Survey of Income and Program Participation

Methods Of Processing Unit
Data Longitudinally On The SIPP

8910 87

Karen E. Smith
Congressional Budget Office

May 1989

The analysis of this paper is that of the author and should not be attributed to the Congressional Budget Office. The author thanks Jodi Korb for being the sounding board for the ideas presented in this paper, and Steve Long, Roald Euller, Roberton Williams, Ivon Katz, and Ron Moore for reviewing this paper and providing helpful comments.

The view expressed is that of the author and do not necessarily reflect that of the Census Bureau.

TABLE OF CONTENTS

INTRODUCTION

Analysts who want to examine events that occur over time face the problem of how best to process information from longitudinal data files. The standard approach for processing such data is to create a "person-linked" file, by linking all periods of information for each person onto one record.1/ Person-level longitudinal analysis involves looking at each person-record to see how characteristics or behavior changed over time. While this approach works if only the individual's characteristics are of interest, it is cumbersome if the analyst wants to look at groups of people such as families or households, because such groups may change over time -- people move out of households, people marry into families, people die.

An alternate approach for processing longitudinal data is to create a "person-month" file, by keeping each period of information for each person on its own record. Person-level longitudinal analysis involves looking at multiple records for each person to see how characteristics or behavior changed over time; looking at multiple records across months is equivalent to looking at a single person-linked record in the other approach. This paper will demonstrate that the person-month file is less cumbersome than the

---

1/ The standard method of processing a longitudinal file is described by, for example, Marita Servais, "Creating SIPP Longitudinal Files Using OSIRIS.IV," U.S. Bureau of the Census, Survey of Income and Program Participation: 1987--Selected Papers Given at the Annual Meeting of the American Statistical Association, San Francisco, California, August 16-20, 1987, pp. 129-134.

person-linked file for analysis of groups of people. This approach also provides substantial resource savings, even if only person-level analysis is done.

This paper illustrates the use of both person-linked files and person-month files for doing longitudinal analysis with monthly data from the Survey of Income and Program Participation (SIPP). First, some background pertaining to the SIPP and general problems of defining units is provided. The paper then describes and examines each method's ability to do person-level analysis and unit-level analysis. In this instance, all examples of unit-level analysis use the family as the unit, although the results are equally applicable to other types of units. It then compares the two methods of processing and their resource requirements. Finally, it includes an appendix with computer code in both PL1 and SAS that implement the person-month method.

UNITS ON THE SIPP

The SIPP collects data for a sample of the population over a thirty-six month period. The survey follows all people contacted for the initial interview over the entire period, and also gathers data for all people living with those people at each interview.[2] A typical unit at the beginning of the survey may contain a head of household, a spouse, and two children. If one of the children gets married and moves out of the household, the child and the child's spouse are surveyed. If an uncle moves in with the parents, he also is surveyed. But if the uncle moves out at any point, he no longer is surveyed because he was not in the original sample unit. The sample unit, therefore, is not static. It may change size, split apart, or merge back together. The dynamic nature of units forces the analyst to make decisions about the definition of a unit over time.

There are several ways one could define a "family" unit.[3] For example, if a husband and wife get divorced between months one and two, one could consider the divorce the end of one family (the husband and wife in month one) and the start of two families (the husband in month two and the wife in month two), or one could consider the husband as the head of the family regardless of any transitions in the family. In this case, the husband would be in the same family in months one and two, and the wife would be in a

---

2/ The survey population is divided into four rotations. Rotations one and two have thirty-six months of data. Rotations three and four have thirty-two months of data. The absence of four months of data for two rotations creates no problems for this analysis.

3/ For more discussion of longitudinal units, see D.B. McMillen and R.A. Herriot, TOWARD A LONGITUDINAL DEFINITION OF HOUSEHOLDS, "Survey of Income and Program Participation and Related Longitudinal Surveys: 1984," compiled by Daniel Kasprzyk and Delma Frankel. U.S. Bureau of the Census, Washington, D.C. 1984.

new family in month two. Within these dynamic families, the person is the only unit that remains constant over time. Rather than attempt a longitudinal definition of families, one solution is to ascribe the family-level characteristics to each individual in the family. To continue the example above, family size for the head of household, spouse, and two children is four. Family size of four would then be associated to each person in the family. When the child gets married, a family size of three would be associated to the head of household, spouse, and remaining child. In this form, all unit changes over time are measured at the person level. Looking at the family size information for the head of household over time would reveal a change from a family size of four to a family size of three. When the uncle moved into the family, the head of household's family size would be four. This paper uses this approach.

PERSON-LINKED FILE

The standard method of longitudinally processing the SIPP data links each person's thirty-six months of data onto a single record, creating a person-linked file (see Figure 1). This file can be created by merging the nine waves of data by the person index and writing out one record per person.[4/] For example, each record contains information for all thirty-six months on that person's income from earnings, child support, alimony, hours worked, etc.

FIGURE 1: PERSON-LINKED FILE

Person	Month 1	Month 2. .	.Month 36
1	XXX	XXX . . .	XXX
2	XXX	XXX . . .	XXX
.	.	. . . . .	.
.	.	. . . . .	.
.	.	. . . . .	.
N	XXX	XXX . . .	XXX

NOTE: XXX symbolizes all data items for person i and month j.

Person-level Analysis

Thirty-six months of information can be observed at the person level by looking across the person-linked file records. A transition is a change in any analysis variable from month n to month n+1. In Figure 2, note that Person 1 earned 950 dollars in months one through thirty-six. Person 2 earned 100 dollars in months one

4/ In the SIPP, a person is linked by sample unit identification number (SU-ID), entry household address identification number (PP-ENTRY), person number (PP-PNUM), and month. Month is a function of the sample unit rotation (SU-ROT), and wave (PP-WAVE).

through four, and 120 dollars in months five through thirty-six. Person N earned 800 dollars in months one through three and 0 dollars in months four through thirty-six. By changing the relevant variable, similar analysis could examine child support, alimony, hours worked, etc.

FIGURE 2: LONGITUDINAL ANALYSIS OF PERSON EARNINGS AND CHILD SUPPORT ON THE PERSON-LINKED FILE

Person		Month									
Number	Variables	1	2	3	4	5	6	7	8	9...	36
1	Earnings	950	950	950	950	950	950	950	950	950...	950
	Child Support	0	0	0	0	0	0	0	0	0...	0
2	Earnings	100	100	100	100	120	120	120	120	120...	120
	Child Support	0	90	90	90	90	90	90	100	100...	100
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
N	Earnings	800	800	800	0	0	0	0	0	0...	0
	Child Support	0	0	0	0	0	0	0	0	0...	0

Family-level Analysis

Suppose that one wanted to look not at the characteristics of individuals, but at those of families--that is, groups of related people living together. If all people originally in a family are present at the same address at each interview for the entire period, a person-linked file presents no problems.

When the person-linked file is sorted by the family index in month one, the people in each family in month one are contiguous records on the file.5/ Consider a family, for example, in which Person 1 is the head of household, Person 2 is the spouse, and Persons 3 and 4 are their children, and they live at the same address for all thirty-six months (see Figure 3). This family has no composition changes throughout the thirty-six months and in that sense is "ideal" for processing. In this case, monthly family-level characteristics can be calculated by looking at the sequential records of all the people in the sample unit for each month. In this example, the family's income from earnings in month one is the sum of individual earnings in month 1 for everyone in the family, or 1050 dollars. Looping through the months for everyone in the sample unit yields a monthly summary of family income from earnings. The family's income from earnings is 1050 dollars in months one through four and 1070 dollars in months five through thirty-six.

4. In the SIPP, families have the sample unit identification number (SU-ID), household address identification number (H*-ADDID), and family number (F*-NUMBR) in common.

FIGURE 3: LONGITUDINAL ANALYSIS OF FAMILY EARNINGS FOR THE IDEAL FAMILY ON THE PERSON-LINKED FILE

Sample Unit	Person Number	Relation	Variables	Month 1	2	3	4	5... 36
1	1	Head	Address ID	1	1	1	1	1... 1
			Earnings	950	950	950	950	950...950
1	2	Spouse	Address ID	1	1	1	1	1... 1
			Earnings	100	100	100	100	120...120
1	3	Child One	Address ID	1	1	1	1	1... 1
			Earnings	0	0	0	0	0... 0
1	4	Child Two	Address ID	1	1	1	1	1... 1
			Earnings	0	0	0	0	0... 0
Family Earnings				1050	1050	1050	1050	1070..1070

The calculation of monthly family-level characteristics is less straightforward when the people in a family change. For example, assume that a family in month one includes a father, mother, and two children. In month two, the parents divorce and the mother and first child move into a new home. The father and second child remain at the same residence. In month three, the first child moves in with the father and the second child moves in with the mother (see Figure 4).

FIGURE 4: CHANGING FAMILY COMPOSITION ON THE PERSON-LINKED FILE

Relation	Variable	Month 1	2	3. . .36
Father	Address ID	1	1	1. . . 1
Mother	Address ID	1	2	2. . . 2
Child One	Address ID	1	2	1. . . 1
Child Two	Address ID	1	1	2. . . 2

In month one, the father, mother, and two children are one family. In month two, the father and the second child are a family living at the original address, and the mother and the first child are a family at a new address. In month three, the father and the first child are a family living at the original address, and the mother and the second child are a family at the new address. As the months change, the people in each family no longer are contiguous records. In month two, the father and second child are a family, but are separated by two records. In month three, the father and the first child are separated by the mother's record, and the mother and the second child are separated by the first child's record. There is no way to arrange the records in Figure 4 and have the people in the same families contiguous for all months.

For the ideal family, the algorithm for creating family analysis variables worked by processing related people who are contiguous in the structure. As family composition changes over time, however, family members will not always be contiguous. Several solutions exist for this problem, none of which is completely satisfactory.

Three possible solutions described here are:

- o Move individual person-months;
- o Move entire person-linked record; and
- o Use record pointers to link family members.

It is possible to move the individual person-months on the person-linked file to get the data for all family members contiguous on the file (see Figure 5). Moving individual months, however, destroys the horizontal representation of time that the person-linked file provides. With this method, before doing longitudinal person-level analysis, the person-months would have to be moved back to their original order.

FIGURE 5. MOVE THE INDIVIDUAL PERSON-MONTHS ON THE PERSON-LINKED FILE

Record Number	Variables	Month 1	2	3. . .36
1	Relation	F	F	F. . . F
	Address ID	1	1	1. . . 1
2	Relation	M	C2	C1. . .C1
	Address ID	1	1	1. . . 1
3	Relation	C1	M	M. . . M
	Address ID	1	2	2. . . 2
4	Relation	C2	C1	C2. . .C2
	Address ID	1	2	2. . . 2

Note: F=Father, M=Mother, C1=Child One, C2=Child Two

It is possible to rearrange the person-linked file records to get the data for all family members contiguous on the file for month n (see Figure 6). This solution changes the family order for months not equal to n, and thus requires sorts for every sample unit and month.

FIGURE 6. REARRANGE THE PERSON-LINKED FILE RECORDS

For Month One Analysis:

Relation	Variables	Month 1	2	3. . .36
Father	Address ID	1	1	1. . . 1
Mother	Address ID	1	2	2. . . 2
Child One	Address ID	1	2	1. . . 1
Child Two	Address ID	1	1	2. . . 2

For Month Two Analysis:

Relation	Variables	Month 1	2	3. . .36
Father	Address ID	1	1	1. . . 1
Child Two	Address ID	1	1	2. . . 2
Mother	Address ID	1	2	2. . . 2
Child One	Address ID	1	2	1. . . 1

For Month Three Analysis:

Relation	Variables	Month 1	2	3. . .36
Father	Address ID	1	1	1. . . 1
Child One	Address ID	1	2	1. . . 1
Mother	Address ID	1	2	2. . . 2
Child Two	Address ID	1	1	2. . . 2

Rather than physically moving individual person-months or person-records to make family members contiguous in the file, pointers can be used to keep track of the family groupings across the months. Within each sample unit and month, the pointers link the people in each household and family (see Figure 7). Although these methods work, they are cumbersome to process as well as conceptualize.

