
257	226	227	2	Tally Roster Designation	ALB100
260	228	229	2	Tally Operation Route Code	ALB100
261	230	230	1	Tally Basic Function	ALB100
262	231	234	4	Tally Activity Code	ALB100
263	235	240	6	Tally Finance Number	ALB100
264	241	241	1	Tally Cag	ALB100
9246	242	245	4	Sample Weight (Heavy/Light Sample)	
9250	246	255	10	Tally Dollar Value	
9253	256	256	1	Is Tally Divided Item (A-X,Blank)	
9253	257	257	1	Is Tally Divided Item (A-X,Blank)	
9476	258	262	5	Total Item Count (Pre Division)	
9478	263	263	1	Type 24L BSPS (Y or N)	

Exhibit USPS-44B

Standard Mail (A) Unit Volume Variable Cost by Weight Increment



Economic Analysis and Consulting

LR-H-182

STANDARD MAIL (A) UNIT VOLUME VARIABLE COST BY WEIGHT INCREMENT U.S. POSTAL SERVICE

4610 University Avenue
Suite 700
Madison, Wisconsin 53705-2164

608 231 2266

LR-H-182 Standard Mail (A) Unit Volume Variable Cost by Weight Increment

Table of Contents

Introduction

This document contains estimates of unit volume variable cost for Standard Mail (A) by weight increment for carrier-route and other bulk mail separately. These estimates are constructed using FY 1996 data, and include both regular and nonprofit costs and volumes. Costs by weight increment for mail processing, window service, and city carrier in-office costs are based upon IOCS information. All other costs are distributed in proportion to either pieces, weight, or cubic volume. The basic conclusion of this analysis is that unit costs have a slight upward trending relationship with mail piece weight.

Table 1 shows the results for carrier route and other mail separately. These results are also presented graphically in Chart 1. Unit costs for carrier route mail are in the range of 5 to 7 cents per piece for the first seven ounce increments. Mail weighing more than seven ounces trends upwards to a peak of 18 cents at sixteen ounces. Other bulk mail shows a slightly more sloped path, beginning at 11 cents at one ounce, trending upwards towards 20 cents in the twelve to fourteen ounce range, and sharply rising to 49 cents at sixteen ounces.

For comparison to the all shape tables, Table 2 and Chart 2 contain the unit cost estimates for flat shaped mail only. The carrier route curve is similar to the curve for all shapes. The other mail curve has a high initial unit cost, then is similar to the all shapes curve in the middle ounce increments. In contrast to the all shapes curve, there is no sharp rise in cost in the heaviest weight increments.

Construction of Costs

The construction of costs by weight increment is shown in Tables 3 through 6 for the all shapes analysis. The construction of costs for the flat analysis is identical, and is contained in the spreadsheet accompanying the library reference. Each table is for a separate subclass and rate category. Each cost segment's cost is based upon the base year CRA report and is distributed to weight increment in the manner listed in the table. The construction of mail processing costs is documented in Appendix A, and the construction of LIOCATT costs for window service and city carrier in-office costs is documented in Appendix B. Estimates of mail volume and weight by weight increment are taken from library reference H-108, appendix A. The following is a summary of the methods used to distribute cost to weight increment:

1. Variable mail processing costs are distributed in proportion to direct IOCS tally costs by weight increment within each cost pool.
2. Window service costs are LIOCATT costs by weight increment.
3. City carrier in-office costs are LIOCATT costs by weight increment.
4. City carrier street costs are distributed to weight increment in proportion to mail volume.
5. Vehicle service driver costs are distributed to weight increment in proportion to estimated cubic volume. (The cube was estimated by applying a density estimate to the weight. The density estimate is constant over the entire range of weight increments for a given shape of mail.)
6. Rural carrier costs are distributed to weight increment in proportion to mail volume.
7. Air and water transportation costs are distributed in proportion to weight.
8. Highway and rail transportation costs are distributed in proportion to estimated cubic volume.
9. All other costs were distributed in proportion to pieces.

Table 1

FY 1996 Attributable Unit Cost by Weight Increment
Standard (A) Bulk Mail

Weight Inc. (oz)	Carrier Route			Other		
	Attributable Costs (000)	Mail Volume (000)	Unit Cost (cents)	Attributable Costs (000)	Mail Volume (000)	Unit Cost (cents)
1	788,270	11,884,976	6.6	2,285,008	19,888,875	11.5
2	386,172	6,618,447	5.8	959,157	8,310,370	11.5
3	310,369	6,100,688	5.1	545,665	4,143,309	13.2
4	215,977	3,024,681	7.1	521,302	3,025,509	17.2
5	120,104	2,352,129	5.1	195,698	1,615,153	12.1
6	62,508	1,145,220	5.5	151,920	904,275	16.8
7	29,064	495,384	5.9	76,972	546,745	14.1
8	16,047	176,959	9.1	84,282	370,421	22.8
9	10,646	137,224	7.8	46,548	255,938	18.2
10	6,992	70,751	9.9	48,357	201,637	24.0
11	3,727	39,292	9.5	39,991	165,235	24.2
12	1,939	21,572	9.0	50,452	168,569	29.9
13	2,239	33,805	6.6	41,204	154,530	26.7
14	1,710	13,118	13.0	42,003	127,321	33.0
15	1,731	12,681	13.7	25,253	62,867	40.2
16	1,946	10,735	18.1	21,044	37,420	56.2
	1,959,439	32,137,662	6.1	5,134,854	39,978,176	12.8

Chart 1

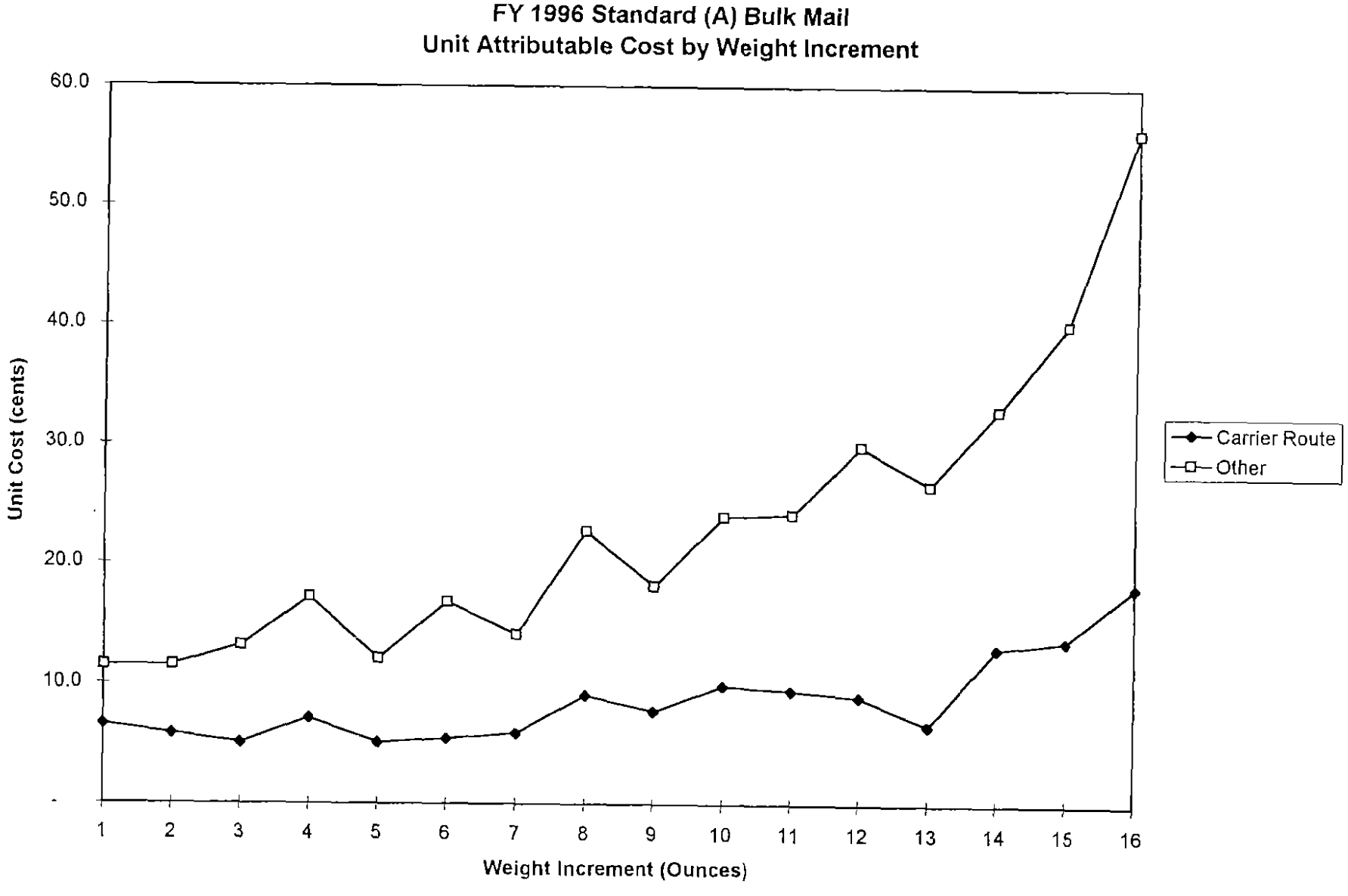

FY 1996 Standard (A) Bulk Mail
Unit Attributable Cost by Weight Increment
60.0
50.0
40.0
30.0
20.0
10.0
-
Unit Cost (cents)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Weight Increment (Ounces)
Carrier Route
Other

Table 2
FY 1996 Attributable Unit Cost by Weight Increment
Standard Mail (A) Bulk Other Flats

	Carrier Route			Other		
Weight Inc. (oz)	Attributable Costs	Mail Volume	Unit Cost	Attributable Costs	Mail Volume	Unit Cost
1	163,993	1,940,793	8.4	293,227	999,913	29.3
2	232,231	3,492,117	6.7	488,694	2,270,219	21.5
3	238,264	4,393,866	5.4	382,288	2,316,990	16.5
4	193,505	2,609,668	7.4	443,034	2,540,075	17.4
5	124,110	2,315,073	5.4	183,001	1,554,744	11.8
6	64,407	1,139,485	5.7	123,629	841,942	14.7
7	30,305	493,084	6.1	59,255	436,500	13.6
8	15,764	175,941	9.0	56,042	291,739	19.2
9	10,264	136,848	7.5	31,649	203,096	15.6
10	6,574	70,577	9.3	27,463	142,388	19.3
11	3,491	39,111	8.9	17,194	78,470	21.9
12	2,003	21,399	9.4	15,622	62,529	25.0
13	2,341	33,746	6.9	12,345	58,855	21.0
14	1,510	13,020	11.6	11,530	41,871	27.5
15	1,069	12,638	8.5	8,301	36,541	22.7
16	1,584	10,727	14.8	7,940	24,172	32.8
	1,091,415	16,898,093	6.5	2,161,215	11,900,045	18.2

Chart 2

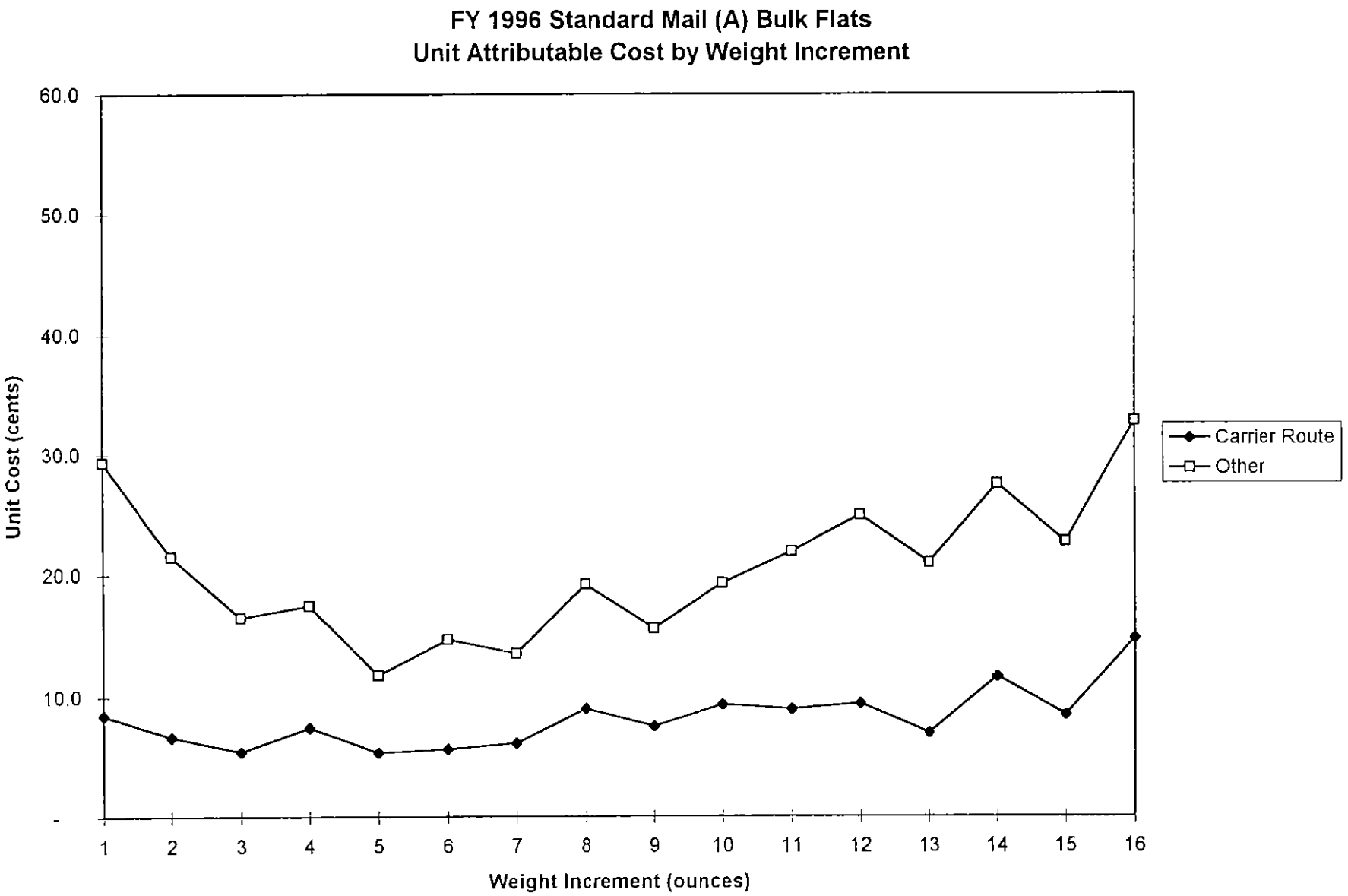

FY 1996 Standard Mail (A) Bulk Flats
Unit Attributable Cost by Weight Increment
60.0
50.0
40.0
30.0
20.0
10.0
-
Unit Cost (cents)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Weight Increment (ounces)
Carrier Route
Other

Table 3

Standard (A) Bulk Regular Carrier-Route Mail
FY 1996 Costs
(thousands of dollars)

CS	Description	Weight Increment (Ounces) 1	2	3	4	5	6	7	8	9	10	11	12
3.1	Mail Processing	172,681	84,721	49,948	59,539	15,602	8,313	3,594	4,017	2,157	2,293	1,362	215
	- Mail processing includes variable mail processing costs, adjustment for premium pay, and piggybacked costs												
3.1	Remote Endoding	3,590	1,232	732	173								
	- Remote encoding includes CS 3 1.1 and CS 16 3.6 costs, which are distributed to weight increments less than or equal to 4 ounces in proportion to letter-shaped pieces.												
3.2	Window Service	911	-	-	76	-	-	-	-	-	-	-	-
	- Window service is LIOCATT cost by weight increment multiplied by piggyback factor. See appendix B for documentation of LIOCATT costs by weight increment												
6	City Carrier Casing	221,616	83,469	51,936	44,818	15,610	8,925	4,611	4,315	2,112	1,205	435	630
	- City carrier casing is LIOCATT cost multiplied by in-office support and piggyback factors. See appendix B for documentation of LIOCATT costs by weight increment												
7	City Carrier Street	164,169	101,311	98,928	49,175	38,817	18,974	8,057	2,930	2,284	1,175	657	360
	- City carrier street includes all CS 7 and piggybacked costs, distributed to weight by proportion of pieces												
8	Vehicle Service Drivers	3,964	8,184	13,421	10,139	10,427	6,162	3,122	1,318	1,157	667	420	249
	- Vehicle service driver costs are distributed in proportion to estimated cubic volume.												
10	Rural Carriers	104,686	64,604	63,084	31,358	24,752	12,100	5,138	1,869	1,456	750	419	229
	- Rural carriers includes all CS 10 and piggybacked costs, distributed to weight by proportion of pieces.												
14	Air Transportation	213	390	620	438	442	263	133	56	49	28	18	11
	- Air and water transportation costs are distributed in proportion to estimated weight												
14	Hwy Transportation	3,117	6,437	10,555	7,974	8,200	4,846	2,455	1,036	910	525	330	196
	- Highway and rail tranportation costs are distributed in proportion to estimated cubic volume												

Table 3

Standard (A) Bulk Regular Carrier-Route Mail
FY 1996 Costs
(thousands of dollars)

CS	Description	Weight Increment (Ounces) 1	2	3	4	5	6	7	8	9	10	11	12
Othr	Remaining Costs	19,759	12,193	11,907	5,919	4,672	2,284	970	353	275	141	79	43

- All the remaining attributable costs are distributed in proportion to pieces.