FIGURE 7. POINTERS TO FAMILIES ON THE PERSON LINKED FILE

Relation		Month 1	2	3	. . .	36
	Monthly Pointers	HH1->FAM1	HH1->FAM1 HH2->FAM1	HH1->FAM1 HH2->FAM1		HH1->FAM1 HH2->FAM1
Father	Address ID	1	1	1	. . .	1
Mother	Address ID	1	2	2	. . .	2
Child One	Address ID	1	2	1	. . .	1
Child Two	Address ID	1	1	2	. . .	2

NOTE: HH - Household Pointer.
FAM - Family Pointer.

PERSON-MONTH FILE

Longitudinal analysis can be simplified if the data for each person are separated into thirty-six records, one for each month, rather than combined in a single record. This file can be created by concatenating the nine waves of data and writing out four records per wave per person. For example, each record contains information for one month on that person's income from earnings, child support, alimony, hours worked, etc. These person-month records then can be sorted in order to group together any type of unit, including people, families, and households.

Person-level Analysis

To analyze people, sort the person-month file by the unique person index.6/ Because each person-month is its own record, the person-month file is a rectangular file and can be sorted using any standard sort program, such as Syncsort. After the sort, the thirty-six months of data for each person are contiguous records on the file (see Figure 8).

6/ The unique person index is the concatenation of sample unit identification number (SU-ID), entry address identification number (PP-ENTRY), person number (PP-PNUM), and month. Month is a function of the sample unit rotation (SU-ROT), and wave (PP-WAVE).

FIGURE 8. **PERSON-MONTH FILE SORTED BY PERSON INDEX**

Person	Month	Data
1	1	XXX
1	2	XXX
1	3	XXX
.	.	.
.	.	.
1	36	XXX
- - -	- - -	- - -
2	1	XXX
2	2	XXX
.	.	.
.	.	.
2	36	XXX
- - -	- - -	- - -
.	.	.
N	36	XXX

Note: Dotted lines differentiate people.

Thirty-six months of information can be observed at the person level by looking through the thirty-six sequential person-month file records. A transition is a change in any analysis variable from month n to month n+1. In Figure 9, note that Persrn 1 earned 950 dollars in months one through thirty-six. Person 2 earned 100 dollars in months one through four, and 120 dollars in months five through thirty-six. Person N earned 800 dollars in months one through three and 0 dollars in months four through thirty-six. By changing the relevant variable, similar analysis could examine child support, alimony, hours worked, etc.

FIGURE 9. LONGITUDINAL ANALYSIS OF PERSON EARNINGS AND CHILD SUPPORT ON THE PERSON-MONTH FILE

Person	Month	Earnings	Child Support
1	1	950	0
1	2	950	0
1	3	950	0
.	.	.	.
.	.	.	.
1	36	950	0
- - -	- - -	- - -	- - -
2	1	100	0
2	2	100	90
2	3	100	90
2	4	100	90
2	5	120	90
.	.	.	.
.	.	.	.
2	36	120	100
- - -	- - -	- - -	- - -
.	.	.	.
.	.	.	.
- - -	- - -	- - -	- - -
N	1	800	0
N	2	800	0
N	3	800	0
N	4	0	0
.	.	.	.
.	.	.	.
N	36	0	0

Note: Dotted lines differentiate people.

Family-level Analysis

To analyze families, sort the person-month file by the unique family index.[7] Again, this sort is done using a standard sort program. After the sort, the person-month records for all people in each family per month are contiguous records on the file (see Figure 10).

Consider the family in Figure 10, for example, in which Person 1 is the head of household, Person 2 is the spouse, and Persons 3 and 4 are their children, and they live at the same address for all thirty-six months. This family has no composition changes throughout the thirty-six months. In this case, monthly family-level characteristics can be calculated by looking at the sequential records of all the people with the same family index. In this example, the family's income from earnings in month one is the sum of individual earnings in month 1 for everyone in the family, or 1050 dollars.[8] Looping through all of the person-month records and grouping families yields family income from earnings for all months. The family's income from earnings is 1050 dollars in months one through four and 1070 dollars in months five through

---

7/ The unique family index is the concatenation of the sample unit identification number (SU-ID), household address identification number (H*-ADDID), family identification number (F*-NUMBR), and month. Month is a function of the sample unit rotation (SU-ROT) and wave (PP-WAVE).

8/ Family earnings is a variable provided on the file by the Bureau of the Census. It would not be necessary to create it as described in this example. It is, however, illustrative of family-type variables.

thirty-six.

The calculation of monthly family-level characteristics is exactly the same when the people in a family change. For example, assume that a family in month one includes a father, mother, and two children. In month two, the parents divorce and the mother and first child move into a new home. The father and second child remain at the same residence. In month three, the first child moves in with the father and the second child moves in with the mother (see Figure 11).

FIGURE 10: LONGITUDINAL ANALYSIS OF FAMILY EARNINGS FOR THE IDEAL FAMILY ON THE PERSON-MONTH FILE SORTED BY FAMILY INDEX

Family Index							
Sample Unit Number	Month	Household Address Id	Family Number	Person Number	Relation	Person Earnings	Family Earnings
1	1	1	1	1	F	950	1050
1	1	1	1	2	M	100	1050
1	1	1	1	3	C1	0	1050
1	1	1	1	4	C2	0	1050
1	2	1	1	1	F	950	1050
1	2	1	1	2	M	100	1050
1	2	1	1	3	C1	0	1050
1	2	1	1	4	C2	0	1050
1	3	1	1	1	F	950	1050
1	3	1	1	2	M	100	1050
1	3	1	1	3	C1	0	1050
1	3	1	1	4	C2	0	1050
1	4	1	1	1	F	950	1050
1	4	1	1	2	M	100	1050
1	4	1	1	3	C1	0	1050
1	4	1	1	4	C2	0	1050
1	5	1	1	1	F	950	1070
1	5	1	1	2	M	120	1070
1	5	1	1	3	C1	0	1070
1	5	1	1	4	C2	0	1070
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	36	1	1	1	F	950	1070
1	36	1	1	2	M	120	1070
1	36	1	1	3	C1	0	1070
1	36	1	1	4	C2	0	1070

Note: Dotted lines differentiate families.
F=Father, M=Mother, C1=Child One, C2=Child Two

FIGURE 11: CHANGING FAMILY COMPOSITION OF THE PERSON-MONTH FILE

Family Index							
Sample Unit Number	Month	Household Address Id	Family Number	Person Number	Relation	Person Earnings	Family* Earnings
1	1	1	1	1	F	950	1050
1	1	1	1	2	M	100	1050
1	1	1	1	3	C1	0	1050
1	1	1	1	4	C2	0	1050
1	2	1	1	1	F	950	950
1	2	1	1	4	C2	0	950
1	2	2	1	2	M	100	100
1	2	2	1	3	C1	0	100
1	3	1	1	1	F	950	950
1	3	1	1	3	C1	0	950
1	3	2	1	2	M	100	100
1	3	2	1	4	C2	0	100
1	4	1	1	1	F	950	950
1	4	1	1	3	C1	0	950
1	4	2	1	2	M	100	500
1	4	2	1	4	C2	0	500
1	5	1	1	1	F	950	950
1	5	1	1	3	C1	0	950
1	5	2	1	2	M	120	120
1	5	2	1	4	C2	0	120
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	36	1	1	1	F	950	950
1	36	1	1	3	C1	0	950
1	36	2	1	2	M	120	120
1	36	2	1	4	C2	0	120

Note: F=Father, M=Mother, C1=Child One, C2=Child Two
*Newly created variable

In month one, the father, mother, and two children are one family. In month two, the father and the second child are a family living at the original address, and the mother and the first child are a family at a new address. In month three, the father and the first child are a family living at the original address, and the mother and the second child are a family at the new address. As the months change, the people in each family are contiguous records. In month two, the father and second child are a family, and are contiguous records. In month three, the father and the first child are a family, and are contiguous records, and the mother and the second child are a family, and are contiguous records. The sort has arranged the records in Figure 11 to have the people in the same family contiguous for all months.

For the ideal family, the algorithm for creating family analysis variables worked by processing related people who are contiguous in the structure. As family composition changes over time, family members will be contiguous.

Calculating any family-level analysis variable involves looping through the sequential person-month records of the people with the same family index and then summarizing the family data. For example, to find family earnings, one would add up the person-level earnings for all of the people in the family. For example, the first family in Figure 11 (sample unit 1, month 1, household address id 1, family number 1) received 1050 dollars of earnings, and the second family (sample unit 1, month 2, household address id 1, family number 1) received 950 dollars of earnings.

While family-level analysis is straightforward with this sort order, the family variables are not associated with any longitudinal notion of time. Since the only unit that remains constant over time is the person, if family variables are associated with people, they may then be associated with time. To associate these variables with people, the new family variables should be tacked onto each person-month record for each person in each family as seen in Figure 11. Associating these variables with time requires a second sort.

The person-month file sorted by the unique person-index has all thirty-six months for each person together on the file. The person-index order associates person data with time. To analyze the new family variables over time, one must sort the family-index sorted file with the newly created family variables attached by the unique person index. After sorting, the new family variables are associated with the ordered person-months and are available for longitudinal analysis (see Figure 12). By looping through the thirty-six sequential person-month records, changes in family characteristics over time can be observed. In Figure 12, note that Person 1's family earned 1050 dollars in month one, and earned 950 dollars in months two through thirty-six. Person 2's family earned 1050 dollars in month one, and earned 100 dollars in months two and three. By month thirty-six, Person 2's family earned 120 dollars.

Person 3's family earned 1050 dollars in month one, 100 dollars in month two, and 950 dollars in months three through thirty-six. Person 4's family earned 1050 dollars in month one, 950 dollars in month two, 100 dollars in month three, and finally 120 dollars in month thirty-six. Transitions in family-level characteristics are analyzed exactly the same way as transitions in person-level characteristics.

FIGURE 12. PERSON-MONTH FILE SORTED BY PERSON INDEX WITH FAMILY ANALYSIS VARIABLES ADDED

Person Index							
Sample Unit Number	Entry Address Id	Person Number	Month	Household Address Id	Relation	Person Earnings	Family Earnings
1	1	1	1	1	F	950	1050
1	1	1	2	1	F	950	950
1	1	1	3	1	F	950	950
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	1	1	36	1	F	950	950
1	1	2	1	1	M	100	1050
1	1	2	2	2	M	100	100
1	1	2	3	2	M	100	100
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	1	2	36	2	M	120	120
1	1	3	1	1	C1	0	1050
1	1	3	2	2	C1	0	100
1	1	3	3	1	C1	0	950
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	1	3	36	1	C1	0	950
1	1	4	1	1	C2	0	1050
1	1	4	2	1	C2	0	950
1	1	4	3	2	C2	0	100
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	1	4	36	2	C2	0	120
2	1	1	1	1	F	0	0
.	.	.	.	.	.	.	.

Note: Dotted lines differentiate people.

DISCUSSION

File creation using the person-linked file requires nine merges by the person index. File creation using the person-month file requires data concatenation and a unit sort.

Person-level analysis on the person-month file is equivalent to a merged person-linked file. Calculating any person-level analysis variable involves looping through the sequential person-month records with the same person index and then summarizing the person data. This is exactly the same process as with the person-linked file except that rather than processing across a single record for thirty-six months, processing is done through individual records for thirty-six months.

Family-level analysis with the person-month file is less complicated than family-level analysis with the person-linked file. Using the person-month structure, any change in family composition over time is irrelevant. There is no need to move person-months, move person-records, or use pointers to form contiguous record-structures of families as with the person-linked file.

The person-month structure is elegant by virtue of its simplicity. Analyzing any unit is a matter of sorting by an index that uniquely defines that unit--to analyze households, sort by household index; to analyze families, sort by family index; to analyze subfamilies, sort by the subfamily index. No matter how the unit is defined, associating that unit with time is a matter of sorting by the person index and processing a simple structure. While this may seem like a lot of sorting, it is only two. With the person-linked file, each of the nine waves must be sorted before they may be linked. That requires nine sorts. Moving person-months, moving person-records, or using pointers to form contiguous record-structures of families all require sorts per sample unit, household address id, family number, and month. Two sorts, by contrast, is small.

RESOURCE REQUIREMENTS

The sheer size of the SIPP presents many real resource constraints. Some of the most significant are tape mounts, logical record length, space, cost, computer time, and programmer time. Although the person-month file structure corrects some of these problems, resources limits will always be a factor when processing the SIPP.

Many computer installations have a limited number of tape drives. The rectangular file supplied by the Bureau of the Census is on nine different data files--one for each four month wave. Creating a person-linked file requires mounting all nine data files simultaneously, drawing nine extracts, sorting nine extracts, and finally merging nine extracts. Many computer systems lack sufficient tape drives to handle nine simultaneous tape mounts. With this constraint, file creation requires several steps.

The person-month file faces no tape mount constraint. Using this file structure, the nine wave data files can be read one file after another, simply by concatenating the tape reels. This requires only one tape drive for the input data and one tape drive for the output data. Once all of the waves are concatenated, the file is ready to be sorted by the unit index for doing analysis.

The person-linked file may be constrained by its record length. Each wave of the SIPP data is 5,352 bytes long. Merging the nine SIPP waves keeping all 5,352 bytes yields a data set with a logical record length of 48,168. Most computers cannot process such a large record length.

The person-month file structure faces no logical record length constraint. Because each record contains only the data for a single month, the logical record length of a person-month record is essentially one thirty-sixth of the logical record length of the person-linked file record.9/

9/ The logical record length is not exactly one thirty-sixth, because non-monthly variables, such as the sample unit identification number, must be duplicated for each month.

The person-linked file requires a vast amount of wasted space. Throughout the course of the survey, people enter and exit the sample. If a person marries, the new spouse enters the sample at the month of the marriage. There is no data for the new spouse for any month prior to the marriage. Over time, people may drop out of the sample. For these people, there is no data after they leave the sample. With the person-linked file, missing data must be filled in for any month a person is absent from the sample (see Figure 13). This wasted space amounts to the number of absent months times the monthly record length.

FIGURE 13: PERSON-LINKED FILE WITH ABSENT MONTHS

	Month				
	1	2	3	4. . .35	36
Person 1	1	1	0	0. . . 0	0
Person 2	1	1	1	0. . . 0	1
.	.	.	.	. .	.
.	.	.	.	. .	.
Person N	0	0	0	0. . . 0	1

NOTE: 0s represents months of wasted space.

The person-month file only contains records for the months each person is actually in the sample (see Figure 14). This property of the person-month structure potentially saves space. On the other hand, because the SIPP data has many variables that are not truly monthly, creating the person-month file requires that the programmer duplicate the non-monthly variables. Depending on the extent of non-monthly variable duplication, the person-month file may or may not save space.

FIGURE 14: PERSON-MONTH FILE WITH ABSENT MONTHS

Person	Month
1	1
1	2
2	1
2	2
2	3
2	36
.	.
.	.
N	36

The person-month file's greatest advantage over the person-linked file is the resource savings in programmer time. There is an extraordinary amount of computer code--that is, of programmer time--required to manipulate units over time using the person-linked file. Each record requires pointers for the household, family, and month. These pointers must be created, a difficult process in itself. Finally, processing these records requires looping through sample units, households, families, and months.

The method for processing units over time using the person-month file, in comparison, is simple. There are no pointers, and processing requires looping only through the unit index. The code needed to manipulate units consists of a packaged sort routine. For the programmer, sorting is equivalent to a function call. Thus, hundreds of lines of computer code needed for the person-linked file may be replaced with a single line.

CONCLUSION

The person-linked file is a logical approach for processing chronologically arranged data if only person-level characteristics are of interest. It is the standard method because it is logical, and many analysts have chosen to process longitudinal data in this way. Unfortunately, it is difficult to use for handling unit-level analysis and its real resource constraints force the analyst to find a new approach.

The person-month file presented here offers a new approach. It provides the same logical person structure that gives the person-linked file structure its appeal, but eliminates the unit manipulation constraint, the tape mount constraint, and the logical record length constraint. It reduces the amount of programmer time needed for analysis. Moreover, because of its simplicity, the chance of error--especially errors that will not be found--is much reduced. In short, the person-month approach saves both money and aggravation.

APPENDIX

The computer code needed to implement the person-month file requires three steps. They are:

1. Concatenate the nine SIPP waves, output a person-month file, and sort the person-month file by the unit index.
2. Process units to create the unit-level analysis variables, and sort by the person index.
3. Process people to do longitudinal analysis.

This appendix includes the computer code for all three steps in both PL1 and SAS. It also includes the outline of a program that converts the SIPP wave 1 to look like the SIPP wave 3. This allows the programmer to treat wave 1 just like all of the other waves.