	Total Costs	694,706	362,540	301,132	209,606	118,523	61,866	28,079	15,893	10,400	6,785	3,720	1,932

Standard (A) Bulk Regular Carrier-Route Mail
FY 1996 Mail Volumes

(thousands of pieces)	Weight Increment (Ounces) 1	2	3	4	5	6	7	8	9	10	11	12
Letters	7,985,368	2,739,608	1,628,272	383,915	28,837	4,132	1,695	599	109	74	88	58
Flats	1,795,697	3,277,499	4,264,703	2,528,305	2,278,293	1,125,887	478,220	173,741	135,825	69,941	39,002	21,319
Parcels	8,411	24,124	6,177	20,105	7,524	1,443	528	392	251	78	89	61
All Shapes	9,789,476	6,041,230	5,899,153	2,932,324	2,314,655	1,131,461	480,443	174,733	136,185	70,093	39,179	21,438

Standard (A) Bulk Regular Carrier-Route Mail
FY 1996 Weight

(thousands of pounds)	Weight Increment (Ounces) 1	2	3	4	5	6	7	8	9	10	11	12
Letters	237,206	246,383	254,314	79,315	7,746	1,393	684	272	58	44	58	42
Flats	74,065	319,362	648,989	554,804	635,833	381,402	193,419	81,228	71,588	41,426	25,913	15,298
Parcels	230	2,693	911	4,398	2,048	496	214	183	132	46	59	44
All Shapes	311,502	568,437	904,213	638,517	645,626	383,290	194,317	81,684	71,779	41,517	26,030	15,384

Standard (A) Bulk Regular Carrier-Route Mail
FY 1996 Cubic Volume Estimat

(thousands of cubic feet)	Weight Increment (Ounces) 1	2	3	4	5	6	7	8	9	10	11	12
Letters	8,346	8,669	8,948	2,791	273	49	24	10	2	2	2	1
Flats	3,586	15,464	31,424	26,864	30,787	18,468	9,365	3,933	3,466	2,006	1,255	741
Parcels	52	612	207	1,000	465	113	49	42	30	11	13	10
All Shapes	11,984	24,744	40,579	30,654	31,525	18,629	9,438	3,984	3,498	2,018	1,270	752

Standard (A) Bulk Regular Carrier-Route Mail
FY 1996 Unit Costs

(cents per piece)	Weight Increment (Ounces) 1	2	3	4	5	6	7	8	9	10	11	12
	7.1	6.0	5.1	7.1	5.1	5.5	5.8	9.1	7.6	9.7	9.5	9.0

Table 3

Standard (A) Bulk Regular Carri
FY 1996 Costs
(thousands of dollars)

CS	Description	13	14	15	16	Total	Description	Value	Source
3 1	Mail Processing	121	760	704	1,100	407,129			Appendix A
	- Mail processing includc								
3.1	Remote Endoding					5,726	WS 3.1.1 CS 16 3 6 REC Total	4,735 991 5,726	Base Year CRA
	- Remote encoding inclu								
3.2	Window Service	-	-	-	-	986	Piggyback	1.42261	LR-H-77
	- Window service is LIO								
6	City Carrier Casing	347	237	313	226	440,805	1) Support: 2) Piggyback' (1)*(2):	1.1736 1.3060 1.5327	LR-H-108 LR-H-77
	- City carrier casing is LI								
7	City Carrier Street	566	216	211	180	488,009	) Sum of CS 7.1 - 7. 2) Pigbk: 3) = (1)*(2)	373,661 1.3060 488,009	Base Year CRA LR-H-77
	- City carrier street inclu								
8	Vehicle Service Drivers	421	173	184	166	60,173	1) C.S. 8 2) Pigbk. 3) = (1)*(2)	38,819 1.5501 60,173	Base Year CRA LR-H-77
	- Vehicle service driver c								
10	Rural Carriers	361	138	134	115	311,192	1) C S. 10 2) Pigbk 3) = (1) * (2)	259,640 1.1986 311,192	Base Year CRA LR-H-77
	- Rural carriers includes								
14	Air Transportation	18	7	8	7	2,701	CS14 Air and Water	2,701	Base Year CRA
	- Air and water transport								
14	Hwy Transportation	331	136	144	130	47,324	CS 14 Highway and Rail	47,324	Base Year CRA
	- Highway and rail tranpo								

Standard (A) Bulk Regular Carri
FY 1996 Costs
(thousands of dollars)

CS	Description	13	14	15	16	Total	Description	Value	Source
Othr	Remaining Costs	68	26	25	22	58,735	1) Total Attributable	1,822,780	Base Year CRA
							2) Total Distributed	1,764,045	Sum of Above
	- All the remaining attrib						3) = (1) - (2)	58,735	
	Total Costs	2,235	1,693	1,724	1,945	1,822,780			

Standard (A) Bulk Regular Carri
FY 1996 Mail Volumes
(thousands of pieces)

	13	14	15	16	Total		
Letters	10	52	32	0	12,772,850		LR-H-108, Appx A
Flats	33,707	12,776	12,522	10,713	16,258,151		LR-H-108, Appx A
Parcels	40	31	11	8	69,273		LR-H-108, Appx A
All Shapes	33,757	12,859	12,565	10,722	29,100,273		

Standard (A) Bulk Regular Carri
FY 1996 Weight
(thousands of pounds)

	13	14	15	16	Total		
Letters	7	43	28	0	827,595		LR-H-108, Appx A
Flats	26,159	10,656	11,391	10,300	3,101,833		LR-H-108, Appx A
Parcels	32	26	10	8	11,529		LR-H-108, Appx A
All Shapes	26,198	10,725	11,429	10,309	3,940,957		

Standard (A) Bulk Regular Carri
FY 1996 Cubic Volume Estimat
(thousands of cubic feet)

	13	14	15	16	Total		Density Estmates	
Letters	0	2	1	0	29,118	Letter	28.4219	LR-H-108
Flats	1,267	516	552	499	150,191	Flats	20 6526	LR-H-108
Parcels	7	6	2	2	2,620	Parcels	4.4	LR-H-108
All Shapes	1,274	523	555	501	181,929			

Standard (A) Bulk Regular Carri
FY 1996 Unit Costs
(cents per piece)

13	14	15	16	Total
6.6	13 2	13.7	18.1	6.3

Table 4

Standard (A) Bulk Regular Other Mail
FY 1996 Costs
(thousands of dollars)

CS	Description	Weight Increment (Ounces) 1	2	3	4	5	6	7	8	9	10	11	12
3.1	Mail Processing	948,209	449,143	267,933	289,220	93,230	81,082	30,123	48,976	22,157	24,355	17,930	26,427
	- Mail processing includes variable mail processing costs, adjustment for premium pay, and piggybacked costs												
3.1	Remote Endoding	19,206	7,010	2,392	640								
	- Remote encoding includes CS 3.1.1 and CS 16.3.6 costs, which are distributed to weight increments less than or equal to 4 ounces in proportion to letter-shaped pieces.												
3.2	Window Service	2,492	511	276	961	254	290	-	-	121	-	252	-
	- Window service is LIOCATT cost by weight increment multiplied by piggyback factor See appendix B for documentation of LIOCATT costs by weight increment												
6	City Carrier Casing	326,701	117,251	66,965	63,016	16,593	13,126	7,591	6,379	3,944	3,691	2,264	2,214
	- City carrier casing is LIOCATT cost multiplied by in-office support and piggyback factors. See appendix B for documentation of LIOCATT costs by weight increment.												
7	City Carrier Street	182,575	86,670	50,027	38,214	20,434	11,595	6,869	4,728	3,314	2,614	2,166	2,237
	- City carrier street includes all CS 7 and piggybacked costs, distributed to weight by proportion of pieces.												
8	Vehicle Service Drivers	4,663	6,161	7,052	7,918	5,732	4,186	3,538	2,857	2,215	2,214	2,530	3,104
	- Vehicle service driver costs are distributed in proportion to estimated cubic volume.												
10	Rural Carriers	159,923	75,918	43,820	33,473	17,899	10,156	6,017	4,141	2,903	2,290	1,897	1,960
	- Rural carriers includes all CS 10 and piggybacked costs, distributed to weight by proportion of pieces.												
14	Air Transportation	2,155	2,609	2,672	2,787	1,934	1,343	944	752	594	526	482	545
	- Air and water transportation costs are distributed in proportion to estimated weight.												
14	Hwy Transportation	18,281	24,156	27,648	31,041	22,471	16,413	13,871	11,202	8,682	8,681	9,921	12,171
	- Highway and rail tranportation costs are distributed in proportion to estimated cubic volume.												

Table 4

Standard (A) Bulk Regular Other Mail
FY 1996 Costs
(thousands of dollars)

		Weight Increment (Ounces)										
CS	Description	1	2	3	4	5	6	7	8	9	10	11	12
Othr	Remaining Costs	13,809	6,555	3,784	2,890	1,545	877	520	358	251	198	164	169

- All the remaining attributable costs are distributed in proportion to pieces

Total Costs	1,678,014	775,984	472,568	470,159	180,092	139,069	69,472	79,392	44,180	44,568	37,607	48,828

Standard (A) Bulk Regular Other Mail
FY 1996 Mail Volumes
(thousands of pieces)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
Letters	12,572,719	4,588,598	1,566,128	418,707	27,609	14,199	5,254	2,305	4,086	1,014	835	1,015
Flats	728,408	1,700,003	2,014,199	2,331,770	1,435,156	787,029	398,968	270,748	191,511	133,153	73,365	58,576
Parcels	5,236	28,095	65,702	34,639	26,508	43,832	96,411	71,499	45,914	56,346	83,648	103,467
All Shapes	13,306,364	6,316,696	3,646,029	2,785,117	1,489,273	845,060	500,632	344,553	241,512	190,513	157,848	163,059

Standard (A) Bulk Regular Other Mail
FY 1996 Weight
(thousands of pounds)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
Letters	428,731	393,946	242,908	84,436	7,482	4,813	2,102	1,028	2,078	594	537	727
Flats	32,509	161,942	319,225	505,092	399,256	267,628	160,615	126,790	100,828	78,574	47,677	41,770
Parcels	250	2,862	10,113	7,445	7,515	15,176	39,365	33,139	24,299	33,532	55,044	74,131
All Shapes	461,489	558,750	572,246	596,973	414,252	287,616	202,082	160,957	127,206	112,699	103,258	116,628

Standard (A) Bulk Regular Other Mail
FY 1996 Cubic Volume Es
(thousands of cubic feet)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
Letters	15,085	13,861	8,546	2,971	263	169	74	36	73	21	19	26
Flats	1,574	7,841	15,457	24,457	19,332	12,959	7,777	6,139	4,882	3,805	2,309	2,023
Parcels	31	350	1,236	910	919	1,855	4,812	4,051	2,971	4,099	6,729	9,063
All Shapes	16,689	22,052	25,240	28,338	20,514	14,983	12,663	10,227	7,926	7,925	9,056	11,111

Standard (A) Bulk Regular Other Mail
FY 1996 Unit Costs
(cents per piece)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
	12.6	12.3	13.0	16.9	12.1	16.5	13.9	23.0	18.3	23.4	23.8	29.9

Standard (A) Bulk Regular
FY 1996 Costs
(thousands of dollars)

CS	Description	13	14	15	16	Total	Description	Value	Source
3.1	Mail Processing	17,638	20,086	15,366	14,995	2,366,869			Appendix A
	- Mail processing include								
3.1	Remote Endoding					29,248	WS 3.1 1	24,186	
							CS 16.3.6	5,062	Base Year CRA
	- Remote encoding inclu						REd Total	29,248	
3 2	Window Service	114	-	-	-	5,271	Piggyback	1.4221	LR-H-77
	- Window service is LIO								
6	City Carrier Casing	2,676	2,450	1,275	523	636,660	1) Support:	1.1736	LR-H-108
							2) Piggyback:	1.3125	LR-H-77
	- City carrier casing is LI						(1)*(2):	1.5402	
7	City Carrier Street	2,069	1,675	818	485	416,489	) Sum of CS 7.1 - 7	317,337	Base Year CRA
							2) Pigbk:	1.3125	LR-H-77
	- City carrier street inclu						3) = (1)*(2)	416,489	
8	Vehicle Service Drivers	3,109	2,814	1,186	705	59,986	1) C.S. 8	38,829	Base Year CRA
							2) Pigbk:	1.5449	LR-H-77
	- Vehicle service driver c						3) = (1)*(2)	59,986	
10	Rural Carriers	1,812	1,467	717	424	364,817	1) C.S 10	304,392	Base Year CRA
							2) Pigbk	1.1985	LR-H-77
	- Rural carriers includes						3) = (1) * (2)	364,817	
14	Air Transportation	550	478	250	159	18,779	CS14 Air and Water	18,779	Base Year CRA
	- Air and water transport								
14	Hwy Transportation	12,189	11,031	4,650	2,762	235,169	CS 14 Highway and Rail	235,169	Base Year CRA
	- Highway and rail tranpo								

Table 4

Standard (A) Bulk Regular
FY 1996 Costs
(thousands of dollars)

CS	Description	13	14	15	16	Total	Description	Value	Source
Othr	Remaining Costs	156	127	62	37	31,500	1) Total Attributable	4,164,788	Base Year CRA
							2) Total Distributed	4,133,288	Sum of Above
	- All the remaining attrib						3) = (1) - (2)	31,500	
	Total Costs	40,313	40,128	24,325	20,090	4,164,788			

Standard (A) Bulk Regular
FY 1996 Mail Volumes

(thousands of pieces)	13	14	15	16	Total		
Letters	811	693	240	176	19,204,390		LR-H-108, Appx A
Flats	55,903	38,834	34,290	22,830	10,274,743		LR-H-108, Appx A
Parcels	94,068	82,545	25,090	12,314	875,315		LR-H-108, Appx A
All Shapes	150,782	122,071	59,620	35,320	30,354,448		