PL1 is a powerful and flexible language. It has the ability to do structure assignment by name and retain records in an array. This feature makes processing the person-month file relatively simple. SAS, however, is a procedure oriented language with comparatively inflexible data manipulation ability. It also has a great deal of overhead both in CPU time and memory space. If the programmer has a choice in language, a PL1 type language will ultimately be easier and cheaper to use.

```
//KESCB06  JOB (65040CB06,BOX-85),NEWWAVE,CLASS=A,
//  NOTIFY=KESCB06,MSGCLASS=5
//*[**************************************************************[*
//*[       PROJECT: NEW WAVE 1 LAYOUT                              [*
//*[       ANALYST:                                                [*
//*[    PROGRAMMER: KAREN SMITH                                    [*
//*[          DATE: 12/87                                          [*
//*[                                                               [*
//*[   DESCRIPTION: READ IN WAVE 1 AND OUTPUT WAVE1 WITH A         [*
//*[                WAVE 3 RECORD LAYOUT.  ALL WAVE 1 ONLY         [*
//*[                VARIABLES ARE WRITTEN TO A SEPARATE DATA       [*
//*[                SET.                                           [*
//*[                                                               [*
//*[***************************************************************/
//SIPP EXEC PLIXCLG,CLASS='*',
// REGION.PLI=2000K,
// PARM.PLI='NX,NM,NOESD,ATTRIBUTES(SHORT),NSTG,NOF',
// PARM.LKED='INCLUDE',
// REGION.GO=2000K
//PLI.SYSLIB DD DSN=KESCB06.NEWWAVE.CNTL,DISP=SHR
//PLI.SYSIN DD *

    (SUBSCRIPTRANGE):
 NEWWAVE: PROC OPTIONS(MAIN);
   DCL SYSPRINT EXTERNAL FILE PRINT;
   DCL DDIN              FILE RECORD INPUT;
   DCL DDOUT1            FILE;
   DCL DDOUT2            FILE;

   /****************************************************************
   * INPUT RECORD LAYOUT                                            *
   * THE WAVE 1 RECORD LAYOUT CAN BE INCLUDED FROM THE MACHINE      *
   * READABLE CODEBOOK.                                             *
   *****************************************************************/
   DCL 1 INPUT,
       %include wave1rec;;

   /****************************************************************
   * OUTPUT RECORD LAYOUT                                           *
   * THE WAVE 3 RECORD LAYOUT CAN BE INCLUDED FROM THE MACHINE      *
   * READABLE CODEBOOK.                                             *
   * USE THE WAVE 3 RECORD LAYOUT BECAUSE WAVE 2 HAS PROBLEMS WITH  *
   * THE HOUSEHOLD WEIGHT AND AFDC AMOUNT.                          *
   * THE 120AMT FOR ALL WAVES AFTER CORRECTING THE WAVE 2 120AMT    *
   * STARTS AT COLUMN 4054 AND IS 6 CHARACTERS LONG.                *
   *****************************************************************/
   DCL 1 ALLWAVE_VARS,
       %INCLUDE WAVE3REC;;
   /****************************************************************
   * VARIABLES ONLY ON WAVE 1                                       *
   * ALL VARIABLES ON WAVE 1 AND NOT ON WAVE 3 CAN BE SAVED.        *
   * THIS RECORD LAYOUT CAN BE GENERATED BY MERGING THE WAVE 1 AND  *
   * WAVE 3 VARIABLE NAMES.  ALL NON-MATCHED VARIABLES GET INCLUDED *
   * HERE.                                                          *
   *****************************************************************/
   DCL 1 ONLY_WAVE1,
       %INCLUDE ONLYW1;;

   DCL EOF          BIT INIT('0'B);
   DCL DEBUG        BIT INIT('0'B);
   DCL REC_WRIT     FIXED BIN(31);
   DCL REC_READ     FIXED BIN(31);
   DCL I            FIXED BIN;
   DCL LONG_ZEROS   CHAR(5352) DEFINED ALLWAVE_VARS;
   DCL SHORT_ZEROS   CHAR(180) DEFINED ONLY_WAVE1;
```

```
    ON ENDFILE(DDIN)
      BEGIN;
        EOF = '1'B;
      END;
    ON ERROR BEGIN;
      ON ERROR SYSTEM;
        PUT SKIP LIST(REC_READ,ALLWAVE_VARS.SU_ROT);
      END;

    READ FILE(DDIN) INTO (INPUT);
    DO WHILE (^EOF);
      REC_READ = REC_READ + 1;
      LONG_ZEROS = (5352)'0';
      SHORT_ZEROS = (180)'0';
      INPUT.G1_FILL1 = '0';
      INPUT.G1_FILL2 = '0';
      INPUT.G1_FILL3 = '0';
      INPUT.G1_FILL4 = '0';
      ALLWAVE_VARS = INPUT, BY NAME;
      ONLY_WAVE1   = INPUT, BY NAME;

      WRITE FILE(DDOUT1) FROM (ALLWAVE_VARS);
      WRITE FILE(DDOUT2) FROM (ONLY_WAVE1);

      REC_WRIT = REC_WRIT + 1;
      READ FILE(DDIN) INTO (INPUT);
    END;

  PUT SKIP LIST('RECORDS READ = '||REC_READ
              ||' RECORDS WRITTEN = '||REC_WRIT );

  END NEWWAVE;

//GO.DDIN DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE1.SRTD.MOD01,DISP=SHR
//GO.DDOUT1 DD DSN=CBO.HRCD.SIPP84.WAVE1.NEW.MAS01,
//          DISP=(NEW,CATLG,DELETE),
//          UNIT=TAPE,DCB=(RECFM=FB,LRECL=5352,BLKSIZE=32112),
//          LABEL=(1,SL,EXPDT=99000)
//GO.DDOUT2 DD DSN=CBO.HRCD.SIPP84.WAVE1.ONLY.MAS01,
//          DISP=(NEW,CATLG,DELETE),
//          UNIT=TAPE,DCB=(RECFM=FB,LRECL=180,BLKSIZE=32760),
//          LABEL=(1,SL,EXPDT=99000)
//*GO.DDIN DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE1.SRTD.X200,DISP=SHR
//*GO.DDOUT1 DD DSN=CBO.HRCD.PB7.SIPP84.CBOREC.WAVE1.X200,
//*           DISP=(NEW,CATLG,DELETE),
//*           UNIT=DISK,DCB=(RECFM=FB,LRECL=5352,BLKSIZE=5352),
//*           SPACE=(TRK,(50,20),RLSE)
//*GO.DDOUT2 DD DSN=&&KESOUT2,
//*         DISP=NEW,
//*         UNIT=DISK,DCB=(RECFM=FB,LRECL=180,BLKSIZE=6120),
//*         SPACE=(TRK,(50,20),RLSE)
```

```
//KESCB06 JOB (65040CB06,BOX-85),ASA1,CLASS=W,
//  NOTIFY=KESCB06,MSGCLASS=5 ,TYPRUN=SCAN
//*|***************************************************************|*
//*|       PROJECT: 1989 ASA WINTER CONFERENCE EXAMPLE              |*
//*|       ANALYST: SMITH                                           |*
//*|    PROGRAMMER: SMITH                                           |*
//*|          DATE: 11/88                                           |*
//*|   DESCRIPTION: READ IN ALL WAVES.  REORGANIZE ORIGINAL THE     |*
//*|                STUCTURE AND OUTPUT A PERSON-MONTH              |*
//*|                RECORD WITH NO FILLER.                          |*
//*|                                                                |*
//*|                THE MONTHLY INDEX IS A REFERENCE MONTH.  IT     |*
//*|                RANGES FROM 1 TO 36.                            |*
//*|                                                                |*
//*|                SORT THE FILE TO CREATE A FAMILY-MONTH FILE     |*
//*|                                                                |*
//*|***************************************************************|/
//STEP1 EXEC PLIXCLG,CLASS='*',
//  PARM.PLI='NX,NM,NOMAP,NOESD,ATTRIBUTES(SHORT),NSTG,NOF',
//  PARM.LKED='NX,INCLUDE',
//  REGION.GO=2000K
//PLI.SYSIN DD *
    (SUBSCRIPTRANGE):
 INCOME: PROC OPTIONS(MAIN);
   DCL PTR                POINTER;
   DCL ADDR               BUILTIN;
   DCL SYSPRINT           EXTERNAL FILE PRINT;
   DCL DDIN               FILE RECORD INPUT;
   DCL DDOUT              FILE;

   /***************************************************************
   * INPUT STRUCTURE                                              *
   ***************************************************************/
   DCL 1 INPUT,
     /*============= SAMPLE-UNIT-LEVEL VARIABLES ============*/
     3 SAMPLE_UNIT,
       4 FILLER1            CHAR(5),         /*#    1        */
       4 SUID               CHAR(9),         /*#    6        */
       4 SUROT              PIC'9',          /*#   15        */
       4 FILLER2            CHAR(27),        /*#   16        */
     /*=============== HOUSEHOLD-LEVEL VARIABLES ============*/
     3 HOUSEHOLD(4),
       4 H_ADDID            CHAR(2),         /*#   43        */
       4 FILLER3            CHAR(254),       /*#   45        */
                                             /*#  299        */
     3 HOUSEHOLD_FILLER     CHAR(24),        /*# 1067        */
     /*================ FAMILY-LEVEL VARIABLES ==============*/
     3 FAMILY(4),
       4 F_NUMBR            CHAR(2),         /*# 1091        */
       4 FILLER4            CHAR(112),       /*# 1093        */
                                             /*# 1205        */
     /*============= SUBFAMILY-LEVEL VARIABLES =============*/
     3 SUBFAM(4),
       4 S_NUMBR            CHAR(2),         /*# 1547        */
       4  FILLER1           CHAR(112),       /*# 1549        */
                                             /*# 1661        */
     /*================ PERSON-LEVEL VARIABLES =============*/
     3 PERSON,
       4 FILLER5            CHAR(2),         /*#  2003       */
       4 PP_WAVE            PIC'9',          /*#  2005       */
       4 PP_INTVW           CHAR(1),         /*#  2006       */
       4 PP_MIS(4)          CHAR(1),         /*#  2007       */
       4 PP_MIS5            CHAR(1),         /*#  2011       */
       4 PP_ENTRY           CHAR(2),         /*#  2012       */
       4 PERNUM             CHAR(3),         /*#  2014       */
    4 PP_WGT(4)         CHAR(10),        /*# 2017      */
    4 FILLER6           CHAR(23),        /*# 2057      */
    4 RRP(4)            CHAR(1),         /*# 2080      */
    4 FILLER7           CHAR(13),        /*# 2084      */
    4 MS(4)             CHAR(1),         /*# 2097      */
    4 FILLER8           CHAR(164),       /*# 2101      */
    4 PP_EARN(4)        CHAR(7),         /*# 2265      */
    4 FILLER9           CHAR(2068),      /*# 2293      */
    4 I28AMT(4)         CHAR(5),         /*# 4361      */
    4 FILLER10          CHAR(972);       /*# 4381      */
                                         /*# 5353      */
/*******************************************************************
* OUTPUT STRUCTURE: ALL VARIABLES REQUIRED IN THE EXTRACT ARE      *
*   PUT IN THE OUTPUT STRUCTURE.  THE ARRAY DIMENSION IS PUT ONLY *
*   ON THE OUTPUT DECLARATION.  AT THE ASSIGNMENT, ALL VARIABLES  *
*   BECOME MONTHLY.                                                *
* NOTE: TAKE CARE WHEN ASSIGNING VARIABLE TYPES.  THE PIC FORMAT  *
*   IS USEFUL FOR CONCATENATING IN A CHARACTER STRING BUT DOES    *
*   NOT INTERPRET NEGATIVE SIGNS PROPERLY.  CONVERTING VARIABLES  *
*   TO ANY NUMERIC TYPE  IS VERY EXPENSIVE AND INTRODUCES PADDING *
*   PROBLEMS.                                                      *
*******************************************************************/
DCL 1 OUTPUT(4),
    4 INDEX             PIC'99',         /*# CREATED VARIABLE */
    4 SUID              CHAR(9),         /*#    6      */
    4 H_ADDID           CHAR(2),         /*#   43      */
    4 F_NUMBR           CHAR(2),         /*# 1091      */
    4 S_NUMBR           CHAR(2),         /*# 1547      */
    4 PP_ENTRY          CHAR(2),         /*# 2012      */
    4 PERNUM            CHAR(3),         /*# 2014      */
    4 PP_INTVW          CHAR(1),         /*# 2006      */
    4 PP_MIS            CHAR(1),         /*# 2007      */
    4 PP_MIS5           CHAR(1),         /*# 2011      */
    4 PP_WGT            CHAR(10),        /*# 2017      */
    4 RRP               CHAR(1),         /*# 2080      */
    4 MS                CHAR(1),         /*# 2097      */
    4 PP_EARN           CHAR(7),         /*# 2265      */
    4 I28AMT            CHAR(5);         /*# 4361      */

DCL EOF                 BIT INIT('0'B);
DCL REC_WRIT            FIXED BIN(31);
DCL REC_READ            FIXED BIN(31);
DCL THROW_OUT(9,4)      FIXED BIN(31) INIT((36)0);
DCL I                   FIXED BIN;
DCL J                   FIXED BIN;

ON ENDFILE(DDIN)
  BEGIN;
    EOF = '1'B;
  END;

READ FILE(DDIN) INTO (INPUT);

DO WHILE (^EOF);
  REC_READ = REC_READ + 1;

  OUTPUT       = INPUT.SAMPLE_UNIT, BY NAME;
  OUTPUT       = INPUT.HOUSEHOLD, BY NAME;
  OUTPUT       = INPUT.FAMILY, BY NAME;
  OUTPUT       = INPUT.SUBFAM, BY NAME;
  OUTPUT       = INPUT.PERSON, BY NAME;
```

```
    /***********************************************************
    * RECODE VARIABLES.                                        *
    * THE MISSING NOTATION IN THE INCOME FIELDS (-0009) CAUSES *
    * PROBLEMS LATER WHEN SUMMING INCOME.  RESET THESE VARIABLES *
    * TO 0S.                                                   *
    ***********************************************************/
    DO I = 1 TO 4;
      IF OUTPUT.I28AMT(I) = '-0009'
        THEN OUTPUT.I28AMT(I) = '00000';
    END;

    /***********************************************************
    * CALCULATE MONTHLY INDEX AND WRITE OUT THE PERSON-MONTH FILE *
    ***********************************************************/
    CALL WRITE_FILE;

    READ FILE(DDIN) INTO (INPUT);
  END;

PUT SKIP LIST('RECORDS READ = '||REC_READ);
PUT SKIP LIST('RECORDS WRITTEN'||REC_WRIT);
PUT SKIP EDIT('RECORDS THROWN OUT')(A);
PUT SKIP EDIT('WAVE', 'ROT 1', 'ROT 2', 'ROT 3', 'ROT 4')
          (5(A(8),X(2)));
DO I = 1 TO 9;
  PUT EDIT (I, (THROW_OUT(I,J) DO J = 1 TO 4))(SKIP,5(F(8),X(2)));
END;


%PAGE;
/**************************************************************
* WRITE_FILE:                                                 *
*   OUTPUT A PERSON-MONTH FILE FROM THE RECTANGULAR WAVE FILES. *
**************************************************************/
WRITE_FILE:PROC;
  DCL WAVE PIC'9';
  /************************************************************
  * LOOP THROUGH MONTHS AND ASSIGN INDEX(I).                  *
  * FOR REFERENCE MONTHS THE FUNCTION IS:                     *
  *    INDEX(I) = (4*(WAVE-1)+I).                             *
  * FOR CALANDER MONTHS THE FUNCTION IS:                      *
  *    INDEX(I) = (4*(WAVE-1)+I) - (4-INPUT.SUROT).           *
  * THIS EXAMPLE CREATES REFERENCE MONTHS.                    *
  ************************************************************/
  DO I = 1 TO 4;
    /*********************************************************
    * ADJUST THE WAVE VALUE FOR MISSED WAVES.                *
    *   ROTATION 3 IS MISSING WAVE 8.                        *
    *   ROTATION 4 IS MISSING WAVE 2.                        *
    *********************************************************/
    SELECT;
      WHEN (INPUT.SUROT=3 AND INPUT.PP_WAVE=9) WAVE= 8;
      WHEN (INPUT.SUROT=4 AND INPUT.PP_WAVE>1) WAVE=INPUT.PP_WAVE-1;
      OTHERWISE WAVE = INPUT.PP_WAVE;
    END;

    /*********************************************************
    * CREATE A REFERENCE MONTH INDEX.                        *
    *********************************************************/
    INDEX(I) = (4*(WAVE-1)+I);

    /*********************************************************
    * ONLY OUTPUT MONTHS WITH A WEIGHT GREATER THAN 0.       *
    *********************************************************/
    IF OUTPUT.PP_WGT(I) > '0000000000'
```

```
       THEN DO;
            WRITE FILE(DDOUT) FROM (OUTPUT(I));
            REC_WRIT = REC_WRIT + 1;
            END;
       ELSE THROW_OUT(INPUT.PP_WAVE,INPUT.SUROT) =
            THROW_OUT(INPUT.PP_WAVE,INPUT.SUROT) + 1;
   END; /* I LOOP */
 END  WRITE_FILE;
 END INCOME;
//**TESTFILE**
//*GO.DDIN DD DSN=CBO.HRCD.SIPP84.WAVE1.NEW.X200,DISP=SHR
//*         DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE3.X200,DISP=SHR
//*         DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE4.X200,DISP=SHR
//*GO.DDOUT DD DSN=&&KESCB06,DISP=(NEW,PASS),
//*            UNIT=DISK,DCB=(RECFM=FB,LRECL=49,BLKSIZE=6223),
//*            SPACE=(TRK,(20,20),RLSE)
//*SORT EXEC DISCSORT,RGN=800K,COND=(9,LT)
//*SORT.SYSOUT DD SYSOUT=*
//*SORT.SORTIN DD DSN=&&KESCB06,DISP=(OLD,PASS)
//*SORT.SORTOUT DD DSN=CBO.HRCD.P88.SIPPASA.WAVE1-9.FMX200,
//**      DISP=(NEW,CATLG,DELETE),
//*       UNIT=DISK,DCB=(RECFM=FB,LRECL=49,BLKSIZE=6223),
//*       SPACE=(TRK,(20,20),RLSE)
//**ENDTEST**
//**BIGFILE**
//GO.DDIN DD DSN=CBO.HRCD.SIPP84.WAVE1.NEW.MAS01,DISP=SHR
//          DD DSN=CBO.HRCD.SIPP84.WAVE2.NEW.MAS01,DISP=SHR,
//             UNIT=AFF=DDIN
//          DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE3.MAS01,DISP=SHR,
//           . UNIT=AFF=DDIN
//          DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE4.MAS01,DISP=SHR,
//             UNIT=AFF=DDIN
//          DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE5.MAS01,DISP=SHR,
//             UNIT=AFF=DDIN
//          DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE6.MAS01,DISP=SHR,
//             UNIT=AFF=DDIN
//          DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE7.MAS01,DISP=SHR,
//             UNIT=AFF=DDIN
//          DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE8.MAS01,DISP=SHR,
//             UNIT=AFF=DDIN
//          DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE9.MAS01,DISP=SHR,
//             UNIT=AFF=DDIN
//GO.DDOUT DD DSN=&&KESCB06,DISP=(NEW,PASS),
//            UNIT=TAPE,DCB=(RECFM=FB,LRECL=49,BLKSIZE=32732),
//            LABEL=(1,SL,EXPDT=99000)
//SORT EXEC DISCSORT,RGN=800K,COND=(9,LT)
//SORT.SYSOUT DD SYSOUT=*
//SORT.SORTWK01 DD SPACE=(CYL,(20,20)),UNIT=SYSDA
//SORT.SORTWK02 DD SPACE=(CYL,(20,20)),UNIT=SYSDA
//SORT.SORTWK03 DD SPACE=(CYL,(20,20)),UNIT=SYSDA
//SORT.SORTWK04 DD SPACE=(CYL,(20,20)),UNIT=SYSDA
//SORT.SORTIN DD DSN=&&KESCB06,DISP=(OLD,PASS)
//SORT.SORTOUT DD DSN=CBO.HRCD.P88.SIPPASA.WAVE1-9.FM01,
//       DISP=(NEW,CATLG,DELETE),
//       UNIT=TAPE,DCB=(RECFM=FB,LRECL=49,BLKSIZE=32732),
//       LABEL=(1,SL,EXPDT=99000)
//**ENDBIG**
//*|**************************************************************|*
//*| SORT BY SUID,INDEX,H_ADDID,F_NUMBR,S_NUMBR                    |*
//*| THIS CREATES A FAMILY MONTH SORTED FILE WITH SORTED           |*
//*| SUBFAMILIES APPEARING AFTER THE PRIMARY FAMILY.               |*
//*|**************************************************************/
//SORT.SYSIN DD *
  SORT FIELDS=(3,9,CH,A,1,2,CH,A,12,6,CH,A),SIZE=E1688328
```

```
//KESCB06 JOB (65040CB06,BOX-85),ASA2,CLASS=A,
//  NOTIFY=KESCB06,MSGCLASS=5 ,TYPRUN=SCAN
//*|***************************************************************|*
//*|                                                               |*
//*|      PROJECT: 1989 ASA WINTER CONFERENCE EXAMPLE              |*
//*|      ANALYST: SMITH                                           |*
//*|   PROGRAMMER: SMITH                                           |*
//*|         DATE: 11/88                                           |*
//*|                                                               |*
//*|  DESCRIPTION: THE INPUT IS FROM ASA1.                         |*
//*|               CALCULATE FAMILY VARIABLES AND OUTPUT PERSON    |*
//*|               RECORDS WITH THE NEW FAMILY VARIABLES ATTACHED. |*
//*|               SORT BY PERSON INDEX TO CREATE A PERSON-MONTH   |*
//*|               SORTED FILE.                                    |*
//*|***************************************************************|/
//SIPP EXEC PLIXCLG,CLASS='*',COND=(5,LE),
// PARM.PLI='NX,NM,NOMAP,NOESD,ATTRIBUTES(SHORT),NSTG,NOF',
// PARM.LKED='NX',
// REGION.GO=2000K
//PLI.SYSIN DD *
   (SUBSCRIPTRANGE):
 EXTRACT: PROC OPTIONS(MAIN);

 DCL 1 FAM_ARRAY(30),
      2 PERSON_MONTH,
       4 INDEX            PIC'99',        /*# CREATED VARIABLE */
       4 SUID             CHAR(9),        /*#     6          */
       4 H_ADDID          CHAR(2),        /*#    43          */
       4 F_NUMBR          CHAR(2),        /*#  1091          */
       4 S_NUMBR          CHAR(2),        /*#  1547          */
       4 PP_ENTRY         CHAR(2),        /*#  2012          */
       4 PERNUM           CHAR(3),        /*#  2014          */
       4 PP_INTVW         CHAR(1),        /*#  2006          */
       4 PP_MIS           CHAR(1),        /*#  2007          */
       4 PP_MIS5          CHAR(1),        /*#  2011          */
       4 PP_WGT           CHAR(10),       /*#  2017          */
       4 RRP              CHAR(1),        /*#  2080          */
       4 MS               CHAR(1),        /*#  2097          */
       4 PP_EARN          PIC'(7)9',      /*#  2265          */
       4 I28AMT           PIC'(5)9',      /*#  4361          */
     2 NEW_VARS,
       3 FAM_CHILD_SUPPORT PIC'(7)9',
       3 FAM_EARNING       PIC'(7)9';

 %PAGE;
 DCL DDIN              FILE RECORD INPUT;
 DCL DDOUT             FILE RECORD OUTPUT;
 DCL EOF               BIT INIT('0'B);
 DCL FAM_COUNT         FIXED BIN INIT(0);
 DCL FAMILY_ID         CHAR(15);
 DCL FAMILY_NUMBER     FIXED BIN(31) INIT(0);
 DCL HBOUND            BUILTIN;
 DCL I                 FIXED BIN;
 DCL J                 FIXED BIN;
 DCL K                 FIXED BIN;
 DCL LAST_FAMILY       CHAR(15);
 DCL REC_READ          FIXED BIN(31) INIT(0);
 DCL REC_WRIT          FIXED BIN(31) INIT(0);
 DCL SYSPRINT          EXTERNAL FILE PRINT;

 ON ERROR BEGIN;
   ON ERROR SYSTEM;
     PUT SKIP DATA(REC_READ, FAM_COUNT, I, J, K);
 END;
```

```
ON ENDFILE(DDIN)
 BEGIN;
  EOF = '1'B;
  REC_READ = REC_READ - 1;
 END;


CALL READER;
FAM_COUNT = 0; /* DON'T MOVE FIRST RECORD */
CALL RESET_FAM_ARRAY;

DO WHILE (^EOF);
  /***************************************************************
  * IF THE CURRENT PERSON IS IN A NEW FAMILY THEN PROCESS         *
  * THE FAMILY IN THE RECORD BUFFER.                              *
  ***************************************************************/
  IF (LAST_FAMILY ^= FAMILY_ID OR FAM_COUNT=30)
    THEN DO;
      CALL PROCESS_FAMILY;
      CALL RESET_FAM_ARRAY;
    END;

  /***************************************************************
  * READ NEXT PERSON INTO THE INPUT BUFFER.                       *
  ***************************************************************/
  CALL READER;

END;                                /*----WHILE ^EOF----*/

/****************************************************************