Standard (A) Bulk Regular
FY 1996 Weight

(thousands of pounds)	13	14	15	16	Total		
Letters	635	577	213	170	1,170,975		LR-H-108, Appx A
Flats	43,515	32,551	30,960	21,978	2,370,910		LR-H-108, Appx A
Parcels	73,605	69,315	22,402	11,875	480,068		LR-H-108, Appx A
All Shapes	117,755	102,443	53,575	34,023	4,021,953		

Standard (A) Bulk Regular
FY 1996 Cubic Volume Es

(thousands of cubic feet)	13	14	15	16	Total		Density Estmates	
Letters	22	20	8	6	41,200	Letter	28.4219	LR-H-108
Flats	2,107	1,576	1,499	1,064	114,800	Flats	20.6526	LR-H-108
Parcels	8,998	8,474	2,739	1,452	58,688	Pardels	8.18	LR-H-108
All Shapes	11,127	10,070	4,245	2,522	214,687			

Standard (A) Bulk Regular
FY 1996 Unit Costs

(cents per piece)	13	14	15	16	Total
	26 7	32.9	40.8	56.9	13.7

Standard (A) Bulk Nonprofit Carrier-Route Mail
FY 1996 Costs
(thousands of dollars)

		Weight Increment (Ounces)											
CS	Description	1	2	3	4	5	6	7	8	9	10	11	12
3.1	Mail Processing	30,386	7,713	2,644	2,891	311	-	398	-	198	172	-	-
	- Mail processing includes variable mail processing costs, adjustment for premium pay, and piggybacked costs.												
3.1	Remote Endoding	-	-	-	-								
	- Remote encoding includes CS 3.1.1 and CS 16.3.6 costs, which are distributed to weight increments less than or equal to 4 ounces in proportion to letter-shaped pieces.												
3.2	Window Service	-	-	-	-	-	-	-	-	-	-	-	-
	- Window service is LIOCATT cost by weight increment multiplied by piggyback factor. See appendix B for documentation of LIOCATT costs by weight increment.												
6	City Carrier Casing	24,595	3,823	1,646	857	76	146	-	58	-	-	-	-
	- City carrier casing is LIOCATT cost multiplied by in-office support and piggyback factors. See appendix B for documentation of LIOCATT costs by weight increment												
7	City Carrier Street	20,493	5,645	1,971	903	366	135	146	22	10	6	1	1
	- City carrier street includes all CS 7 and piggybacked costs, distributed to weight by proportion of pieces.												
8	Vehicle Service Drivers	991	764	509	354	187	88	112	19	10	8	1	2
	- Vehicle service driver costs are distributed in proportion to estimated cubic volume.												
10	Rural Carriers	11,437	3,150	1,100	504	205	75	82	12	6	4	1	1
	- Rural carriers includes all CS 10 and piggybacked costs, distributed to weight by proportion of pieces.												
14	Air Transportation	87	60	37	23	12	6	7	1	1	0	0	0
	- Air and water transportation costs are distributed in proportion to estimated weight.												
14	Hwy Transportation	1,898	1,465	976	678	358	168	214	37	19	15	3	3
	- Highway and rail tranportation costs are distributed in proportion to estimated cubic volume.												

Standard (A) Bulk Nonprofit Carrier-Route Mail
FY 1996 Costs
(thousands of dollars)

		Weight Increment (Ounces)											
CS	Description	1	2	3	4	5	6	7	8	9	10	11	12
Othr	Remaining Costs	3,677	1,013	354	162	66	24	26	4	2	1	0	0

- All the remaining attributable costs are distributed in proportion to pieces.

	Total Costs	93,563	23,633	9,237	6,372	1,580	641	985	153	245	206	6	7

Standard (A) Bulk Nonprofit Carrier-Route Mail
FY 1996 Mail Volumes
(thousands of pieces)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
Letters	1,950,336	362,165	72,132	10,510	624	104	36	6	12	5	2	53
Flats	145,095	214,618	129,163	81,363	36,779	13,598	14,864	2,199	1,024	636	109	80
Parcels	68	433	240	484	71	56	41	21	3	18	2	-
All Shapes	2,095,499	577,217	201,535	92,357	37,475	13,758	14,942	2,226	1,039	658	113	134

Standard (A) Bulk Nonprofit Carrier-Route Mail
FY 1996 Weight
(thousands of pounds)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
Letters	66,642	30,607	10,779	2,142	172	35	15	3	6	3	1	37
Flats	6,133	19,677	20,065	17,427	10,070	4,715	6,060	1,021	530	373	72	57
Parcels	3	40	34	106	19	19	17	10	2	10	1	-
All Shapes	72,778	50,323	30,878	19,675	10,261	4,769	6,093	1,033	538	386	75	95

Standard (A) Bulk Nonprofit Carrier-Route Mail
FY 1996 Cubic Volume Estimat
(thousands of cubic feet)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
Letters	2,345	1,077	379	75	6	1	1	0	0	0	0	1
Flats	297	953	972	844	488	228	293	49	26	18	3	3
Parcels	1	9	8	24	4	4	4	2	0	2	0	-
All Shapes	2,642	2,039	1,359	943	498	234	298	52	26	20	4	4

Standard (A) Bulk Nonprofit Carrier-Route Mail
FY 1996 Unit Costs
(cents per piece)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
	4.5	4.1	4.6	6.9	4 2	4 7	6.6	6.9	23 6	31.4	5.5	5.1

Table 5

Standard (A) Bulk Nonprofit Ca
FY 1996 Costs
(thousands of dollars)

CS	Description	13	14	15	16	Total	Description	Value	Source
3.1	Mail Processing	-	-	-	-	44,713			Appendix A
	- Mail processing include								
3.1	Remote Endoding					-	WS 3.1.1	-	
							CS 16.3.6	-	Base Year CRA
	- Remote encoding inclu						REC Total	-	
3.2	Window Service	-	-	-	-	-	Piggyback	1.42261	LR-H-77
	- Window service is LIO								
6	City Carrier Casing	-	-	-	-	31,202	1) Support:	1.1736	LR-H-108
							2) Piggyback:	1.3062	LR-H-77
	- City carrier casing is LI						(1)*(2):	1.5329	
7	City Carrier Street	0	3	1	0	29,704	) Sum of CS 7 1 - 7	22,741	Base Year CRA
							2) Pigbk:	1.3062	LR-H-77
	- City carrier street inclu						3) = (1)*(2)	29,704	
8	Vehicle Service Drivers	1	4	2	0	3,051	1) C.S. 8	1,961	Base Year CRA
							2) Pigbk:	1 5558	LR-H-77
	- Vehicle service driver c						3) = (1)*(2)	3,051	
10	Rural Carriers	0	1	1	0	16,578	1) C.S. 10	13,834	Base Year CRA
							2) Pigbk	1.1983	LR-H-77
	- Rural carriers includes						3) = (1) * (2)	16,578	
14	Air Transportation	0	0	0	0	235	CS14 Air and Water	235	Base Year CRA
	- Air and water transport								
14	Hwy Transportation	2	8	4	0	5,847	CS 14 Highway and Rail	5,847	Base Year CRA
	- Highway and rail tranpo								

Standard (A) Bulk Nonprofit Ca
FY 1996 Costs
(thousands of dollars)

CS	Description	13	14	15	16	Total	Description	Value	Source
Othr	Remaining Costs	0	0	0	0	5,330	1) Total Attributable	136,659	Base Year CRA
							2) Total Distributed	131,329	Sum of Above
	- All the remaining attrib						3) = (1) - (2)	5,330	
	Total Costs	4	17	8	1	136,659			

Standard (A) Bulk Nonprofit Ca
FY 1996 Mail Volumes
(thousands of pieces)

	13	14	15	16	Total		
Letters	1	10	-	-	2,395,996		LR-H-108, Appx A
Flats	39	244	116	14	639,942		LR-H-108, Appx A
Parcels	8	5	-	-	1,451		LR-H-108, Appx A
All Shapes	48	259	116	14	3,037,389		

Standard (A) Bulk Nonprofit Ca
FY 1996 Weight
(thousands of pounds)

	13	14	15	16	Total		
Letters	1	8	-	-	110,451		LR-H-108, Appx A
Flats	30	206	106	13	86,554		LR-H-108, Appx A
Parcels	6	4	-	-	272		LR-H-108, Appx A
All Shapes	37	218	106	13	197,277		

Standard (A) Bulk Nonprofit Ca
FY 1996 Cubic Volume Estimat
(thousands of cubic feet)

	13	14	15	16	Total		Density Estmates	
Letters	0	0	-	-	3,886	Letter	28.4219	LR-H-108
Flats	1	10	5	1	4,191	Flats	20.6526	LR-H-108
Parcels	1	1	-	-	62	Parcels	4.4	LR-H-108
All Shapes	3	11	5	1	8,139			

Standard (A) Bulk Nonprofit Ca
FY 1996 Unit Costs
(cents per piece)

13	14	15	16	Total
8.5	6.5	6.6	6.9	4.5

Table 6

Standard (A) Bulk Nonprofit Other Mail
FY 1996 Costs
(thousands of dollars)

CS	Description	Weight Increment (Ounces) 1	2	3	4	5	6	7	8	9	10	11	12
3.1	Mail Processing	343,673	106,389	47,951	34,753	7,881	8,164	3,945	2,470	866	1,497	1,473	810
	- Mail processing includes variable mail processing costs, adjustment for premium pay, and piggybacked costs												
3.1	Remote Endoding	4,587	1,030	139	21								
	- Remote encoding includes CS 3 1.1 and CS 16.3.6 costs, which are distributed to weight increments less than or equal to 4 ounces in proportion to letter-shaped pieces.												
3 2	Window Service	1,445	811	-	-	-	-	-	-	-	-	-	-
	- Window service is LIOCATT cost by weight increment multiplied by piggyback factor. See appendix B for documentation of LIOCATT costs by weight increment												
6	City Carrier Casing	107,391	20,567	7,864	6,214	1,487	1,286	243	365	234	1,250	76	150
	- City carrier casing Is LIOCATT cost multiplied by in-office support and piggyback factors. See appendix B for documentation of LIOCATT costs by weight increment.												
7	City Carrier Street	69,822	21,147	5,275	2,550	1,335	628	489	274	153	118	78	58
	- City carrier street includes all CS 7 and piggybacked costs, distributed to weight by proportion of pieces.												
8	Vehicle Service Drivers	3,015	2,389	1,193	893	623	372	407	262	167	139	118	95
	- Vehicle service driver costs are distributed In proportion to estimated cubic volume.												
10	Rural Carriers	57,394	17,383	4,336	2,096	1,098	516	402	226	126	97	64	48
	- Rural darriers indludes all dS 10 and piggybadked dosts, distributed to weight by proportion of piedes.												
14	Air Transportation	2,340	1,677	758	519	347	201	187	120	76	66	48	39
	- Air and water transportation dosts are distributed in proportion to estimated weight.												
14	Hwy Transportation	13,343	10,573	5,281	3,951	2,758	1,647	1,799	1,158	738	615	521	421
	- Highway and rail tranportation costs are distributed in proportion to estimated cubic volume.												

Standard (A) Bulk Nonprofit Other Mail
FY 1996 Costs
(thousands of dollars)

		Weight Increment (Ounces)											
CS	Description	1	2	3	4	5	6	7	8	9	10	11	12
Othr	Remaining Costs	3,985	1,207	301	146	76	36	28	16	9	7	4	3

- All the remaining attributable costs are distributed in proportion to pieces.

Total Costs	606,994	183,173	73,097	51,142	15,606	12,851	7,499	4,890	2,368	3,789	2,384	1,624

Standard (A) Bulk Nonprofit Other Mail
FY 1996 Mail Volumes
(thousands of pieces)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
Letters	6,309,831	1,417,181	191,442	28,878	3,136	1,595	870	430	337	283	280	112
Flats	271,505	570,217	302,791	208,305	119,588	54,912	37,533	20,990	11,584	9,235	5,105	3,953
Parcels	1,175	6,277	3,047	3,210	3,156	2,708	7,710	4,448	2,505	1,606	2,002	1,445
All Shapes	6,582,512	1,993,675	497,280	240,393	125,881	59,215	46,113	25,868	14,426	11,124	7,387	5,510

Standard (A) Bulk Nonprofit Other Mail
FY 1996 Weight
(thousands of pounds)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
Letters	222,468	114,764	28,789	5,930	866	546	349	204	178	170	183	80
Flats	12,245	52,856	46,788	45,472	33,108	18,722	15,193	9,779	6,131	5,473	3,336	2,834
Parcels	57	656	466	694	888	936	3,177	2,049	1,331	960	1,318	1,027
All Shapes	234,770	168,276	76,043	52,095	34,862	20,204	18,719	12,032	7,640	6,602	4,837	3,941

Standard (A) Bulk Nonprofit Other Mail
FY 1996 Cubic Volume Es
(thousands of cubic feet)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
Letters	7,827	4,038	1,013	209	30	19	12	7	6	6	6	3
Flats	593	2,559	2,265	2,202	1,603	907	736	474	297	265	162	137
Parcels	7	80	57	85	109	114	388	251	163	117	161	126
All Shapes	8,427	6,677	3,335	2,495	1,742	1,040	1,136	731	466	388	329	266

Standard (A) Bulk Nonprofit Other Mail
FY 1996 Unit Costs
(cents per piece)

	Weight Increment (Ounces)											
	1	2	3	4	5	6	7	8	9	10	11	12
	9.2	9 2	14.7	21.3	12 4	21 7	16 3	18.9	16.4	34.1	32.3	29.5

Standard (A) Bulk Nonprof
FY 1996 Costs
(thousands of dollars)

CS	Description	13	14	15	16	Total	Description	Value	Source
3.1	Mail Processing	355	892	438	525	562,081			Appendix A
	- Mail processing include								
3 1	Remote Endoding					5,777	WS 3.1.1	4,777	
							CS 16.3.6	1,000	Base Year CRA
	- Remote encoding inclu						REC Total	5,777	
3.2	Window Service	-	-	-	-	2,255	Piggyback	1.42225	LR-H-77
	- Window service is LIO								
6	City Carrier Casing	76	162	-	75	147,441	1) Support.	1.1736	LR-H-108
							2) Piggyback:	1.3189	LR-H-77
	- City carrier casing is LI						(1)*(2):	1.5478	
7	City Carrier Street	40	56	34	22	102,080	) Sum of CS 7.1 - 7	77,397	Base Year CRA
							2) Pigbk.	1.3189	LR-H-77
	- City carrier street inclu						3) = (1)*(2)	102,080	
8	Vehicle Service Drivers	66	124	73	54	9,988	1) C S. 8	6,426	Base Year CRA
							2) Pigbk:	1.5544	LR-H-77
	- Vehicle service driver c						3) = (1)*(2)	9,988	
10	Rural Carriers	33	46	28	18	83,910	1) C.S. 10	70,010	Base Year CRA
							2) Pigbk	1.1986	LR-H-77
	- Rural darriers indludes						3) = (1) * (2)	83,910	
14	Air Transportation	29	44	29	20	6,501	CS14 Air and Water	6,501	Base Year CRA
	- Air and water transport								
14	Hwy Transportation	291	549	323	238	44,205	CS 14 Highway and Rail	44,205	Base Year CRA
	- Highway and rail tranpo								

Standard (A) Bulk Nonprof
FY 1996 Costs
(thousands of dollars)

CS	Description	13	14	15	16	Total	Description	Value	Source
Othr	Remaining Costs	2	3	2	1	5,826	1) Total Attributable	970,066	Base Year CRA
							2) Total Distributed	964,240	Sum of Above
	- All the remaining attrib						3) = (1) - (2)	5,826	
	Total Costs	891	1,875	929	954	970,066			

Standard (A) Bulk Nonprof
FY 1996 Mail Volumes

(thousands of pieces)	13	14	15	16	Total		
Letters	57	67	62	32	7,954,594		LR-H-108, Appx A
Flats	2,952	3,037	2,251	1,342	1,625,302		LR-H-108, Appx A
Parcels	738	2,145	933	726	43,833		LR-H-108, Appx A
All Shapes	3,747	5,249	3,247	2,100	9,623,728		

Standard (A) Bulk Nonprof
FY 1996 Weight

(thousands of pounds)	13	14	15	16	Total		
Letters	44	57	57	31	374,715		LR-H-108, Appx A
Flats	2,306	2,561	2,036	1,297	260,138		LR-H-108, Appx A
Parcels	576	1,804	848	706	17,494		LR-H-108, Appx A
All Shapes	2,927	4,422	2,941	2,035	652,347		

Standard (A) Bulk Nonprof
FY 1996 Cubic Volume Es

(thousands of cubic feet)	13	14	15	16	Total		Density Estmates	
Letters	2	2	2	1	13,184	Letter	28 4219	LR-H-108
Flats	112	124	99	63	12,596	Flats	20.6526	LR-H-108
Parcels	70	221	104	86	2,139	Parcels	8.18	LR-H-108
All Shapes	184	347	204	150	27,919			

Standard (A) Bulk Nonprof
FY 1996 Unit Costs

(cents per piece)	13	14	15	16	Total
	23.8	35.7	28.6	45.4	10.1

Appendix A

Mail Processing Cost Computer Documentation

Program Documentation

I. Assign IOCS tallies to mail processing cost pools.

A. Program : cadoc22c_w.f - This program uses various IOCS fields to assign mail processing tallies to cost pool.

B. Input files:
   1. IOCS file extract - supplied in LR-H-23.
   2. fins.dat - map from finance number to ROG code This file is being provided in LR-H146 for data security reasons.