* AT END OF FILE,  PROCESS THE LAST FAMILY.                      *
****************************************************************/
CALL PROCESS_FAMILY;

/**************************************************************
* PRINT OUT RUN SUMMARY                                        *
**************************************************************/
PUT SKIP LIST('RECORDS READ: '||REC_READ);
PUT SKIP LIST('RECORDS WRITTEN: '||REC_WRIT);
PUT SKIP LIST('NUMBER OF FAMILIES: '||FAMILY_NUMBER);


%PAGE;
/**************************************************************
* READER: READ IN A NEW PERSON AND ASSIGN THEIR ID.            *
*         EACH PERSON IN THE FAMILY IS PUT INTO A FAMILY ARRAY. *
*         TO PUT SUBFAMILIES INTO SEPARATE UNITS, ADD S_NUMBR  *
*         TO THE FAMILY_ID.                                    *
**************************************************************/
READER: PROC;
  FAM_COUNT = FAM_COUNT + 1;
  READ FILE(DDIN) INTO(PERSON_MONTH(FAM_COUNT));
  FAMILY_ID = SUID(FAM_COUNT)
          ||INDEX(FAM_COUNT)
          ||H_ADDID(FAM_COUNT)
          ||F_NUMBR(FAM_COUNT);

  REC_READ = REC_READ + 1;
END READER;
```

```
/*****************************************************************
* RESET_FAM_ARRAY: EACH PERSON IS READ INTO THE FAMILY ARRAY      *
*            INCREMENTALLY.  THE FIRST PERSON IN EACH NEW         *
*            FAMILY IS THE LAST PERSON INPUT INTO THE ARRAY.      *
*            THAT PERSON NEEDS TO BE MOVED TO THE FIRST ARRAY     *
*            LOCATION AT THE BEGINNING OF EACH FAMILY SETUP.      *
*****************************************************************/
RESET_FAM_ARRAY: PROCEDURE;
  FAM_ARRAY(1) = FAM_ARRAY(FAM_COUNT+1);
  FAM_COUNT = 1;
  LAST_FAMILY = FAMILY_ID;
END RESET_FAM_ARRAY;

/*****************************************************************
* PROCESS_FAMILY:                                                 *
*   PROCESS EACH RECORD IN THE BUFFER TO AGGREGATE FAMILY INCOME  *
*   BY INCOME SOURCE.                                             *
*   WRITE OUT EACH PERSON WITH THE NEW FAMILY VARIABLES ATTACHED. *
*****************************************************************/
PROCESS_FAMILY: PROCEDURE;
 FAM_COUNT = FAM_COUNT - 1;
 FAMILY_NUMBER = FAMILY_NUMBER + 1;

 /****************************************************************
 * INITIALIZE FAMILY VARIABLES TO 0.                              *
 * LOOP THROUGH ALL PEOPLE IN THE FAMILY TO CALCULATE THE FAMILY  *
 * VARIABLES.                                                     *
 ****************************************************************/
 NEW_VARS = 0;
 DO I = 1 TO FAM_COUNT;
   FAM_CHILD_SUPPORT(1) = FAM_CHILD_SUPPORT(1) + I28AMT(I);
   FAM_EARNING(1) = FAM_EARNING(1) + PP_EARN(I);
 END; /* I LOOP */

 /****************************************************************
 * COPY UNIT INFORMATION FROM THE FIRST PERSON IN THE UNIT TO     *
 * EVERYONE ELSE IN THE UNIT.                                     *
 ****************************************************************/
 DO I = 2 TO FAM_COUNT;
   NEW_VARS(I) = NEW_VARS(1);
 END;

 /****************************************************************
 * WRITE OUT EACH PERSON WITH THE NEW FAMILY VARIABLES ATTACHED.  *
 ****************************************************************/
 DO I = 1 TO FAM_COUNT;
   WRITE FILE(DDOUT) FROM(FAM_ARRAY(I));
   REC_WRIT = REC_WRIT + 1;
 END;
END PROCESS_FAMILY;
END EXTRACT;
//**TESTFILE**
//*GO.DDIN DD DSN=CBO.HRCD.PBB.SIPPASA.WAVE1-9.FMX200,DISP=SHR
//*GO.DDOUT DD DSN=&&FAMVARS,
//*        DISP=(NEW,PASS),
//*        UNIT=DISK,DCB=(RECFM=FB,LRECL=63,BLKSIZE=6174),
//*        SPACE=(TRK,(50,20),RLSE)
//*SORT EXEC DISCSORT,RGN=800K,COND=(9,LT)
//*SORT.SYSOUT DD SYSOUT=*
//*SORT.SORTIN DD DSN=&&FAMVARS,DISP=(OLD,PASS)
//*SORT.SORTOUT DD DSN=CBO.HRCD.PBB.SIPPASA.WAVE1-9.PMX200,
//*       DISP=(NEW,CATLG,DELETE),
//*       UNIT=DISK,DCB=(RECFM=FB,LRECL=63,BLKSIZE=6174),
//*       SPACE=(TRK,(20,20),RLSE)
//**ENDTEST**
```

```
//**BIGFILE**
//GO.DDIN DD DSN=CBO.NRCD.P88.SIPPASA.WAVE1-9.FM01,DISP=SHR
//GO.DDOUT DD DSN=&&FAMVARS,
//           DISP=(NEW,PASS),
//           UNIT=TAPE,DCB=(RECFM=FB,LRECL=63,BLKSIZE=32760),
//           LABEL=(1,SL,EXPDT=99000)
//SORT EXEC DISCSORT,RGN=800K,COND=(9,LT)
//SORT.SYSOUT DD SYSOUT=*
//SORT.SORTWK01 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTWK02 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTWK03 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTWK04 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTWK05 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTWK06 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTWK07 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTWK08 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTWK09 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTWK10 DD SPACE=(CYL,(50,50)),UNIT=SYSDA
//SORT.SORTIN DD DSN=&&FAMVARS,DISP=(OLD,PASS)
//SORT.SORTOUT DD DSN=CBO.NRCD.P88.SIPPASA.WAVE1-9.PM01,
//       DISP=(NEW,CATLG,DELETE),
//       UNIT=TAPE,DCB=(RECFM=FB,LRECL=63,BLKSIZE=32760),
//       LABEL=(1,SL,EXPDT=99000)
//**ENDBIG**
//*********************************************************************
//****   SORT BY SUID,PP_ENTRY,PERNUM,INDEX                        ****
//****   OUTPUT IS A PERSON MONTH FILE                             ****
//*********************************************************************
//SORT.SYSIN DD *
      SORT  FIELDS=(3,9,CH,A,18,5,CH,A,1,2,CH,A),SIZE=E1688328
```

```
//KESCB06  JOB (65540CB06,BOX-85),ASA3,CLASS=B,
//  NOTIFY=KESCB06,MSGCLASS=5
//*|*****************************************************************|*
//*|                                                                 |*
//*|       PROJECT: ASA WINTER CONFERENCE                            |*
//*|       ANALYST: SMITH                                            |*
//*|    PROGRAMMER: SMITH                                            |*
//*|          DATE: 11/88                                            |*
//*|                                                                 |*
//*|   DESCRIPTION: READ IN PEOPLE FROM THE OUTPUT FILE CREATED      |*
//*|                IN ASA2.                                         |*
//*|                PRINT OUT A 36 MONTH LIST OF PERSON EARNINGS     |*
//*|                AND FAMILY EARNINGS.                             |*
//*|*****************************************************************/
//SIPP EXEC PLIXCLG,CLASS='*',
//  PARM.PLI='NX,NM,NOMAP,NOESD,ATTRIBUTES(SHORT),NSTG,NOF',
//  PARM.LKED='NX,INCLUDE',
//  REGION.GO=2000K
//PLI.SYSIN DD *
    (SUBSCRIPTRANGE):
 EXTRACT: PROC OPTIONS(MAIN);

 DCL 1 PERSON_ARRAY(37),
      2 PERSON_MONTH,
       4 INDEX              PIC'99',          /*# CREATED VARIABLE */
       4 SUID               CHAR(9),          /*#      6        */
       4 H_ADDID            CHAR(2),          /*#     43        */
       4 F_NUMBR            CHAR(2),          /*#   1091        */
       4 S_NUMBR            CHAR(2),          /*#   1547        */
       4 PP_ENTRY           CHAR(2),          /*#   2012        */
       4 PERNUM             CHAR(3),          /*#   2014        */
       4 PP_INTVW           CHAR(1),          /*#   2006        */
       4 PP_MIS             CHAR(1),          /*#   2007        */
       4 PP_MIS5            CHAR(1),          /*#   2011        */
       4 PP_WGT             CHAR(10),         /*#   2017        */
       4 RRP                CHAR(1),          /*#   2080        */
       4 MS                 CHAR(1),          /*#   2097        */
       4 PP_EARN            PIC'(7)9',        /*#   2265        */
       4 I28AMT             PIC'(5)9',        /*#   4361        */
     2 NEW_VARS,
       3 FAM_CHILD_SUPPORT PIC'(7)9',
       3 FAM_EARNING       PIC'(7)9';

 DCL DDIN                 FILE RECORD INPUT;
 DCL DDOUT                FILE RECORD OUTPUT;
 DCL EOF                  BIT INIT('0'B);
 DCL PER_COUNT            FIXED BIN INIT(0);
 DCL PERSON_ID            CHAR(14);
 DCL PERSON_NUMBER        FIXED BIN(31) INIT(0);
 DCL HBOUND               BUILTIN;
 DCL I                    FIXED BIN;
 DCL J                    FIXED BIN;
 DCL K                    FIXED BIN;
 DCL LAST_PERSON          CHAR(14);
 DCL REC_READ             FIXED BIN(31) INIT(0);
 DCL SYSPRINT             EXTERNAL FILE PRINT;

 ON ERROR BEGIN;
   ON ERROR SYSTEM;
     PUT SKIP DATA(REC_READ, PER_COUNT, I, J, K);
 END;

 ON ENDFILE(DDIN)
  BEGIN;
   EOF = '1'B;
  REC_READ = REC_READ - 1;
 END;

CALL READER;
PER_COUNT = 0; /* DON'T MOVE FIRST RECORD */
CALL RESET_PERSON_ARRAY;

DO WHILE (^EOF);
  /*****************************************************************
   * IF THE CURRENT PERSON IS IN A NEW PERSON THEN PROCESS          *
   * THE PERSON IN THE RECORD BUFFER.                               *
   *****************************************************************/
  IF (LAST_PERSON ^= PERSON_ID OR PER_COUNT=37)
    THEN DO;
      CALL PROCESS_PERSON;
      CALL RESET_PERSON_ARRAY;
    END;

  /*****************************************************************
   * READ NEXT PERSON INTO THE INPUT BUFFER.                        *
   *****************************************************************/
  CALL READER;
END; /*----WHILE ^EOF----*/

/*****************************************************************
* AT END OF FILE,  PROCESS THE LAST PERSON.                       *
*****************************************************************/
CALL PROCESS_PERSON;

/****************************************************************
* PRINT OUT A RUN SUMMARY.                                       *
****************************************************************/
PUT SKIP LIST('RECORDS READ: '||REC_READ);
PUT SKIP LIST('NUMBER OF PEOPLE: '||PERSON_NUMBER);

%PAGE;
/****************************************************************
* READER: READ IN A NEW PERSON AND ASSIGN THEIR ID.              *
*         EACH PERSON IS PUT INTO A PERSON ARRAY.                *
****************************************************************/
READER: PROC;
  PER_COUNT = PER_COUNT + 1;
  READ FILE(DDIN) INTO(PERSON_ARRAY(PER_COUNT));
  PERSON_ID = SUID(PER_COUNT)
           ||PP_ENTRY(PER_COUNT)
           ||PERNUM(PER_COUNT);

  REC_READ = REC_READ + 1;
END READER;

/****************************************************************
* RESET_PERSON_ARRAY: EACH PERSON IS READ INTO THE PERSON ARRAY  *
*             INCREMENTALLY.  THE FIRST PERSON RECORD FOR EACH NEW *
*             PERSON IS THE LAST PERSON INPUT INTO THE ARRAY.    *
*             THAT PERSON NEEDS TO BE MOVED TO THE FIRST ARRAY   *
*             LOCATION AT THE BEGINNING OF EACH PERSON SETUP.    *
****************************************************************/
RESET_PERSON_ARRAY: PROCEDURE;
  PERSON_ARRAY(1) = PERSON_ARRAY(PER_COUNT+1);
  PER_COUNT = 1;
  LAST_PERSON = PERSON_ID;
END RESET_PERSON_ARRAY;
```

```
/*****************************************************************
* PROCESS_PERSON                                                 *
*   PROCESS EACH RECORD IN THE BUFFER.                           *
*   THIS MODULE IS WHERE YOU WOULD DO THE ANALYSIS.              *
*****************************************************************/
PROCESS_PERSON: PROCEDURE;
 PER_COUNT = PER_COUNT - 1;
 PERSON_NUMBER = PERSON_NUMBER + 1;
 END;
END PROCESS_PERSON;
END EXTRACT;

//**TESTFILE**
//*GO.DDIN DD DSN=CBO.HRCD.P88.SIPPASA.WAVE1-9.PMX200,DISP=SHR
//**ENDTEST**
//**BIGFILE**
//GO.DDIN DD DSN=CBO.HRCD.P88.SIPPASA.WAVE1-9.PM01,DISP=SHR
//**ENDBIG**
```

```
//KESCB06 JOB (65040CB06,BOX-85),ASASAS1,CLASS=A,
//  NOTIFY=KESCB06,MSGCLASS=5 ,TYPRUN=SCAN
//*[***************************************************************[*
//*[                                                                [*
//*[       PROJECT: 1989 ASA WINTER CONFERENCE EXAMPLE             [*
//*[       ANALYST: SMITH                                           [*
//*[    PROGRAMMER: SMITH                                           [*
//*[          DATE: 11/88                                           [*
//*[                                                                [*
//*[   DESCRIPTION: READS IN ALL WAVES.  REORGANIZE ORIGINAL        [*
//*[                STUCTURE AND OUTPUT FINAL PERSON MONTH          [*
//*[                RECORD WITH NO FILLER.                          [*
//*[                                                                [*
//*[                THE MONTHLY INDEX IS A REFERENCE MONTH.  IT     [*
//*[                RANGES FROM 1 TO 36.                            [*
//*[                                                                [*
//*[                SORT THE FILE TO CREATE A FAMILY MONTH FILE     [*
//*[                                                                [*
//*[***************************************************************/
//SASCBK EXEC SAS,CLASS='*',REGION=5000K,COND=(5,LE)
//WORK     DD UNIT=SYSDA,SPACE=(CYL,(100,100))
//**TESTFILE**
//*DDIN DD DSN=CBO.HRCD.SIPP84.WAVE1.NEW.X200,DISP=SHR
//*     DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE3.X200,DISP=SHR
//*     DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE4.X200,DISP=SHR
//*DDOUT1 DD DSN=&&KESCB06,
//*        DISP=(NEW,PASS),
//*        UNIT=DISK,SPACE=(TRK,(20,20),RLSE)
//*DDOUT2 DD DSN=CBO.HRCD.P88.SIPPASA.WAVE1-9.SASFMX,
//*        DISP=(NEW,CATLG,DELETE),
//*        UNIT=DISK,SPACE=(TRK,(20,20),RLSE)
//**ENDTEST**
//**BIGFILE**
//DDIN DD DSN=CBO.HRCD.SIPP84.WAVE1.NEW.MAS01,DISP=SHR
//        DD DSN=CBO.HRCD.SIPP84.WAVE2.NEW.MAS01,DISP=SHR,
//           UNIT=AFF=DDIN
//        DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE3.MAS01,DISP=SHR,
//           UNIT=AFF=DDIN
//        DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE4.MAS01,DISP=SHR,
//           UNIT=AFF=DDIN
//        DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE5.MAS01,DISP=SHR,
//           UNIT=AFF=DDIN
//        DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE6.MAS01,DISP=SHR,
//           UNIT=AFF=DDIN
//        DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE7.MAS01,DISP=SHR,
//           UNIT=AFF=DDIN
//        DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE8.MAS01,DISP=SHR,
//           UNIT=AFF=DDIN
//        DD DSN=CBO.HRCD.SIPP.YEAR84.WAVE9.MAS01,DISP=SHR,
//           UNIT=AFF=DDIN
//DDOUT1 DD DSN=&&KESCB06,DISP=(NEW,PASS),
//      UNIT=TAPE,LABEL=(1,SL,EXPDT=99000)
//SORTWK01 DD SPACE=(CYL,(20,20)),UNIT=SYSDA
//SORTWK02 DD SPACE=(CYL,(20,20)),UNIT=SYSDA
//SORTWK03 DD SPACE=(CYL,(20,20)),UNIT=SYSDA
//SORTWK04 DD SPACE=(CYL,(20,20)),UNIT=SYSDA
//DDOUT2 DD DSN=CBO.HRCD.P88.SIPPASA.WAVE1-9.SASFM01,
//      DISP=(NEW,CATLG,DELETE),
//      UNIT=TAPE,LABEL=(1,SL,EXPDT=99000)
//**ENDBIG**
//SYSIN DD *
 OPTIONS MPRINT SYMBOLGEN MLOGIC MACROGEN NOCENTER;
```

```
/****************************************************************
* RECODE MISSING NOTATION (-0009) FROM INCOME FIELDS           *
****************************************************************/
%MACRO RECODE;
   %DO I = 1 %TO 4;
      IF I28AMT&I = '-0009'
        THEN I28AMT&I = '00000';
   %END;
%MEND;

/****************************************************************
* WRITER:                                                       *
* OUTPUT    A PERSON MONTH FILE FROM THE WAVE FILE.             *
*                                                               *
* LOOP THROUGH MONTHS AND ASSIGN INDEX(I)                       *
*                                                               *
* FOR REFERENCE MONTHS THE FUNCTION IS:                         *
*    INDEX(I) = (4*(WAVE-1)+I)                                  *
*                                                               *
* FOR CALANDER MONTHS THE FUNCTION IS:                          *
*    INDEX(I) = (4*(WAVE-1)+I) - (4-SUROT)                      *
*                                                               *
* THIS EXAMPLE CREATES REFERENCE MONTHS                         *
*                                                               *
****************************************************************/
%MACRO WRITER;
 %DO I = 1 %TO 4;
    /************************************************************
    * ADJUST THE WAVE VALUE FOR MISSED WAVES.                   *
    *   ROTATION 3 IS MISSING WAVE 8                            *
    *   ROTATION 4 IS MISSING WAVE 2                            *
    ************************************************************/
    SELECT;
      WHEN (SUROT=3 AND PP_WAVE=9) WAVE= 8;
      WHEN (SUROT=4 AND PP_WAVE>1) WAVE=PP_WAVE-1;
      OTHERWISE WAVE = PP_WAVE;
    END;

    /************************************************************
    * CREATE REFERENCE MONTH INDEX                              *
    ************************************************************/
    INDEX = (4*(WAVE-1)+&I);

    /************************************************************
    * ONLY OUTPUT MONTHS WITH A POSITIVE WEIGHT                 *
    ************************************************************/
    IF WGT&I > '0000000000'
      THEN DO;
        HADDID = HADDID&I;
        FNUMBR = FNUMBR&I;
        SNUMBR = SNUMBR&I;
        MIS    = MIS&I;
        WGT    = WGT&I;
        RRP    = RRP&I;
        MS     = MS&I;
        EARN   = EARN&I;
        I28AMT = I28AMT&I;

        OUTPUT;
    END; /* WEIGHT CHECK */
  %END;
%MEND WRITER;
```

```
DATA DDOUT1.PEOPLE(
  KEEP= INDEX      SUID       HADDID      FNUMBR
        SNUMBR     INTVW      MIS         MISI
        ENTRY      PERNUM     WGT         RRP
        MS         EARN       I28AMT);

  INFILE DDIN;
  INPUT
  @   6 SUID              $CHAR9.        /*#    6        */
  @  15 SUROT                  1.        /*#   15        */
  @  43 HADDID1           $CHAR2.        /*#   43        */
  @ 299 HADDID2           $CHAR2.        /*#  299        */
  @ 555 HADDID3           $CHAR2.        /*#  555        */
  @ 811 HADDID4           $CHAR2.        /*#  811        */
  @1091 FNUMBR1           $CHAR2.        /*# 1091        */
  @1205 FNUMBR2           $CHAR2.        /*# 1205        */
  @1319 FNUMBR3           $CHAR2.        /*# 1319        */
  @1433 FNUMBR4           $CHAR2.        /*# 1433        */
  @1547 SNUMBR1           $CHAR2.        /*# 1547        */
  @1661 SNUMBR2           $CHAR2.        /*# 1661        */
  @1775 SNUMBR3           $CHAR2.        /*# 1775        */
  @1889 SNUMBR4           $CHAR2.        /*# 1889        */
  @2005 PP_WAVE                1.        /*#  2005       */
  @2006 INTVW             $CHAR1.        /*#  2006       */
  @2007 MIS1              $CHAR1.        /*#  2007       */
  @2008 MIS2              $CHAR1.        /*#             */
  @2009 MIS3              $CHAR1.        /*#             */
  @2010 MIS4              $CHAR1.        /*#             */
  @2011 MISI              $CHAR1.        /*#  2011       */
  @2012 ENTRY             $CHAR2.        /*#  2012       */
  @2014 PERNUM            $CHAR3.        /*#  2014       */
  @2017 WGT1              $CHAR10.       /*#  2017       */
  @2027 WGT2              $CHAR10.       /*#             */
  @2037 WGT3              $CHAR10.       /*#             */
  @2047 WGT4              $CHAR10.       /*#             */
  @2080 RRP1              $CHAR1.        /*#  2080       */
  @2081 RRP2              $CHAR1.        /*#             */
  @2082 RRP3              $CHAR1.        /*#             */
  @2083 RRP4              $CHAR1.        /*#             */
  @2097 MS1               $CHAR1.        /*#  2097       */
  @2098 MS2               $CHAR1.        /*#             */
  @2099 MS3               $CHAR1.        /*#             */
  @2100 MS4               $CHAR1.        /*#             */
  @2265 EARN1                  7.        /*#  2265       */
  @2272 EARN2                  7.        /*#             */
  @2279 EARN3                  7.        /*#             */
  @2286 EARN4                  7.        /*#             */
  @4361 I28AMT1                5.        /*#  4361       */
  @4366 I28AMT2                5.        /*#             */
  @4371 I28AMT3                5.        /*#             */
  @4376 I28AMT4                5.        /*#             */
     ;

    %RECODE;

    /*************************************************************
    * CALCULATE MONTHLY INDEX AND WRITE OUT THE PERSON-MONTH FILE *
    *************************************************************/
    %WRITER;

PROC SORT DATA=DDOUT1.PEOPLE OUT=DDOUT2.PEOPLE;
  BY SUID INDEX HADDID FNUMBR SNUMBR;

PROC PRINT DATA=DDOUT2.PEOPLE (OBS=100);
```

```
//KESCB06 JOB (65040CB06,BOX-85),ASASAS2,CLASS=E,
//  NOTIFY=KESCB06,MSGCLASS=5 ,TYPRUN=SCAN
//*[*************************************************************[*
//*[                                                              [*
//*[        PROJECT: 1989 ASA WINTER CONFERENCE EXAMPLE           [*
//*[        ANALYST: SMITH                                        [*
//*[     PROGRAMMER: SMITH                                        [*
//*[           DATE: 11/88                                        [*
//*[                                                              [*
//*[    DESCRIPTION: INPUT IS FROM ASASAS1.                       [*
//*[                 CALCULATE FAMILY VARIABLES AND OUTPUT PERSON [*
//*[                 RECORDS WITH THE NEW FAMILY VARIABLES ATTACHED.[*
//*[                 SORT BY PERSON INDEX TO CREATE A PERSON MONTH [*
//*[                 SORTED FILE.                                 [*
//*[*************************************************************/
//SASCBK EXEC SAS,CLASS='*',REGION=5000K,COND=(5,LE)