C. Output files
   1. mods12_mp96_cw.dat2 - Mail processing tallies in MODs 1 & 2 cost pools.
   2. bmcs_mp96_cw.dat2 - Mail processing tallies in BMC cost pools
   3. nonmods_mp96_cw.dat2 - Mail processing tallies in the non-MODs cost pool.

II. Develop direct tally Standard (A) costs for MODs 1 & 2 cost pools.

A. Program: windxmod.f - Adds up Standard (A) cost by cost pool, activity code, and weight increment.

B. Input files:
   1. mods12_mp96_cw.dat2
   2. activity.3c - list of Standard (A) activity codes.

C. Output files:
   1. modwind.csv - file to be imported into Excel.

III. Develop direct tally Standard (A) costs for BMC cost pools.

A. Program: windxbmc.f - Adds up Standard (A) cost by cost pool, activity code, and weight increment.

B. Input files:
   1. bmcs_mp96_cw.dat2
   2. activity.3c - list of Standard (A) activity codes.

C. Output files:
   1. bmcwind.csv - file to be imported into Excel.

IV. Develop direct tally Standard (A) costs for the non-MODs cost pool.

A. Program: windxnmod.f - Adds up Standard (A) cost by cost pool, activity code, and weight increment.

B. Input files:
   1. nonmods_mp96_cw.dat2
   2. activity.3c - list of Standard (A) activity codes

C. Output files:

1. nmodwind.csv - file to be imported into Excel.

V. Summarize results in Excel spreadsheets.

A. Files: stdampwt.xls (regular) and sanpmpwt.xls (nonprofit)
   1. Sheet modwind: This sheet contains the file produced by windxmod.f. One column was added to the left of the text file. This contain a cost pool and subclass index code for each row. This is used by the sumif() functions in sheets BCL through BRP to summarize the data in the proper place.
   2. Sheet bmcwind: Similar to sheet modwind, this contains the corresponding data for BMC cost pools.
   3. Sheet nmodwind: Similar to sheet modwind, this contains the corresponding data for the non-MODs cost pool.
   4. Sheet poolmaps: This sheet contain maps from the cost pool numbering used in the Fortran code to the cost pool number in the spreadsheet.
   5. Sheet maps: This sheet contains a map from activity code to subclass code.
   6. Sheet BCL, BCF, BCP, BRL, BRF, BRP (Bxy x:{C- carrier route, R - other}, y: {L - letters, F - flats, P - parcels}) Each of these sheets contains the variable costs by cost pool including piggybacks from library reference H-106 in column D. In columns E through T is the summarized direct tally cost from the modwind, bmcwind, and nmodwind sheets. In columns U though AJ is the variable costs spread to weight increments in proportion to the direct costs in columns E through T. At the bottom of each sheet the variable cost distributed to weight increment is summed over cost pools
   7. Sheet summary: For each shape and rate category the variable mail processing cost distributed to weight increment is shown.

B. Files: stdawght.xls and stdafwt.xls contain the tables for the all shapes and flat mail analyses respectively.

Appendix B

Computer Documentation for Window Service and City Carrier In-Office Estimates

To estimate window service costs and city carrier in-office costs by weight increment IOCS direct tallies were marked with a weight increment index. Once this had been done, the LIOCATT procedure was run to distribute mixed-mail costs to direct costs. The results presented in this library reference are the sum of direct and distributed mixed-mail costs.

Program Documentation

I. Read IOCS tally file and encode fields as integer indexes. Read weight of sampled piece fields and code as integer index.

- A. Program : encode_wi.f - This program converts character fields into integer index fields.
- B. Input files:
  1. IOCS file extract - LR-H-23
  2. activity.s - sorted list of IOCS activity codes.
  3. fincag.s - sorted list of tally CAG and tally finance numbers
- C. Output files:
  1. encdata - file of encoded IOCS tally information
  2. encode.out - diagnostic output

II. Sort encoded IOCS tally information to correct order for LIOCATT process.

- A. Program: encdata.sm
- B. Input files:
  1. encdata
- C Output files:
  1. encdata.s - sorted file of encoded IOCS tally information
  2. list.encdata.sm - diagnostic output

III. Distribute mixed mail tally costs to direct tallies (LIOCATT process).

- A. Program: liocatt.f
- B. Subroutines:
  1. loaddata.f
  2. fillmixmap.f
  3. fungroup.f
  4. sortcost.f
  5. level1.f
  6. level2.f
  7. sortlev2a.f
  8. level3.f
  9. report.f
- C. Input files:

1. encdata.s
2. activity.s
3. mxmail.dat - map of distribution keys from mixed mail activity codes to direct activity codes.
4. mmcodes - list of mixed mail activity codes
5. operrtemap - map from mail processing operation codes to functional groups.

D. Output files:
1. level1b - Level 1 distributed mixed mail costs
2. level2b - Level 2 distributed mixed mail costs
3. level3b - Level 3 distributed mixed mail costs
4. level3a - Direct mail costs

IV. Sum direct and distributed mixed mail costs for clerk and mail handler functional groups by activity code.

A. Program: rptwi3c.f

B. Input files:
1. level?? - Level 1-3 distributed mixed and direct mail costs
2. activity.s
3. map.3c - List of Standard (A) activity codes.

C. Output files:
1. rwi3c96.csv - clerk and mail handler costs summed by activity code and functional group

V. Put results into spreadsheet for final table.

A. Spreadsheet: Stdawght.xls - Costs are loaded into sheet rwi3c96.

Appendix C

Program Source Codes

```
      Program cadoc22c_std

       Purpose:  To generate IOCS cost pool groups including admin/windows

      IMPLICIT NONE

      integer*4    keep, hand, type, bmcgrp, ier, ct, i1, i2, i3, i4, i5
      integer*4    fins, costpool, searchc, i,  x, y, ipool, route, ibmc
      integer*4    ld, rd, inonmod, ct0
      parameter    (fins=32685)
      parameter    (ipool=59)
      parameter    (ibmc=41)
      parameter    (inonmod=30)

      integer*4    pool          !  function to assign cost pool group
      integer*4    q19           !  function to use Q19 to assign invalid modes codes
      integer*4    type1         !  function for manual type for Q19
      integer*4    type2         !  funciton for transportation type for Q19
      integer*4    rog(fins)/fins*0/,cpool(ipool)/ipool*0/,group(ibmc)/ibmc*0/
      integer*4    count(ipool)/ipool*0/, countb(ibmc)/ibmc*0/
      integer*4    countn(inonmod)/inonmod*0/,groupn(inonmod)/inonmod*0/
      integer*4    ct1/0/, ct2/0/, ct3/0/, ctaw1/0/, ctmp1/0/, ctaw2/0/
      integer*4    ctmp2/0/, ctaw3/0/, ctmp3/0/, ctkeep/0/, cthand/0/,ctnonh/0/
      integer*4    f262, f260, f257, iw, actv, f244, i6, sort

      real*8       f9250, wgt, dlrs
      real*8       doll(ipool)/ipool*0.0/,dollb(ibmc)/ibmc*0.0/
      real*8       dolln(inonmod)/inonmod*0.0/

      character*263 rec
      character*6  fin(fins)/fins*' '/, f2
      character*1  f1,f7,f116,f118,f119,f121,f122,f9209,f128,f9211,f9212
      character*1  f9214,f9219,f9602,f117,f155,f156,f157,f9606,f9632
      character*3  f114,f246,f247,f248,f249
      character*4  cf244

c     read finance numbers map

      open(10,file='fins.dat',iostat=ier)

11    format(a6,i2)
      do i=1,fins
         read(10,11) fin(i), rog(i)
      end do

      close(10)

21    format(a263)
      open(30,file='mods12_mp96_cw.dat2',recl=300,iointent='output')
      open(35,file='mods12_aw96_cw.dat2',recl=300,iointent='output')
31    format(a263,f15.5,i2,i1,i3,i5)
32    format(a263,f15.5,i2,i1,i1,i3,i5)
      open(40,file='bmcs_mp96_cw.dat2',recl=300,iointent='output')
      open(45,file='bmcs_aw96_cw.dat2',recl=300,iointent='output')
41    format(a263,f15.5,i2,i3,i5)
      open(50,file='nonmods_mp96_cw.dat2',recl=300,iointent='output')
      open(55,file='nonmods_aw96_cw.dat2',recl=300,iointent='output')
      open(80,file='iocs_bad.dat',recl=300,iointent='output')

51    format(a263,f15.5,i1,i3,i5)

      ier=0
      ct=0
      ct0=0
      i1=0
      i2=0
      i3=0
      i4=0
      i5=0
330   format(a1,1x,a6,9x,a1,13x,a3,4a1,4x,2a1,4x,a1,
     +   2x,4a1,2x,a1,4x,a1,73x,a1,33x,4a1,35x,i4,4a3,i2,i2,1x,i4,11x,f10.0)

99    do while (ier.eq.0)
         keep=0
         hand=0
         type=0
         bmcgrp=0
         read(5,21,iostat=ier,end=100) rec
         if (rec(228:229).eq.'$$') then
            write (80,21) rec
```

```
      ct0 = ct + 1
      print *, "$$ in record ",ct0
      rec(228:229) = '  '
      goto 99
    end if
    iw = 1
    read(rec,330) f1, f2, f7, f114, f116, f117, f118, f119, f121, f122,
+      f9209, f128, f9211, f9212, f9602, f9214, f9219, f9606, f9632, f155,
+      f156, f157, f244, f246, f247, f248, f249, f257, f260, f262, f9250

    cf244 = rec(210:213)

    ct=ct+1
    i1=searchc(fin,fins,f2)
    if ((f257.eq.11).or.(f257.eq.12).or.
+      (f257.eq.31).or.(f257.eq.32).or.(f257.eq.41)
+      .or.(f257.eq.42).or.(f257.eq.61).or.
+      (f257.eq.62).or.(f257.eq.81).or.(f257.eq.82))
+      then
       keep=1
    end if

    if (f9250.le.0.0) then
       keep=0
    end if

    if (f7.eq.'K') then
       keep=0
    end if

    if (keep.eq.1) then
       ctkeep=ctkeep+1
    end if

    wgt=f9250/100000

    if (keep.eq.1) then
       if (rec(235:241).eq.'666666A') then !  BMCs
          type=1
          ct1=ct1+1
       else if (((f262.ge.10).and.(f262.le.4950)).or.
+           ((f9219.ge.'A').and.(f9219.le.'J')).or.
+           ((f9214.ge.'A').and.(f9214.le.'P'))) then
          hand=1
          cthand=cthand+1
       else
          hand=2
          ctnonh=ctnonh+1
       end if
       if ((hand.gt.0).and.(i1.gt.0)) then
          if ((rog(i1).eq.1).or.(rog(i1).eq.2)) then
             type=2
             ct2=ct2+1
          else
             type=3
             ct3=ct3+1
          end if
       end if

       if ((hand.gt.0).and.(i1.eq.0)) then
          type=3
          ct3=ct3+1
       end if

       if (type.eq.2) then
          i2=pool(f114)
          i3=q19(f128)
          i4=type1(f9211,f9602)
          i5=type2(f9212)
          i6=sort(f9602)

          if (i2.eq.1) then
             if (f1.eq.'1') then
                if ((i3.eq.1).and.(i4.eq.1)) then
                   i2=2
                else if ((i3.eq.1).and.(i4.eq.2)) then
                   i2=3
                else if ((i3.eq.1).and.(i4.eq.3)) then
                   i2=4
                else if ((i3.eq.1).and.(i4.eq.4)) then
```

```
      i2=36
    else if ((i3.eq.1).and.(i4.eq.5)) then
      i2=35
    else if ((i3.eq.1).and.(i4.eq.6)) then
      i2=30
    else if ((i3.eq.1).and.(i4.eq.7)) then
      i2=32
    else if ((i3.eq.1).and.(i4.eq.8)) then
      i2=29
    else if (i3.eq.2) then
      i2=10
    else if ((i3.ge.3).and.(i3.le.5)) then
      i2=11
    else if (i3.eq.6) then
      i2=8
    else if ((i3.ge.7).and.(i3.le.8)) then
      i2=36
    else if (i3.eq.9) then
      i2=34
    else if (i3.eq.10) then
      i2=5
    else if (i3.eq.11) then
      i2=9
    else if (i3.eq.12) then
      i2=6
    else if (i3.eq.13) then
      i2=34
    else if (i3.eq.14) then
      i2=33
    else if (i3.eq.15) then
      i2=30
    else if (i3.eq.16) then
      i2=37
    else if (i3.eq.17) then
      i2=32
    else if (i3.eq.18) then
      i2=36
    else if (i3.eq.19) then
      i2=17
    else if ((i3.eq.20).and.((i5.ge.1).and.(i5.le.4))) then
      i2=29
    else if (f260.eq.7) then
      i2=26
    else if (f260.eq.8) then
      i2=29
    else if ((f118.eq.'A').or.(f118.eq.'C').or.
+         (f118.eq.'E').or.(f118.eq.'F').or.
+         (f118.eq.'I').or.(f118.eq.'K')) then
      i2=36
    else if ((f118.eq.'B').or.(f118.eq.'D').or.
+         (f118.eq.'H').or.(f118.eq.'J')) then
      i2=30
    else if (f118.eq.'G') then
      i2=19
    else if (((f119.ge.'A').and.(f119.le.'F')).and.
+         (f122.eq.' ').and.(i6.eq.1)) then
      i2=4
    else if (((f119.ge.'A').and.(f119.le.'F')).and.
+         (f122.eq.' ').and.(i6.eq.3)) then
      i2=33
    else if (((f119.ge.'A').and.(f119.le.'F')).and.
+         (f122.eq.' ').and.(i6.eq.4)) then
      i2=4
    else if (((f119.ge.'A').and.(f119.le.'G')).and.
+         (f122.eq.' ').and.((i3.eq.1).and.(i4.eq.9))) then
      i2=42
    else if (((f119.ge.'A').and.(f119.le.'G')).and.
+         (f122.eq.' ').and.(i6.eq.2)) then
      i2=32
    else if ((f122.ge.'A').and.(f122.le.'F')) then
      i2=32
    else if ((f122.ge.'I').and.(f122.le.'L')) then
      i2=32
    else if (f122.eq.'G') then
      i2=42
    else if (f122.eq.'M') then
      i2=42
    else if (f122.eq.'H') then
      i2=38
    else if (f260.eq.0) then
```

```
                  i2=18
               else if (f260.eq.6) then
                  i2=40
               else if (f260.eq.9) then
                  i2=39
               else if (f260.eq.10) then
                  i2=58
               else if (f260.eq.14) then
                  i2=25
               else if (f260.eq.17) then
                  i2=45
               else if (f260.eq.18) then
                  i2=16
               else if (f260.eq.19) then
                  i2=20
               else if (f260.eq.20) then
                  i2=40
               else if (f260.eq.21) then
                  i2=24
               else if (f260.eq.22) then
                  i2=15
               else if (f260.eq.23) then
                  i2=24
               else if ((f260.ge.24).and.(f260.le.26)) then
                  i2=39
               else
                  i2=42
               end if
            end if

            if (f1.eq.'4') then
               i2 = 59
               if ((i3.ge.2).and.(i3.le.5)) then
                  i2=12
               else if (i3.eq.6) then
                  i2=13
               else if (i3.eq.11) then
                  i2=13
               else if (f260.eq.0) then
                  i2=23
               else if (f260.eq.6) then
                  i2=23
               else if ((f260.ge.11).and.(f260.le.13).or.
     +                (f260.eq.20)) then
                  i2=27
               else if ((f260.eq.9).or.((f260.ge.24).and.
     +                (f260.le.26))) then
                  i2=39
               else if (f260.eq.10) then
                  i2=59
               else if (f260.eq.14) then
                  i2=25
               else if (f260.eq.17) then
                  i2=45
               else if (f260.eq.18) then
                  i2=23
               else if (f260.eq.19) then
                  i2=20
               else if (f260.eq.21) then
                  i2=23
               else if (f260.eq.22) then
                  i2=21
               else if (f260.eq.23) then
                  i2=23
               else
                  i2=28
               end if
            end if
         end if

c     MODS-based encirclement rule

            if (((f262.ge.10).and.(f262.le.300))
     +         .and.(f9214.eq.' ')) then
               if (f262.eq.60) then
                  actv = f262
               else if (f262.eq.190) then
                  if ((f155.eq.'1').or.(f156.eq.'1').or.(f9606.eq.'A')
     +                .or.(f9606.eq.'B')) then
                     actv = f262
```

```
               else if (((f262.eq.190).and.(cf244(2:4).eq.'510')) then
                  actv = f262
               else if (((f246.eq.'   ').or.(f247.eq.'   ').or.
     +               (f248.eq.'   ').or.(f249.eq.'   ')).and.

     +               ((i2.eq.26).or.(i2.eq.39).or.(i2.eq.28)

     +               .or.(i2.eq.24).or.(i2.eq.23).or.(i2.eq.22)
     +               .or.(i2.eq.40).or.(i2.eq.41).or.(i2.eq.42)
     +               .or.(i2.ge.44))) then
                  actv = f262
               end if
            else if ((f262.eq.300).and.
     +               ((f157.eq.'1').or.(f9606.eq.'C')
     +               .or.(f9632.eq.'1'))) then
               actv = f262
            else if ((f246.eq.'   ').and.(f247.eq.'   ').and.
     +               (f248.eq.'   ').and.(f249.eq.'   ')) then
               actv = f244
               if (f262.eq.210) then
                  actv = f262
               else if ((f262.eq.90).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.35).or.(i2.eq.37).or.(i2.eq.32)
     +                  .or.(i2.eq.36).or.(i2.eq.30).or.(i2.eq.31)
     +                  .or.(i2.eq.33).or.(i2.eq.42).or.(i2.eq.40)
     +                  .or.(i2.eq.41).or.(i2.eq.28).or.
     +                  (i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if (((f262.eq.30).or.(f262.eq.70).or.
     +                  (f262.eq.80)).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.42).or.(i2.eq.41)
     +                  .or.(i2.eq.40).or.(i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if ((f262.eq.10).and.
     +                  ((i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.21).or.(i2.eq.15).or.(i2.eq.42)
     +                  .or.(i2.eq.40).or.(i2.eq.41))) then
                  actv = f262
               else if ((f262.eq.50).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.42).or.(i2.eq.41)
     +                  .or.(i2.eq.40).or.(i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if ((f262.eq.20).and.
     +                  ((i2.eq.35).or.(i2.eq.37).or.(i2.eq.32)
     +                  .or.(i2.eq.36).or.(i2.eq.30).or.(i2.eq.31)
     +                  .or.(i2.eq.33).or.(i2.eq.26).or.(i2.eq.29)
     +                  .or.(i2.eq.28).or.(i2.eq.39))) then
                  actv = f262
               end if
            else if ((f246.gt.'001').or.(f247.gt.'001')
     +               .or.(f248.gt.'001').or.(f249.gt.'001')) then
               actv = f244
               if (((f262.eq.30).or.(f262.eq.70).or.
     +                  (f262.eq.80)).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.42).or.(i2.eq.41)
     +                  .or.(i2.eq.40).or.(i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if ((f262.eq.10).and.
     +                  ((i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.21).or.(i2.eq.15).or.(i2.eq.42)
     +                  .or.(i2.eq.40).or.(i2.eq.41))) then
                  actv = f262
               else if ((f262.eq.50).and.
     +                  ((i2.eq.18).or.(i2.eq.24).or.(i2.eq.23)
     +                  .or.(i2.eq.26).or.(i2.eq.29).or.(i2.eq.28)
     +                  .or.(i2.eq.39).or.(i2.eq.42).or.(i2.eq.41)
     +                  .or.(i2.eq.40).or.(i2.eq.22).or.(i2.ge.44))) then
                  actv = f262
               else if ((f262.eq.20).and.
     +                  ((i2.eq.35).or.(i2.eq.37).or.(i2.eq.32)
     +                  .or.(i2.eq.36).or.(i2.eq.30).or.(i2.eq.31)
```

```
+                    .or.(i2.eq.33).or.(i2.eq.26).or.(i2.eq.29)
+                    .or.(i2.eq.28).or.(i2.eq.39))) then
                  actv = f262
               end if
            end if
         else
            actv = f262
         end if


         if ((i2.eq.39).or.(i2.ge.44)) then
            if (i2.eq.39) then
               dlrs=wgt*634148939/728038888
            else if (i2.eq.49) then
               dlrs=0.0
            else
               dlrs=wgt*680668580/864658254
            end if
            if (i2.eq.39) then
               costpool=1
            else
               costpool=2
            end if
            write(35,32) rec, dlrs, i2, hand, costpool, iw, actv
            ctaw2=ctaw2+1
         else
            write(30,31) rec, wgt, i2, hand, iw, actv
            ctmp2=ctmp2+1
         end if
         if (i2.le.59) then
            if ((i2.eq.39).or.(i2.ge.44)) then
               doll(i2)=doll(i2)+dlrs
            else
               doll(i2)=doll(i2)+wgt
            end if
            cpool(i2)=i2
            count(i2)=count(i2)+1
         end if
      end if               ! type 2

      if (type.eq.1) then
         if (((f260.ge.0).and.(f260.le.8)).or.
+           ((f260.ge.11).and.(f260.le.16)).or.
+           ((f260.ge.18).and.(f260.le.23)).or.
+           ((f260.ge.27).and.(f260.le.29))) then
            if ((f128.eq.'I').and.(f121.eq.'N')) then
               bmcgrp=31
            else if ((f128.eq.'I').and.(f121.eq.'Y')) then
               bmcgrp=32
            else if ((f128.eq.'J').and.(f121.eq.'N')) then
               bmcgrp=33
            else if ((f128.eq.'J').and.(f121.eq.'Y')) then
               bmcgrp=34
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                 (f128.eq.'L')) then
               bmcgrp=35
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                 (f9602.eq.'A')) then
               bmcgrp=35
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                 (f9602.eq.'B')) then
               bmcgrp=35
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                 (f128.eq.'M')) then
               bmcgrp=36
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                 (f9211.eq.'C').and.(f9602.eq.'C')) then
               bmcgrp=36
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
+                 (f9211.eq.'C').and.(f9602.eq.'D')) then
               bmcgrp=36
            else if (((f116.ge.'A').and.(f116.le.'H')).and.
+                 (f9209.eq.' ')) then
               bmcgrp=37
            else if (((f118.ge.'A').and.(f118.le.'K')).and.
+                 (f9209.eq.' ')) then
               bmcgrp=38
            else if (f9209.eq.' ') then
               bmcgrp=39
            else if ((f9209.ge.'A').and.(f9209.le.'F')) then
```

```
                bmcgrp=40
             end if
          else
             bmcgrp=f260
          end if

          if (bmcgrp.eq.0) then
             bmcgrp=41
          end if
          if (bmcgrp.le.30) then
             dlrs=wgt* 1.015935996
             write(45,41) rec, dlrs, bmcgrp, iw, actv
             ctaw1=ctaw1+1
          else
             dlrs=wgt * 1.015935996
             write(40,41) rec, dlrs, bmcgrp, iw, actv
             ctmp1=ctmp1+1
          end if
          dollb(bmcgrp)=dollb(bmcgrp)+dlrs
          group(bmcgrp)=bmcgrp
          countb(bmcgrp)=countb(bmcgrp)+1
       end if

       if (type.eq.3) then
          if (f260.eq.0) then
             f260=30
          end if
          if (((f260.ge.9).and.(f260.le.10)).or.
     +       (f260.eq.17).or.((f260.ge.24).and.
     +       (f260.le.26))) then
             dlrs=wgt * 1.051561404
             write(55,51) rec, dlrs, hand, iw, actv
             ctaw3=ctaw3+1
          else
             dlrs=wgt * 1.051561404
             write(50,51) rec, dlrs, hand, iw, actv
             ctmp3=ctmp3+1
          end if
          if (f260.gt.0) then
             dolln(f260)=dolln(f260)+dlrs
             groupn(f260)=f260
             countn(f260)=countn(f260)+1
          end if
       end if              !type.eq.3
    end if                 !keep

 end do

00 print*, "ier is ", ier
 print*, "Total Records ", ct
 print*, "Number of obs kept ", ctkeep
 print*, "BMC  Total Obs ", ct1, " BMC  Adm/Win ", ctaw1,
+   " BMC  MP ", ctmp1
 print *, "Number of handling ", cthand
 print *, "Number of non-handling ", ctnonh
 print*, "MODS Total Obs ", ct2, " MODS Adm/Win ", ctaw2,
+   " MODS MP ", ctmp2
 print*, "NMOD Total Obs ", ct3, " NMOD Adm/Win ", ctaw3,
+   " NMOD MP ", ctmp3


 open(60,file='mods12.dat',iointent='output')
 open(70,file='bmc.dat',iointent='output')
 open(75,file='nonmod.dat',iointent='output')
31 format(i3,f15.5,i10)
71 format(i3,f15.5,i10)

 do i=1,ipool
    write(60,61) cpool(i), doll(i), count(i)
 end do

 do i=1, ibmc
    if (countb(i).gt.0) then
       write(70,71) group(i), dollb(i), countb(i)
    end if
 end do

 do i=1, inonmod
    if (countn(i).gt.0) then
       write(75,71) groupn(i), dolln(i), countn(i)
```

```
      end if
    end do

    end

    ------------------------------------------------------------------
     assigns pool groups to MODS numbers

    function   pool(mod)

    integer*4    pool
    character*3  mod

    pool = 0

     OCR OPERATIONS
    if (((mod.ge.'830').and.(mod.le.'837')).or.
   +   ((mod.ge.'840').and.(mod.le.'847')).or.
   +   ((mod.ge.'850').and.(mod.le.'857')).or.
   +   ((mod.ge.'880').and.(mod.le.'887'))) then
       pool = 10


c     BCS OPERATIONS
    else if ((mod.eq.'292').or.(mod.eq.'295').or.(mod.eq.'299').or.
   +      ((mod.ge.'860').and.(mod.le.'869')).or.
   +      ((mod.ge.'870').and.(mod.le.'879')).or.
   +      ((mod.ge.'890').and.(mod.le.'899')).or.
   +      ((mod.ge.'910').and.(mod.le.'911')).or.
   +      ((mod.ge.'914').and.(mod.le.'919')).or.
   +      ((mod.ge.'970').and.(mod.le.'979'))) then
       pool=11

     LSM OPERATIONS
    else if (((mod.ge.'080').and.(mod.le.'089')).or.
   +      (mod.eq.'091').or.
   +      ((mod.ge.'093').and.(mod.le.'099'))) then
       pool=8

     FSM OPERATIONS
    else if (((mod.ge.'190').and.(mod.le.'191')).or.
   +      ((mod.ge.'194').and.(mod.le.'197')).or.
   +      ((mod.ge.'140').and.(mod.le.'148')).or.
   +      ((mod.ge.'441').and.(mod.le.'444')).or.
   +      (mod.eq.'446').or.(mod.eq.'448').or.
   +      ((mod.ge.'960').and.(mod.le.'967'))) then
       pool=9

     Mechanized sort-sack outside
    else if ((mod.ge.'238').and.(mod.le.'239')) then
       pool=34

c     MECHANIZED PARCEL SORTER
    else if ((mod.ge.'105').and.(mod.le.'106')) then
       pool=5

c     SMALL PARCEL BUNDLE SORTER
    else if ((mod.ge.'134').and.(mod.le.'137')) then
       pool=6
    else if ((mod.ge.'138').and.(mod.le.'139')) then
       pool=7

c     MANUAL FLAT OPERATIONS
    else if (((mod.ge.'060').and.(mod.le.'061')).or.
   +      ((mod.ge.'064').and.(mod.le.'079')).or.
   +      ((mod.ge.'170').and.(mod.le.'179'))) then
       pool=3

c     MANUAL LETTERS OPERATIONS
    else if (((mod.ge.'029').and.(mod.le.'031')).or.
   +      ((mod.ge.'034').and.(mod.le.'038')).or.
   +      ((mod.ge.'040').and.(mod.le.'049')).or.
   +      ((mod.ge.'150').and.(mod.le.'159')).or.
   +      ((mod.ge.'160').and.(mod.le.'169'))) then
       pool=2

c     MANUAL PARCEL OPERATIONS
    else if (((mod.ge.'100').and.(mod.le.'101')).or.
   +      (mod.eq.'104').or.(mod.eq.'130').or.
   +      ((mod.ge.'200').and.(mod.le.'207'))) then
```

```
      pool=4

     MANUAL PRIORITY
     else if ((mod.ge.'050').and.(mod.le.'059')) then
      pool=14

     LDC15
     else if ((mod.eq.'771').or.((mod.ge.'774').and.(mod.le.'776')).or.
    +     (mod.eq.'779')) then
      pool=17

     ALLIED OPERATIONS

     ACDCS
     else if ((mod.ge.'118').and.(mod.le.'119')) then
      pool=37

     Bulk presort
     else if ((mod.ge.'002').and.(mod.le.'009')) then
      pool=35

     Cancellation/mail prep
     else if (((mod.ge.'010').and.(mod.le.'019')).or.
    +   ((mod.ge.'020').and.(mod.le.'028'))) then
      pool=36

     manual sack sort
     else if ((mod.ge.'235').and.(mod.le.'237')) then
      pool=33

     opening unit - pref
     else if (((mod.ge.'110').and.(mod.le.'114')).or.
    +   ((mod.ge.'180').and.(mod.le.'184'))) then
      pool=30

     opening unit - bbm
     else if (((mod.ge.'115').and.(mod.le.'117')).or.
    +   ((mod.ge.'185').and.(mod.le.'189'))) then
      pool=31

     platform
     else if ((mod.ge.'210').and.(mod.le.'234')) then
      pool=29

     pouching
     else if (((mod.ge.'120').and.(mod.le.'129')).or.
    +   ((mod.ge.'208').and.(mod.le.'209'))) then
      pool=32

     BUSINESS REPLY / POSTAGE DUE
     else if (mod.eq.'930') then
      pool=18

     DAMAGED PARCEL REWRAP
     else if (mod.eq.'109') then
      pool=19

     empty equipment
     else if (mod.eq.'549') then
      pool=38

     EXPRESS
     else if ((mod.eq.'131').or.(mod.eq.'669').or.(mod.eq.'793')) then
      pool=15

     MAILGRAM
     else if (mod.eq.'584') then
      pool=20

     MAIL PROCESSING SUPPORT
     else if (((mod.ge.'340').and.(mod.le.'341')).or.
    +     ((mod.ge.'554').and.(mod.le.'555')).or.
    +     (mod.eq.'547').or.(mod.eq.'548').or.(mod.eq.'607').or.
    +     (mod.eq.'612').or.(mod.eq.'620').or.(mod.eq.'625').or.
    +     (mod.eq.'630').or.(mod.eq.'677').or.(mod.eq.'755').or.
    +     (mod.eq.'798')) then
      pool=40

     MISCELLANEOUS
     else if ((mod.ge.'560').and.(mod.le.'564')) then
```

```
          pool=42

         REGISTRY
        else if ((mod.ge.'585').and.(mod.le.'590')) then
          pool=16

         LDC41 AND LDC42
        else if (((mod.ge.'821').and.(mod.le.'829')).or.
     +       ((mod.ge.'905').and.(mod.le.'906')).or.
     +       ((mod.ge.'912').and.(mod.le.'913'))) then
          pool=12
        else if ((mod.ge.'801').and.(mod.le.'819')) then
          pool=13

c       MANUAL DISTRIBUTION - STATION/BRANCH (LDC43)
        else if ((mod.ge.'240').and.(mod.le.'339')) then
          pool=28

c       STATION/BRANCH - BOX SECTION (LDC44)
        else if (mod.eq.'769') then
          pool=27

c       WINDOW Service
        else if (((mod.ge.'355').and.(mod.le.'453')).or.(mod.eq.'568')) then
          pool=39

c       LDC48
        else if (mod.eq.'583') then
          pool=21
        else if ((mod.eq.'353').or.(mod.eq.'558').or.(mod.eq.'559').or.
     +       (mod.eq.'608').or.(mod.eq.'621').or.(mod.eq.'626').or.
     +       (mod.eq.'631').or.(mod.eq.'678')) then
          pool=22
        else if ((mod.ge.'542').and.(mod.le.'544')) then
          pool=23
        else if ((mod.eq.'741').or.(mod.eq.'742').or.(mod.eq.'794')) then
          pool=24

c       ADDRESS INFO SYSTEM & CENTRAL MAIL MARK-UP
        else if ((mod.eq.'539').or.((mod.ge.'795').and.(mod.le.'797'))) then
          pool=25

c       MAILING REQUIREMENTS & BUSINESS MAIL ENTRY
        else if ((mod.eq.'001').or.(mod.eq.'550').or.(mod.eq.'660').or.
     +    (mod.eq.'697')) then
          pool=26

c       invalid mods code for mail processing
        else
          pool = 1
        end if

        if (pool.eq.1) then

c       INTERNATIONAL
          if ((mod.eq.'032').or.(mod.eq.'033')) pool=43
          if ((mod.eq.'062').or.(mod.eq.'063')) pool=43
          if ((mod.eq.'090').or.(mod.eq.'092')) pool=43
          if ((mod.eq.'192').or.(mod.eq.'193')) pool=43
          if ((mod.eq.'102').or.(mod.eq.'103')) pool=43
          if ((mod.eq.'107').or.(mod.eq.'108')) pool=43
          if (mod.eq.'132') pool=43
          if ((mod.ge.'346').and.(mod.le.'347')) pool=43
          if (mod.eq.'349') pool=43
          if (((mod.ge.'343').and.(mod.le.'345')).or.
     +       (mod.eq.'348').or.((mod.ge.'350').and.(mod.le.'352'))) pool=43
          if (mod.eq.'454') pool=43
          if ((mod.eq.'545').or.(mod.eq.'546').or.((mod.ge.'573').and.
     +       (mod.le.'578')).or.(mod.eq.'580').or.(mod.eq.'681')) pool=43

c       ADMINISTRATION

c       2adm_out
          if (mod.eq.'597') pool=49
          if ((mod.eq.'920').or.(mod.eq.'922').or.(mod.eq.'924')) pool=49
          if ((mod.eq.'342').or.(mod.eq.'598').or.(mod.eq.'698').or.
     +       (mod.eq.'699').or.((mod.ge.'700').and.(mod.le.'702')).or.
     +       (mod.eq.'770').or.((mod.ge.'927').and.(mod.le.'928')).or.
     +       (mod.eq.'932')) pool=49
          if ((mod.eq.'354').or.(mod.eq.'613').or.(mod.eq.'614').or.
```

```
     +      (mod.eq.'622').or.(mod.eq.'627').or.(mod.eq.'632').or.
     +      (mod.eq.'705').or.((mod.ge.'707').and.(mod.le.'711')).or.
     +      ((mod.ge.'713').and.(mod.le.'740')).or.
     +      (mod.eq.'743').or.(mod.eq.'744').or.(mod.eq.'757').or.
     +      (mod.eq.'768')) pool=49
            if ((mod.eq.'758').or.(mod.eq.'759').or.(mod.eq.'760')) pool=49
            if ((mod.eq.'676').or.(mod.eq.'933').or.((mod.ge.'951').and.
     +      (mod.le.'955'))) pool=49
            if ((mod.eq.'706').or.(mod.eq.'929')) pool=49
            if ((mod.eq.'599').or.(mod.eq.'635').or.(mod.eq.'703').or.
     +      (mod.eq.'923').or.((mod.ge.'935').and.(mod.le.'939'))) pool=49
            if ((mod.eq.'641').or.(mod.eq.'704').or.((mod.eq.'940').and.
     +      (mod.eq.'945'))) pool=49
            if ((mod.eq.'601').or.(mod.eq.'655').or.((mod.eq.'946').and.
     +      (mod.eq.'950'))) pool=49
            if (mod.eq.'671') pool=49
            if (mod.eq.'664') pool=49
            if (((mod.ge.'455').and.(mod.le.'462')).or.((mod.ge.'471').and.
     +      (mod.le.'504'))) pool=49

             function 0 admin
            if (mod.eq.'582') pool=50
            if ((mod.eq.'581').or.(mod.eq.'573')) pool=50
            if (((mod.ge.'594').and.(mod.le.'596')).or.(mod.eq.'674')) pool=50
            if ((mod.eq.'645').or.(mod.eq.'672')) pool=50
            if ((mod.eq.'668').or.(mod.eq.'900').or.(mod.eq.'905')) pool=50
            if ((mod.eq.'646').or.(mod.eq.'675')) pool=50

             function 3 admin
            if ((mod.eq.'615').or.(mod.eq.'617').or.(mod.eq.'679').or.
     +      (mod.eq.'763').or.(mod.eq.'764').or.(mod.eq.'901').or.
     +      (mod.eq.'906')) pool=53
            if ((mod.eq.'761').or.(mod.eq.'762')) pool=53
            if (mod.eq.'647') pool=50
            if ((mod.eq.'765').or.(mod.eq.'766').or.(mod.eq.'772').or.
     +      (mod.eq.'773')) pool=53
            if ((mod.ge.'750').and.(mod.le.'752')) pool=53
            if ((mod.ge.'753').and.(mod.le.'754')) pool=50
            if ((mod.ge.'747').and.(mod.le.'749')) pool=53
            if ((mod.eq.'616').or.(mod.eq.'624').or.(mod.eq.'629').or.
     +      (mod.eq.'634').or.(mod.eq.'680').or.(mod.eq.'745').or.
     +      (mod.eq.'746')) pool=53

              function 4 admin
            if ((mod.ge.'980').and.(mod.le.'987')) pool=51

              function 5 admin
            if ((mod.eq.'540').or.(mod.eq.'556').or.(mod.eq.'569').or.
     +      (mod.eq.'579').or.(mod.eq.'591').or.(mod.eq.'592').or.
     +      (mod.eq.'610').or.(mod.eq.'623').or.(mod.eq.'628').or.
     +      (mod.eq.'633').or.((mod.ge.'636').and.(mod.le.'640')).or.
     +      ((mod.ge.'648').and.(mod.le.'651')).or.
     +      ((mod.ge.'682').and.(mod.le.'685')).or.
     +      ((mod.ge.'968').and.(mod.le.'969'))) pool=52

              function 6 admin
            if (mod.eq.'958') pool=46
            if (mod.eq.'959') pool=46
            if ((mod.eq.'541').or.(mod.eq.'557').or.(mod.eq.'566').or.
     +      (mod.eq.'572').or.(mod.eq.'611').or.
     +      ((mod.ge.'642').and.(mod.le.'644')).or.
     +      ((mod.ge.'652').and.(mod.le.'654')).or.
     +      ((mod.ge.'686').and.(mod.le.'692')).or.
     +      (mod.eq.'902').or.(mod.eq.'907')) pool=54

              function 7 admin
            if (mod.eq.'656') pool=55
            if ((mod.eq.'657').or.(mod.eq.'693').or.(mod.eq.'695')) pool=55
            if ((mod.eq.'658').or.(mod.eq.'694')) pool=55
            if ((mod.eq.'659').or.(mod.eq.'696')) pool=55
            if ((mod.ge.'551').and.(mod.le.'552')) pool=45
            if (mod.eq.'661') pool=55
            if (mod.eq.'662') pool=55
            if ((mod.eq.'663').or.(mod.eq.'903')) pool=55

              function 8 admin
            if (((mod.ge.'463').and.(mod.le.'470')).or.((mod.ge.'505').and.
     +      (mod.le.'538')).or.(mod.eq.'570').or.(mod.eq.'571').or.
     +      (mod.eq.'648').or.(mod.eq.'665').or.(mod.eq.'666').or.
     +      (mod.eq.'670').or.(mod.eq.'682').or.(mod.eq.'904')) pool=44
```

```
        function 9 admin
      if ((mod.ge.'780').and.(mod.le.'789')) pool=47

c       2adm_spc
      if (((mod.ge.'777').and.(mod.le.'778')).or.(mod.eq.'888').or.
     +    ((mod.ge.'988').and.(mod.le.'999'))) pool=57
      end if

      return
      end

c     ------------------------------------------------------------------
c     Uses Q19 to assign invalid tallies to appropriate MODS group

      function   q19(mm)

      integer*4    q19
      character*1  mm

      q19=0
      if (mm.eq.'A') q19=1   ! Manual
      if (mm.eq.'B') q19=2   ! OCR
      if (mm.eq.'C') q19=3   ! BCR/BCS
      if (mm.eq.'D') q19=4   ! BCR/S
      if (mm.eq.'E') q19=5   ! Carrier BCS
      if (mm.eq.'F') q19=6   ! LSM
      if (mm.eq.'G') q19=7   ! Letter Facer
      if (mm.eq.'H') q19=8   ! Flat Facer
      if (mm.eq.'I') q19=9   ! Sack Sorter
      if (mm.eq.'J') q19=10  ! Parcel Sorter
      if (mm.eq.'K') q19=11  ! FSM
      if (mm.eq.'L') q19=12  ! SPBS
      if (mm.eq.'M') q19=13  ! NMO
      if (mm.eq.'N') q19=14  ! Multi-Slide
      if (mm.eq.'O') q19=15  ! Conveyor
      if (mm.eq.'P') q19=16  ! ACDCS
      if (mm.eq.'Q') q19=17  ! Banding
      if (mm.eq.'R') q19=18  ! Culling
      if (mm.eq.'S') q19=19  ! RBCS
      if (mm.eq.'T') q19=20  ! Transportation
      if (mm.eq.'U') q19=21  ! Other

      return
      end

c     ------------------------------------------------------------------
c     Assigns manual labor type for Q19 answer A

      function type1(man)

      integer*4    type1
      character*1  man

      type1=0
      if (man.eq.'A') type1=1   ! Letter
      if (man.eq.'B') type1=2   ! Flat
      if (man.eq.'C') type1=3   ! Parcel
      if (man.eq.'D') type1=4   ! Coll/Cancel/Mtr
      if (man.eq.'E') type1=5   ! Prsrt Units
      if (man.eq.'F') type1=6   ! Open Units
      if (man.eq.'G') type1=7   ! Pouch/Rack
      if (man.eq.'H') type1=8   ! Platf Units
      if (man.eq.'I') type1=9   ! Other

      return
      end

c     ------------------------------------------------------------------

      function sort(f9602)

      integer*4    sort
      character*1  f9602

      sort=0

      if (f9602.eq.'A') sort=1   ! Sort Sacks
      if (f9602.eq.'B') sort=2   ! Sort Trays
      if (f9602.eq.'C') sort=3   ! Sort Pallets
```

```
      if (f9602.eq.'D') sort=4    !  Sort Rolling Containers

      return
      end
c
c      Assigns transportation type for Q19 answer T

      function type2(trp)

      integer*4    type2
      character*1  trp

      type2=0
      if (trp.eq.'A') type2=1
      if (trp.eq.'B') type2=2
      if (trp.eq.'C') type2=3
      if (trp.eq.'D') type2=4
      if (trp.eq.'E') type2=5
      return
      end
```

```
      Program windxmod

;     BY:   mike mcgrane
;     Date: 7/30/96
C
C     Purpose: Sum direct costs for Std A by weight increment in each cost pool
;


      implicit none

      integer*4     nact, n3c, npool, nw

      parameter     (n3c = 18)
      parameter     (nact = 278)
      parameter     (npool = 43)
      parameter     (nw = 16)


      real*8        dols3c(nw,npool,n3c)
      real*8        dols


      integer*4     ier, count, searchc, iact, windx
      integer*4     ipool, ihand, iw, iactv, i, j

      character*259 rec
      character*1   shape , rcshape
      character*4   act, cact
      character*4   acts(nact)
      character*4   acts3c(n3c)



      count=0
      ier=0

      open(15,file='activity.3c',iointent='input')
      read(15,'(a4)') acts3c

      do iact = 1, n3c
         do ipool = 1, npool
            do iw = 1, nw
               dols3c(iw,ipool,iact) = 0.
            end do
         end do
      end do


      open(20,file='../maps/activity.s',iointent='input')
21    format(a4)
      read(20,21) acts

!1    format(a259,f15.5,i2,i1,i3,i5)

      do while (ier.eq.0)
         read(5,11,iostat=ier,end=100) rec, dols, ipool, ihand, iw, iactv
!         act = rec(231:234)     ! f262
         write (act,'(i4.4)') iactv
         if (rec(231:234).ne.act) then
            print *,' F262 ',rec(231:234),' ne Actv field ',act
         end if

         iact = searchc(acts3c,n3c,act)

         if (iact.gt.0) then
            iw = windx(rec(175:179))
            if (iw.gt.0) then
               dols3c(iw,ipool,iact) = dols3c(iw,ipool,iact) + dols
            end if
         end if

      end do

100   print *, ' Read exit code = ',ier
      print *, ' Number of records processed ',count



      open(40,file='modwind.csv',iointent='output',recl=500)
41    format(a4,',',i2,',',20(f15.5,','))
```

```
      do iact = 1, n3c
         do ipool = 1, npool
            write (40,41) acts3c(iact), ipool, (dols3c(iw,ipool,iact),iw=1,nw)
         end do
      end do


      call exit
      end
C
C ---------------------------------------------------------------------


C       wind  - return index of weight graduation
C
C       mrm  8-14-92

      function windx(wchar)

      integer*4   windx
      integer*4   ilt, ioz, ilb,ier
      character*5 wchar
      character*1 flag, cd

199   format(1x,2i2)

      ier = 0

      cd = wchar(1:1)
      if ((cd.eq.'A').or.(cd.eq.'B')) then
         windx = 1
      else if ( (cd.eq.'C').or.(cd.eq.'D')) then
         windx = 2
      else if ((cd.eq.'E').or.(cd.eq.'F')) then
         windx = 3
      else if ((cd.eq.'G').or.(cd.eq.'H')) then
         windx = 4
      else if (cd.eq.'I') then
         read(wchar,199,iostat=ier) ilb, ioz
         if (ilb.eq.0) then
            if (ioz.le.16) then
               windx = ioz
            else
               windx = 0        ! bad entry in ounce field
            end if
         else
            if ((ilb.eq.1).and.(ioz.eq.0)) then
               windx = 16
            else
               windx = 0
            end if
         end if
      else
         windx = 0              ! bad weight code field
      end if

      if (ier.ne.0) windx = 0   ! read error - invalid weight index

      return
      end
C ---------------------------------------------------------------------
```

```
      Program windxbmc

C     BY:   mike mcgrane
C     Date: 7/30/96
C
C     Purpose: Sum direct costs for Std A by weight increment in each cost pool
C

      implicit none

      integer*4     nact, n3c, npool, nw

      parameter     (n3c = 18)
      parameter     (nact = 278)
      parameter     (npool = 11)
      parameter     (nw = 16)


      real*8        dols3c(nw,npool,n3c)
      real*8        dols


      integer*4     ier, count, searchc, iact, windx
      integer*4     ipool, ihand, iw, iactv, i, j

      character*259 rec
      character*1   shape , rcshape
      character*4   act, cact
      character*4   acts(nact)
      character*4   acts3c(n3c)



      count=0
      ier=0

      open(15,file='activity.3c',iointent='input')
      read(15,'(a4)') acts3c

      do iact = 1, n3c
         do ipool = 1, npool
            do iw = 1, nw
               dols3c(iw,ipool,iact) = 0.
            end do
         end do
      end do


      open(20,file='../maps/activity.s',iointent='input')
21    format(a4)
      read(20,21) acts

11    format(a259,f15.5,i2,i3)

      do while (ier.eq.0)
         read(5,11,iostat=ier,end=100) rec, dols, ipool, iw
         act = rec(231:234)     ! f262
         ipool = ipool - 30  ! map onto 1 to 11

         iact = searchc(acts3c,n3c,act)

         if (iact.gt.0) then
            iw = windx(rec(175:179))
            if (iw.gt.0) then
               dols3c(iw,ipool,iact) = dols3c(iw,ipool,iact) + dols
            end if
         end if

      end do

100   print *, ' Read exit code = ',ier
      print *, ' Number of records processed ',count



      open(40,file='bmcwind.csv',iointent='output',recl=500)
41    format(a4,',',i2,',',20(f15.5,','))

      do iact = 1, n3c
         do ipool = 1, npool
```

```
            write (40,41) acts3c(iact), ipool+30, (dols3c(iw,ipool,iact),iw=1,nw)
         end do
      end do


      call exit
      end
C
C ---------------------------------------------------------------------


C       wind  - return index of weight graduation
C
C       mrm  8-14-92

      function windx(wchar)

      integer*4   windx
      integer*4   ilt, ioz, ilb,ier
      character*5 wchar
      character*1 flag, cd

199   format(1x,2i2)

      ier = 0

      cd = wchar(1:1)
      if ((cd.eq.'A').or.(cd.eq.'B')) then
         windx = 1
      else if ( (cd.eq.'C').or.(cd.eq.'D')) then
         windx = 2
      else if ((cd.eq.'E').or.(cd.eq.'F')) then
         windx = 3
      else if ((cd.eq.'G').or.(cd.eq.'H')) then
         windx = 4
      else if (cd.eq.'I') then
         read(wchar,199,iostat=ier) ilb, ioz
         if (ilb.eq.0) then
            if (ioz.le.16) then
               windx = ioz
            else
               windx = 0        ! bad entry in ounce field
            end if
         else
            if ((ilb.eq.1).and.(ioz.eq.0)) then
               windx = 16
            else
               windx = 0
            end if
         end if
      else
         windx = 0              ! bad weight code field
      end if

      if (ier.ne.0) windx = 0   ! read error - invalid weight index

      return
      end
C ---------------------------------------------------------------------
```

```
      Program windxnmod

:     BY:   mike mcgrane
:     Date: 7/30/96
C
C     Purpose: Sum direct costs for Std A by weight increment in each cost pool
:

      implicit none

      integer*4     nact, n3c, npool, nw

      parameter     (n3c = 18)
      parameter     (nact = 278)
      parameter     (npool = 1)
      parameter     (nw = 16)


      real*8        dols3c(nw,npool,n3c)
      real*8        dols


      integer*4     ier, count, searchc, iact, windx
      integer*4     ipool, ihand, iw, iactv, i, j

      character*259 rec
      character*1   shape , rcshape
      character*4   act, cact
      character*4   acts(nact)
      character*4   acts3c(n3c)



      count=0
      ier=0

      open(15,file='activity.3c',iointent='input')
      read(15,'(a4)') acts3c

      do iact = 1, n3c
         do ipool = 1, npool
            do iw = 1, nw
               dols3c(iw,ipool,iact) = 0.
            end do
         end do
      end do


      open(20,file='../maps/activity.s',iointent='input')
21    format(a4)
      read(20,21) acts

11    format(a259,f15.5,i1,i3)

      do while (ier.eq.0)
         read(5,11,iostat=ier,end=100) rec, dols, ihand, iw
         act = rec(231:234)     ! f262
         ipool = 1

         iact = searchc(acts3c,n3c,act)

         if (iact.gt.0) then
            iw = windx(rec(175:179))
            if (iw.gt.0) then
               dols3c(iw,ipool,iact) = dols3c(iw,ipool,iact) + dols
            end if
         end if

      end do

100   print *, ' Read exit code = ',ier
      print *, ' Number of records processed ',count



      open(40,file='nmodwind.csv',iointent='output',recl=500)
41    format(a4,',',i2,',',20(f15.5,','))

      do iact = 1, n3c
         do ipool = 1, npool
```

```
            write (40,41) acts3c(iact), ipool, (dols3c(iw,ipool,iact),iw=1,nw)
         end do
      end do


      call exit
      end
C
C ---------------------------------------------------------------------


C       Wind  - return index of weight graduation
C
C       mrm  8-14-92

      function windx(wchar)

      integer*4   windx
      integer*4   ilt, ioz, ilb,ier
      character*5 wchar
      character*1 flag, cd

199   format(1x,2i2)

      ier = 0

      cd = wchar(1:1)
      if ((cd.eq.'A').or.(cd.eq.'B')) then
         windx = 1
      else if ( (cd.eq.'C').or.(cd.eq.'D')) then
         windx = 2
      else if ((cd.eq.'E').or.(cd.eq.'F')) then
         windx = 3
      else if ((cd.eq.'G').or.(cd.eq.'H')) then
         windx = 4
      else if (cd.eq.'I') then
         read(wchar,199,iostat=ier) ilb, ioz
         if (ilb.eq.0) then
            if (ioz.le.16) then
               windx = ioz
            else
               windx = 0        ! bad entry in ounce field
            end if
         else
            if ((ilb.eq.1).and.(ioz.eq.0)) then
               windx = 16
            else
               windx = 0
            end if
         end if
      else
         windx = 0              ! bad weight code field
      end if

      if (ier.ne.0) windx = 0   ! read error - invalid weight index

      return
      end
C ---------------------------------------------------------------------
```

```
      PROGRAM encode
C
C     file: enocode_wi.f
C
C       PURPOSE: Encode tallies with indexes of arrays instead of
C                actual data. Delete leave tallies.
C
C       AUTHOR: mrm
C       DATE  : 24-JUL-90
C       LAST MODIFIED: August 03, 1990 by mrm
C                      10-15-94  amr to split out to new cost groups
C       PROJECT: usps/iocs
C
C       SUBROUTINES:
C       LINKING:
C

      IMPLICIT NONE

      integer*4   numcf, numact, numor
      parameter   (numcf = 11)  ! number of cag - finnance number combs.
      parameter   (numor = 17)  ! number of operation/route codes
      parameter   (numact = 276) ! number of activity codes
      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   qtrind        ! function to assign quarter
      integer*4   wind          ! function to assign weight index
      integer*2   i1, i2, i3, i4, i5, i6, i8, i7, i9, iwalk
      integer*4   i, j, k, l, find,searchc,z
      integer*4   totall/0/
      integer*4   nonpb
      integer*4   ier/0/
      integer*4   countlv/0/
      integer*4   counto/0/
      integer*4   countg/0/
      integer*4   countsp/0/
      integer*4   countk/0/
      integer*4   countf/0/
      integer*4   lastq

      real*8   tvalue

      character*7    cagfins(numcf)
      character*4    acodes(numact), activity
      character*1    manmech,type,procdist,plat,prep,tallyact,misc
      character*1    wflag(numact) ! array of weight type flags
      character*7    junk
      character*211   rec

      logical DEBUG/.true./
C       logical DEBUG/.false./

C       read cagfin file

      open(14,file='../maps96/fincag.s',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       read activity.s file

      open(16,file='../maps96/activity.s',iostat=ier )
      if (ier.ne.0) then
         print *,' error opening activity.s = ',ier
         stop
      end if

17    format(a4,1x,a1)
      do i = 1,numact
         read(16,17) acodes(i), wflag(i)
      end do
      close (14)
```

```
      close (16)
      print *,'finished reading activity codes  '

C       open files

      open(30,file='encdata',iointent='output')
31    format(i2,i1,i1,i3,i2,i3,i2,f11.2)
      open(40,file='nocodes',iointent='output')
21    format(a,i3)
      open(44,file='itemcost',iointent='output')
45    format(a4,f11.2)

C       read through and encode file
      ier = 0
      z=0
      do while (ier.eq.0)
         read(5,21,iostat=ier,end=100) rec, iwalk
         z=z+1
         i2 = searchc(cagfins,numcf,rec(195:201))
         i3 = desigind(rec(186:187)) ! da code  - occupation
         i4 = orind(rec(188:189)) ! operation or route code
         if (i3.eq.1) i4 = 1    ! supervisors have no route or oper code
         i5 = bfind(rec(190:190))
         if (i3.eq.1) i5 = 1    ! supervisors have bf set to 0
         activity = rec(191:194)

         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

         i6 = searchc(acodes,numact,activity)


C         read (rec(56:57),'(i2)') i9 ! space category
C         if (i9.eq.0) i9 = 29
         i9 = 1

         i7 = wind(wflag(i6),rec(153:157))
         if (i7.eq.0) i7 = 17  ! counted items are placed in wi 17

         read(rec(202:211),'(f10.2)') tvalue

         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
         else
            if (rec(191:191).eq.'9') then  ! first digit of activity code f262
               countlv = countlv + 1
            else if (rec(187:187).eq.'4') then  ! second digit of da code f257
               countsp = countsp + 1
            else if (rec(201:201).eq.'K') then  ! cag k tally  f264
               countk = countk + 1
            else if (i2.eq.0) then
               print *,' invalid cagfin = ',rec
               countf = countf + 1
               write (40,21) rec
            else
               print *,' unknown exclusion, rec = ',rec
               print *,'    indexes = ',i2,i3,i4,i5,i6,i7
               counto = counto + 1
               write (40,21) rec
            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

      call exit
      end


C ------------------------------------------------------------------------
C       desigind     mrm 7-24-90
```

```
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  - mrm 7-24-90
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        -  returns index value for operation or route code
C
C  mrm   8-3-90

        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     wind  - return index of weight graduation
C
C     mrm  8-14-92

      function wind(flag,wchar)

      integer*4   wind
      integer*4   ilt, ioz, ilb,ier
      character*5 wchar
      character*1 flag, cd
```

```
*99     format(1x,2i2)

        ier = 0

        if (flag.eq.'0') then   ! no weight expected for this activity code
           wind = 1
        else if (flag.eq.'1') then ! Tallies separated by ounce increments
           cd = wchar(1:1)
           if ((cd.eq.'A').or.(cd.eq.'B')) then
              wind = 1
           else if ( (cd.eq.'C').or.(cd.eq.'D')) then
              wind = 2
           else if ((cd.eq.'E').or.(cd.eq.'F')) then
              wind = 3
           else if ((cd.eq.'G').or.(cd.eq.'H')) then
              wind = 4
           else if (cd.eq.'I') then
              read(wchar,199,iostat=ier) ilb, ioz
              if (ilb.eq.0) then
                 if (ioz.le.16) then
                    wind = ioz
                 else
                    wind = 0    ! bad entry in ounce field
                 end if
              else
                 if ((ilb.eq.1).and.(ioz.eq.0)) then
                    wind = 16
                 else
                    wind = 0
                 end if
              end if
           else
              wind = 0          ! bad weight code field
           end if
        else if (flag.eq.'2') then ! second class - new indexing
           read (wchar,199,iostat=ier) ilb, ioz
           ioz = ioz + ilb * 16
           if (ioz.le.16) then
              wind = (ioz+1) / 2
           else if (ioz.le.32) then
              if (ioz.le.24) then
                 wind = 9
              else
                 wind = 10
              end if
           else
              if (ilb.eq.3) then
                 wind = 11
              else if (ilb.eq.4) then
                 wind = 12
              else if (ilb.eq.5) then
                 wind = 13
              else if (ilb.le.10) then
                 wind = 14
              else if (ilb.le.25) then
                 wind = 15
              else if (ilb.le.70) then
                 wind = 16
              else
                 wind = 0
              end if
           end if
        else                    ! fourth or priority - max = 75 lbs
           read (wchar,199,iostat=ier) ilb, ioz
           if (ioz.gt.0) ilb = ilb + 1 ! round up to next whole pound
           if (ilb.le.1) then
              wind = 1
           else if (ilb.eq.2) then
              wind = 2
           else if (ilb.eq.3) then
              wind = 3
           else if (ilb.eq.4) then
              wind = 4
           else if (ilb.eq.5) then
              wind = 5
           else if (ilb.eq.6) then
              wind = 6
           else if (ilb.eq.7) then
              wind = 7
```

```
         else if (ilb.eq.8) then
            wind = 8
         else if (ilb.eq.9) then
            wind = 9
         else if (ilb.eq.10) then
            wind = 10
         else if (ilb.le.15) then
            wind = 11
         else if (ilb.le.20) then
            wind = 12
         else if (ilb.le.25) then
            wind = 13
         else if (ilb.le.30) then
            wind = 14
         else if (ilb.le.35) then
            wind = 15
         else if (ilb.le.75) then
            wind = 16
         else
            wind = 0          ! piece too heavy
         end if
      end if

      if (ier.ne.0) wind = 0  ! read error - invalid weight index

      return
      end
C -----------------------------------------------------------------------
```

```
% Name: Encdata.sm
; Sort encoded IOCS tally info
 nput file is 'encdata', recs are data sensitive upto 30 chars.
utput file is 'encdata.s', recs are data sensitive  upto 30 chars.
sort.
nd.
```

```
      PROGRAM liocatt

        PURPOSE: Allocate costs to raw tallies, and perform LIOCATT mixed
                 mail cost distribution
C
C       SUBROUTINES: loaddata, fillmixmap, costalloc, fungroup, level1, level2, level3
                     sortcost, sortlevb, report

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   nfrec, nlev1a, nlev1b, nfun
      integer*4   nlev2a, nlev2b, nlev3a, nlev3b
      integer*4   i, irun, iseed
      integer*4   argnum, length, num
      integer*2   data(6)
      integer*4   j
      real*8      value
      character*3 runtype
      logical     dofun

C     Command lines has either fun or op
      j = iargc()
      if (j.eq.0) then
         print *,' usage: liocatt <arg> '
         print *,'    where <arg> is "op" or "fun" '
      end if
      call  getarg(1,runtype)
      if (runtype.eq.'fun') then
         dofun = .true.
      else
         dofun = .false.
      end if

      call  fillmixmap          ! load mixed mail distribution map, and other files
      call  loaddata            ! load encdata.s
      if (dofun) then
         call  fungroup(nfun)   ! form function groups from operations
      else
         call  noop()           ! reorder indices for sortation
         nfun = nrec
      end if
      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 1993 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 *,' ************************************************** '

      call exit
      end

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

```
      subroutine loaddata
C
C       PURPOSE: Load encoded iocs data into encdata array

      IMPLICIT NONE

      include "liocatt.h"

      integer*4   i, j, k
      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',iointent='input')
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

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

```
      subroutine fillmixmap

C
C     PURPOSE: Read the mixed mail map "mmkey.codes" and
C              produce map for distributing mix mail codes
C
      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   i, j, k, ind
      integer*4   ier/0/
      integer*4   searchc, qtr, ofg, pay

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

      logical  flag

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

C     read activity.s file

      open(16,file='activity.s')
17    format(a4)
      do i = 1,numact
         read(16,17) acodes(i)
      end do

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

C     read mmcodes file - list of mixed mail activity codes

      open(18,file='mmcodes')
19    format(a4)
      do i = 1,nummix
         read(18,19) mmcodes(i)
      end do

C     read mmkey file - map from mixed mail codes to direct activity codes

      open(20,file='mxmail.dat')
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."    ") 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
C ---------------------------------------------------------------------------
```

```
      subroutine fungroup(numout)
C
C       PURPOSE: Add clerk/mail handler records up into functional groups
C                from operation codes.
C

      IMPLICIT NONE

      include  'liocatt.h'
      real*8       original_costs(numopr)
      real*8       fun_costs(numfun)
      real*8       sum, cost

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


      logical      flags(numact)
      logical      same, mixflag

      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)

        PURPOSE: Sort costalloc output for fungroup processing


      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
C       PURPOSE: Perform level one distribution of mixed mail costs to
C                direct mail codes. ( See flowchart LIOCATT Level I )
C

      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
      real*8      mixed

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

      logical     flags(numact)
      logical     mixflag(numact)
      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 exeeeeded , 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.maxl1b) 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 *,' maxl1b 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
C       PURPOSE: Perform level one distribution of mixed mail costs to
C                direct mail codes. ( See flowchart LIOCATT Level II )
C

      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) ! points to index in group of beginning
      ! of activity, basic function group
      ! and stores length of the group
      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      direct, mixed, mixeddist, ratio
      real*8      mixpig(npig)
      real*8      sdirect, smixed, smixeddist, sratio
      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, cmix, cw, cpig
      integer*2   ofg, pay, opr, bf, act, mixkey, w, ind, end, pig
      integer*2   bf4/4/
      integer*4   i, j, k
      integer*4   ier/0/
      integer*4   searchc

      logical     flags(numact)
      logical     mixflag(numact)
      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 exceeeded , 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 level3(numin,numouta,numoutb)
C
C       PURPOSE: Perform level three distribution of mixed mail costs to
C                direct mail codes. ( See flowchart LIOCATT Level III )
C

      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) ! points to index in group of beginning
      ! of activity, basic function group
      ! and stores length of the group
      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   cofg, cpay, copr, cbf, cact, cmix, cw, cpig
      integer*2   ofg, pay, opr, bf, act, mixkey, w, ind, end, pig
      integer*2   bf4/4/
      integer*4   i, j, k
      integer*4   ier/0/
      integer*4   searchc

      logical     flags(numact)
      logical     mixflag(numact)
      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 sortlev2a(n)

C
C       PURPOSE: Sort costalloc output for fungroup processing
C

      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 report(nlev1b,nlev2b,nlev3a,nlev3b)
C
C       PURPOSE: Produce report on results of Liocatt by activity for
C                clerks and mailhandlers

      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   unit, member, i, j, k
      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

!5    format(7a2,a8)
$5    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)

:     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 rptwi3c
C
C     PURPOSE: Produce report on results of Liocatt by activity for
C              clerks and mailhandlers
C
C     AUTHOR: MRM
C     DATE  : 17-AUG-90
C     LAST MODIFIED: 8-22-90
C
C     PROJECT:
C
C     SUBROUTINES:
C     LINKING:
C

      IMPLICIT NONE

      integer*4    numfun, numact, nwi, nmap, n3c

      parameter   (numact = 276) ! number of activity codes
      parameter   (nwi = 17)
      parameter   (n3c = 12)
      parameter   (nmap = 10)

      integer*4   size1, size2
      parameter   (size1 = nwi*n3c)

      real*8      mpdols(nwi,n3c)/size1*0./
      real*8      wsdols(nwi,n3c)/size1*0./
      real*8      crdols(nwi,n3c)/size1*0./

      real*8      indata

      integer*4   cofg, cpay, copr, cbf, cact, cmix, cw
      integer*4   ofg, pay, opr, bf, act, mix, w
      integer*4   unit, member, i, j, k, imap, iact
      integer*4   ier/0/
      integer*4   searchc
      integer*4   arg, length
      integer*4   page, line
      integer*4   map(nmap)

      character*4 acodes(numact), acode
      character*4 mapcodes(nmap)
      character*2 buffer

      character*40   paytitle, oprtitle

      logical flag/.false./
      logical DEBUG/.true./
C     logical DEBUG/.false./


C     read activity.s file

      open(16,file='../maps96/activity.s')
17    format(a4)
      do i = 1,numact
         read(16,17) acodes(i)
      end do

C     read map to 3rd class lines

      open(18,file='map.3c',iointent='input')
      do i = 1, nmap
         read(18,'(a4,i2)') mapcodes(i), map(i)
      end do


C     Open input files

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

C     Assemble data for report
```

```
      do unit = 20,21
         do while (ier.eq.0)
            read (unit,25,iostat=ier,end=100) pay,opr,bf,act,w, indata
            iact = searchc(mapcodes,nmap,acodes(act))
            if (iact.gt.0) then
               imap = map(iact)
               if (pay.eq.2) then
                  if (opr.eq.3) then
                     mpdols(w,imap) = mpdols(w,imap) + indata
                  else if (opr.eq.4) then
                     wsdols(w,imap) = wsdols(w,imap) + indata
                  end if
               else if (pay.eq.3) then
                  crdols(w,imap) = crdols(w,imap) + indata
               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
            iact = searchc(mapcodes,nmap,acodes(act))
            if (iact.gt.0) then
               imap = map(iact)
               if (pay.eq.2) then
                  if (opr.eq.3) then
                     mpdols(w,imap) = mpdols(w,imap) + indata
                  else if (opr.eq.4) then
                     wsdols(w,imap) = wsdols(w,imap) + indata
                  end if
               else if (pay.eq.3) then
                  crdols(w,imap) = crdols(w,imap) + indata
               end if
            end if
         end do
101      print *,' Read exit of unit ',unit,' = ',ier
         ier = 0
      end do

      open(40,file='rwi3c96.csv',iointent='output',recl=500)
41    format(i2,',',17(f14.2,','))

      write (40,*) 'Mail Processing'
      do i = 1, n3c
         write (40,41) i, (mpdols(w,i),w=1,nwi)
      end do
      write (40,*) 'Window Service'
      do i = 1, n3c
         write (40,41) i, (wsdols(w,i),w=1,nwi)
      end do
      write (40,*) 'City Carrier In-Office'
      do i = 1, n3c
         write (40,41) i, (crdols(w,i),w=1,nwi)
      end do



      call exit
      end

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

Appendix D

Input and Output Files

Table D-1: Format of FY96 IOCS Extract Used for Cadoc22c_Std

Field	First	Last	Length	Description	Source
1	1	2	2	Region and Division	PDC File
2	3	8	6	Finance Number	PDC File
6	9	17	9	Social Security Number	PDC File
7	18	18	1	CAG	PDC File
16	19	24	6	Begin Date	PDC File
9601	25	26	2	Pay Period	01 thru 27
27	27	27	1	Night Differential	Codes Q5A
28	28	28	1	Sunday Premium	Codes Q5A
112	29	29	1	In Office Activity	Codes Q16F
9206	30	30	1	Other Selections For 16F	Codes
9207	31	31	1	Work Center Type	Codes
114	32	34	3	FONS PSDS BMC Number	Codes Q18A
116	35	35	1	Platform Operations	Codes Q18B
117	36	36	1	Type Of Other Platform	
118	37	37	1	Collection And Prep Of Mail	Codes Q18C
119	38	38	1	Mail Proc And Distribution	Codes Q18D
120	39	39	1	Type Of INC SEC/CASE Dist	
9208	40	40	1	Type of Dist To Carriers	
9411	41	41	1	Type Sector/Segment Pass	
9412	42	42	1	Type DPS/Walk Sequence Pass	
121	43	43	1	Allied Labor	Codes 18D
122	44	44	1	Type Of Allied Labor	
123	45	45	1	Miscellaneous Operations	Codes Q18E
124	46	46	1	Window Service Part 1	Codes Q18F
125	47	47	1	Window Service Part 2	
126	48	48	1	Admin & Other Activities	Codes Q18G
9209	49	49	1	Employee On Break From	
9210	50	50	1	Other Selections For 18G	
9419	51	51	1	Type Distribution on Break From	
128	52	52	1	Manual Or Mech Operation	Codes Q19
9211	53	53	1	Type Of Manual Operation	
9212	54	54	1	Type of Transport Equipment	
9602	55	55	1	Sorting to	Codes
129	56	56	1	Directional Statement	Codes Q20
9213	57	57	1	Emp Hndl PC Item, Container	Codes 21A
9214	58	58	1	Single Item Type	Codes 21B
9215	59	59	1	Is The Item Empty	
9216	60	60	1	Identical Mail In Item	
9217	61	61	1	Top Piece Rule Apply	
9218	62	62	1	Contents Countable	
9219	63	63	1	Container Type	Codes 21C
9220	64	64	1	Is The Container Empty	
9221	65	65	1	Identical Mail In Container	
9901	66	69	4	Loose Card	Count
9902	70	73	4	Loose Letters	Count
9903	74	77	4	Loose Flats	Count
9904	78	81	4	Loose IPPS	Count
9905	82	85	4	Loose Parcels	Count
9906	86	88	3	Bundles	Count
9907	89	91	3	Con Cons	Count
9908	92	94	3	Flat Trays	Count

Table D-1 Continued

Field	First	Last	Length	Description	Source
9909	95	97	3	Letter Trays	Count
9910	98	100	3	SM Parcel Trays	Count
9911	101	103	3	Pallets	Count
9912	104	106	3	Other Items	Count
9913	107	109	3	Blue And Orange	SK Pouch CT
9914	110	112	3	Green	SK Pouch CT
9915	113	115	3	Orange Or Yellow	SK Pouch CT
9916	116	118	3	Brown	SK Pouch CT
9917	119	121	3	White	SK Pouch CT
9420	122	124	3	White # 2	SK Pouch CT
9421	125	127	3	White #3	SK Pouch CT
9918	128	130	3	Other	SK Pouch CT
9919	131	133	3	International	SK Pouch CT
133	134	134	1	Shape	Codes Q22A
134	135	135	1	Automation Compatible	Codes Q22B
9635	136	136	1	Shape Single Piece	Codes after June 31
9606	137	137	1	USPS Form	Codes After June 31
9222	138	138	1	RBCS Label	
135	139	139	1	Detached Address Card	Codes Q22A
911	140	140	1	Address Block	Codes Q22B
9608	141	141	1	Automation Rate Barcode	Yes or No after June 3
136	142	142	1	Indicia	Codes Q23A
137	143	143	1	Class Of Mail	Codes Q23B
9224	144	144	1	Carrier Route	
9611	145	145	1	Mail Markings - Class of Mail	Codes after June 31
9223	146	146	1	Dedicated Space	
138	147	147	1	Attachment Enclosure	
139	148	148	1	Secondary Markings-Zip+4	Codes Q23C
140	149	149	1	-Zip+4 Barcoded	
9501	150	150	1	-Walk Sequence	
9612	151	151	1	Auto First Class	
9613	152	152	1	Auto	
9614	153	153	1	Auto CR	
141	154	154	1	-Presorted	
9615	155	155	1	First Class Presorted	
9616	156	156	1	Single Piece	
142	157	157	1	-Carr Rte	
143	158	158	1	-Bulk Rate	
9617	159	159	1	ECRLOT	
9618	160	160	1	ECRWSH	
9619	161	161	1	ECRWSS	
144	162	162	1	-Nonprofit	
145	163	163	1	-Printed Matter	
9620	164	164	1	Special Standard Mail	
9621	165	165	1	Markings - Library Rate	
9463	166	166	1	-DMBC	
9464	167	167	1	-BSPS	
9225	168	168	1	-FIM	
146	169	169	1	-Business Reply	
9226	170	170	1	-Courtesy Mail	
9632	171	171	1	Additional Services Being Provided	
155	172	172	1	-Ancillary Mrkgs-Form 3811	Codes Q23D
156	173	173	1	-Form 3811A	

Table D-1 Continued

Field	First	Last	Length	Description	Source
157	174	174	1	-Form 3547/3579	
165	175	175	1	Weight Of Piece-LS 1/2 1 Oz	Codes Q23G
166	176	177	2	-Pounds	Codes Q23G
167	178	179	2	-Ounces	Codes Q23G
168	180	182	3	Revenue On Piece-Dollars	Codes Q23H
169	183	184	2	-Cents	Codes Q23H
170	185	185	1	-Mills	Codes Q23H
171	186	186	1	Postage Due Or Overpaid	Codes Q23I
172	187	189	3	Amount-Dollars	
173	190	191	2	-Cents	
174	192	192	1	-Mills	
176	193	200	8	ISSN Number	Codes Q23K
178	201	206	6	Publication Number	Codes Q23L
179	207	207	1	Application Pending	Codes Q23L
9227	208	209	2	Number Of Records Counted	Codes Q24
244	210	213	4	Activity Code	ALB040
246	214	216	3	-2	
247	217	219	3	-3	
248	220	222	3	-4	
249	223	225	3	-5	
257	226	227	2	Tally Roster Designation	ALB100
260	228	229	2	Tally Operation Route Code	ALB100
261	230	230	1	Tally Basic Function	ALB100
262	231	234	4	Tally Activity Code	ALB100
263	235	240	6	Tally Finance Number	ALB100
264	241	241	1	Tally Cag	ALB100
9246	242	245	4	Sample Weight (Heavy/Light Sample)	
9250	246	255	10	Tally Dollar Value	
9253	256	256	1	Is Tally Divided Item (A-X,Blank)	
9253	257	257	1	Is Tally Divided Item (A-X,Blank)	
9476	258	262	5	Total Item Count (Pre Division)	
9478	263	263	1	Type 24L BSPS (Y or N)	

File: fincag.s
sorted map of tally finance numbers and tally CAGs

333333A
444444A
555555A
666666A
777777B
777777C
777777D
777777E
777777F
777777G
777777H

File: mmcodes
sorted list of mixed mail activity codes

5300
5301
5302
5303
5331
5340
5341
5345
5460
5461
5610
5620
5650
5700
5750