//WORK     DD UNIT=SYSDA,SPACE=(CYL,(100,100))
//DDIN DD DSN=CBO.HRCD.P88.SIPPASA.WAVE1-9.SASFMX,DISP=SHR
//DDOUT1 DD DSN=&&KESCB06,
//          UNIT=DISK,SPACE=(TRK,(20,20),RLSE),
//          DISP=(NEW,PASS)
//DDOUT2 DD DSN=CBO.HRCD.P88.SIPPASA.WAVE1-9.SASPMX,
//         DISP=(NEW,CATLG,DELETE),
//         UNIT=DISK,SPACE=(TRK,(20,20),RLSE)
//SYSIN DD *
 OPTIONS MPRINT SYMBOLGEN MLOGIC MACROGEN;
 OPTIONS NOCENTER;

 /*******************************************************************
 * READER: INCREMENT THE FAMILY COUNTER AND ASSIGN EACH VARIABLE    *
 *         TO THE APPROPRIATE ARRAY INDEX.                          *
 *         THE FAMILY ID IS SUID[[INDEX[[HADDID[[FNUMBER.           *
 *         WITH THIS ID, RELATED SUBFAMILIES ARE INCLUDED WITH THE  *
 *         PRIMARY FAMILY.                                          *
 *******************************************************************/
 %MACRO READER;

   FAMCOUNT = FAMCOUNT + 1;
   RECREAD = RECREAD + 1;

   AINDEX(FAMCOUNT)  = INDEX;
   ASUID(FAMCOUNT)   = SUID;
   AHADDID(FAMCOUNT) = HADDID;
   AFNUMBR(FAMCOUNT) = FNUMBR;
   ASNUMBR(FAMCOUNT) = SNUMBR;
   AINTVW(FAMCOUNT)  = INTVW;
   AMIS(FAMCOUNT)    = MIS;
   AMIS1(FAMCOUNT)   = MIS1;
   AENTRY(FAMCOUNT)  = ENTRY;
   APERNUM(FAMCOUNT) = PERNUM;
   AWGT(FAMCOUNT)    = WGT;
   ARRP(FAMCOUNT)    = RRP;
   AMS(FAMCOUNT)     = MS;
   AEARN(FAMCOUNT)   = EARN;
   AI28AMT(FAMCOUNT) = I28AMT;

   CINDEX = AINDEX(FAMCOUNT);
   FAMID=ASUID(FAMCOUNT)[[
         CINDEX[[
         AHADDID(FAMCOUNT)[[
         AFNUMBR(FAMCOUNT);

 %MEND READER;
```

```
/*****************************************************************
* PROCESS:  LOOP THROUGH ALL PEOPLE IN THE FAMILY AND ADD UP    *
*           EARNINGS AND CHILD SUPPORT.                         *
*           WRITE OUT EACH PERSON RECORD WITH THE NEW FAMILY    *
*           VARIABLES ATTACHED.                                 *
*****************************************************************/
%MACRO PROCESS;
  FAMCOUNT = FAMCOUNT-1;

  DO I = 1 TO FAMCOUNT;
    F_EARN  = F_EARN + AEARN(I);
    F_I28A  = F_I28A + AI28AMT(I);
  END;

  DO I = 1 TO FAMCOUNT;
    INDEX =  AINDEX(I) ;
    SUID  =  ASUID(I)  ;
    HADDID=  AHADDID(I);
    FNUMBR=  AFNUMBR(I);
    SNUMBR=  ASNUMBR(I);
    INTVW =  AINTVW(I) ;
    MIS   =  AMIS(I)   ;
    MISI  =  AMISI(I)  ;
    ENTRY =  AENTRY(I) ;
    PERNUM=  APERNUM(I);
    WGT   =  AWGT(I)   ;
    RRP   =  ARRP(I)   ;
    MS    =  AMS(I)    ;
    EARN  =  AEARN(I)  ;
    I28AMT=  AI28AMT(I);

    OUTPUT;
    RECWRIT = RECWRIT + 1;
  END;
%MEND PROCESS;

/*****************************************************************
* RESET: THE LAST PERSON READ IN IS THE FIRST PERSON IN THE NEXT *
*        FAMILY.  MOVE THE LAST PERSON'S INFORMATION TO THE FIRST *
*        ARRAY INDEX.  ASSIGN THE LAST FAMILY INDEX TO THE NEW   *
*        FAMILY.  RESET THE FAMILY COUNTER TO 1 AND FAMILY       *
*        VARIABLES TO 0.                                         *
*****************************************************************/
%MACRO RESET;
   I = FAMCOUNT + 1;
   AINDEX(1)  = AINDEX(I)  ;
   ASUID(1)   = ASUID(I)   ;
   AHADDID(1) = AHADDID(I) ;
   AFNUMBR(1) = AFNUMBR(I) ;
   ASNUMBR(1) = ASNUMBR(I) ;
   AINTVW(1)  = AINTVW(I)  ;
   AMIS(1)    = AMIS(I)    ;
   AMISI(1)   = AMISI(I)   ;
   AENTRY(1)  = AENTRY(I)  ;
   APERNUM(1) = APERNUM(I) ;
   AWGT(1)    = AWGT(I)    ;
   ARRP(1)    = ARRP(I)    ;
   AMS(1)     = AMS(I)     ;
   AEARN(1)   = AEARN(I)   ;
   AI28AMT(1) = AI28AMT(I) ;
   F_EARN = 0;
   F_I28A = 0;
   LASTFAM = FAMID;
   FAMCOUNT = 1;
%MEND RESET;
```

```
DATA DDOUT1.PEOPLE(
  KEEP= INDEX       SUID        NADDID      FNUMBR
        SNUMBR      INTVW       MIS         MISI
        ENTRY       PERNUM      WGT         RRP
        MS          EARN        I28AMT      F_EARN
        F_I28A         );
    ARRAY   AINDEX (22)        INDEX1 -INDEX22;
    ARRAY   ASUID  (22) $ 9   SUID1  -SUID22;
    ARRAY   ANADDID(22) $ 2   NADDID1-NADDID22;
    ARRAY   AFNUMBR(22) $ 2   FNUMBR1-FNUMBR22;
    ARRAY   ASNUMBR(22) $ 2   SNUMBR1-SNUMBR22;
    ARRAY   AINTVW (22) $ 1   INTVW1 -INTVW22;
    ARRAY   AMIS   (22) $ 1   MIS1   -MIS22;
    ARRAY   AMISI  (22) $ 1   MISI1  -MISI22;
    ARRAY   AENTRY (22) $ 2   ENTRY1 -ENTRY22;
    ARRAY   APERNUM(22) $ 3   PERNUM1-PERNUM22;
    ARRAY   AWGT   (22) $10   WGT1   -WGT22;
    ARRAY   ARRP   (22) $ 1   RRP1   -RRP22;
    ARRAY   AMS    (22) $ 1   MS1    -MS22;
    ARRAY   AEARN  (22)       EARN1  -EARN22;
    ARRAY   AI28AMT(22)       I28AMT1-I28AMT22;

    LENGTH CINDEX  $2.;
    LENGTH FAMID   $15.;
    LENGTH LASTFAM $15.;
    RECREAD = 0;
    RECWRIT = 0;
    FAMCOUNT = 0;
    SET DDIN.PEOPLE END=EOF;
    %READER;
    F_EARN = 0;
    F_I28A = 0;
    LASTFAM=FAMID;

  DO WHILE(^EOF);
    /*************************************************************
    * IF THE CURRENT PERSON IS IN A NEW FAMILY THEN PROCESS      *
    * THE FAMILY IN THE ARRAYS.                                  *
    *************************************************************/
    IF LASTFAM ^= FAMID OR FAMCOUNT = 22
      THEN DO;
        %PROCESS;
        %RESET;
      END;

    /*************************************************************
    * READ NEXT PERSON INTO ARRAYS                               *
    *************************************************************/
    SET DDIN.PEOPLE (FIRSTOBS=2) END=EOF;
    %READER;
  END;
  FAMCOUNT = FAMCOUNT + 1;
  %PROCESS;

  PUT RECREAD= RECWRIT=;
  STOP;

PROC SORT DATA=DDOUT1.PEOPLE OUT=DDOUT2.PEOPLE;
  BY SUID ENTRY PERNUM INDEX;
PROC PRINT DATA=DDOUT2.PEOPLE (OBS=100);
```

```
//KESCB06 JOB (65040CB06,BOX-85),ASASAS3,CLASS=E,
//  NOTIFY=KESCB06,MSGCLASS=5 ,TYPRUN=SCAN
//*[*************************************************************[*
//*[       PROJECT: 1989 ASA WINTER CONFERENCE EXAMPLE            [*
//*[       ANALYST: SMITH                                         [*
//*[    PROGRAMMER: SMITH                                         [*
//*[          DATE: 11/88                                         [*
//*[                                                              [*
//*[   DESCRIPTION: INPUT IS FROM ASASAS2.                        [*
//*[                PRINT OUT A 36 MONTH LIST OF PERSON EARNINGS  [*
//*[                AND FAMILY EARNINGS.                          [*
//*[**************************************************************/
//SASCBK EXEC SAS,CLASS='*',REGION=5000K,COND=(5,LE)
//WORK     DD UNIT=SYSDA,SPACE=(CYL,(100,100))
//DDIN DD DSN=CBO.NRCD.P88.SIPPASA.WAVE1-9.SASPMX,DISP=SHR
//SYSIN DD *
 OPTIONS MPRINT SYMBOLGEN MLOGIC MACROGEN NOCENTER;

 /****************************************************************
 * READER: INCREMENT THE PERSON COUNTER AND ASSIGN EACH VARIABLE  *
 *         TO THE APPROPRIATE ARRAY INDEX.                        *
 *         THE PERSON ID IS SUID[[ENTRY[[PERNUM.                  *
 ****************************************************************/
 %MACRO READER;

   PERCOUNT = PERCOUNT + 1;
   RECREAD = RECREAD + 1;

   AINDEX(PERCOUNT)  = INDEX;
   ASUID(PERCOUNT)   = SUID;
   ANADDID(PERCOUNT) = NADDID;
   AFNUMBR(PERCOUNT) = FNUMBR;
   ASNUMBR(PERCOUNT) = SNUMBR;
   AINTVW(PERCOUNT)  = INTVW;
   AMIS(PERCOUNT)    = MIS;
   AMIS1(PERCOUNT)   = MIS1;
   AENTRY(PERCOUNT)  = ENTRY;
   APERNUM(PERCOUNT) = PERNUM;
   AWGT(PERCOUNT)    = WGT;
   ARRP(PERCOUNT)    = RRP;
   AMS(PERCOUNT)     = MS;
   AEARN(PERCOUNT)   = EARN;
   AI28AMT(PERCOUNT) = I28AMT;
   AF_EARN(PERCOUNT) = F_EARN;
   AF_I28A(PERCOUNT) = F_I28A;

   PERID=ASUID(PERCOUNT)[[
         AENTRY(PERCOUNT)[[
         APERNUM(PERCOUNT);

 %MEND READER;

 /****************************************************************
 * PROCESS:  LOOP THROUGH EACH PERSON IN THE ARRAYS AND SUMMARIZE *
 *           THE DATA.                                            *
 ****************************************************************/
 %MACRO PROCESS;
   PERCOUNT = PERCOUNT-1;
   PERINDX = PERINDX + 1;

   DO I = 1 TO PERCOUNT;
     INDEX =  AINDEX(I) ;
     SUID  =  ASUID(I)  ;
     NADDID=  ANADDID(I);
     FNUMBR=  AFNUMBR(I);
```

```
      SNUMBR=  ASNUMBR(I);
      INTVW =  AINTVW(I) ;
      MIS   =  AMIS(I)   ;
      MISI  =  AMISI(I)  ;
      ENTRY =  AENTRY(I) ;
      PERNUM=  APERNUM(I);
      WGT   =  AWGT(I)   ;
      RRP   =  ARRP(I)   ;
      MS    =  AMS(I)    ;
      EARN  =  AEARN(I)  ;
      I28AMT=  AI28AMT(I);
      F_EARN=  AF_EARN(I);
      F_I28A=  AF_I28A(I);

      OUTPUT;
      RECWRIT = RECWRIT + 1;

   END;
%MEND PROCESS;

/**********************************************************************
* RESET: THE LAST PERSON READ IN IS A NEW PERSON.  MOVE THE LAST     *
*        PERSON'S INFORMATION TO THE FIRST ARRAY INDEX.  ASSIGN      *
*        THE LAST PERSON INDEX TO THE NEW PERSON.  RESET THE         *
*        PERSON COUNTER TO 1.                                        *
**********************************************************************/
%MACRO RESET;
    I = PERCOUNT + 1;

    AINDEX(1)  = AINDEX(I)  ;
    ASUID(1)   = ASUID(I)   ;
    AHADDID(1) = AHADDID(I) ;
    AFNUMBR(1) = AFNUMBR(I) ;
    ASNUMBR(1) = ASNUMBR(I) ;
    AINTVW(1)  = AINTVW(I)  ;
    AMIS(1)    = AMIS(I)    ;
    AMISI(1)   = AMISI(I)   ;
    AENTRY(1)  = AENTRY(I)  ;
    APERNUM(1) = APERNUM(I) ;
    AWGT(1)    = AWGT(I)    ;
    ARRP(1)    = ARRP(I)    ;
    AMS(1)     = AMS(I)     ;
    AEARN(1)   = AEARN(I)   ;
    AI28AMT(1) = AI28AMT(I) ;
    AF_EARN(1) = AF_EARN(I) ;
    AF_I28A(1) = AF_I28A(I) ;

    LASTPER = PERID;
    PERCOUNT = 1;

%MEND RESET;


DATA PEOPLE(
  KEEP= INDEX       SUID        HADDID      FNUMBR
        SNUMBR      INTVW       MIS         MISI
        ENTRY       PERNUM      WGT         RRP
        MS          EARN        I28AMT      F_EARN
        F_I28A      PERINDX);
    ARRAY   AINDEX (32)       INDEX1 -INDEX32;
    ARRAY   ASUID  (32) $ 9  SUID1  -SUID32;
    ARRAY   AHADDID(32) $ 2  HADDID1-HADDID32;
    ARRAY   AFNUMBR(32) $ 2  FNUMBR1-FNUMBR32;
    ARRAY   ASNUMBR(32) $ 2  SNUMBR1-SNUMBR32;
    ARRAY   AINTVW (32) $ 1  INTVW1 -INTVW32;
    ARRAY   AMIS   (32) $ 1  MIS1   -MIS32;
    ARRAY   AMISI  (32) $ 1  MISI1  -MISI32;
    ARRAY   AENTRY (32) $ 2  ENTRY1 -ENTRY32;
    ARRAY   APERNUM(32) $ 3  PERNUM1-PERNUM32;
    ARRAY   AWGT   (32) $10  WGT1   -WGT32;
    ARRAY   ARRP   (32) $ 1  RRP1   -RRP32;
    ARRAY   AMS    (32) $ 1  MS1    -MS32;
    ARRAY   AEARN  (32)      EARN1  -EARN32;
    ARRAY   AI28AMT(32)      I28AMT1-I28AMT32;
    ARRAY   AF_EARN(32)      F_EARN1-F_EARN32;
    ARRAY   AF_I28A(32)      F_I28A1-F_I28A32;

    LENGTH PERID   $15.;
    LENGTH LASTPER $15.;
    PERINDX = 0;
    RECREAD = 0;
    RECWRIT = 0;
    PERCOUNT = 0;
    SET DDIN.PEOPLE END=EOF;
    %READER;
    LASTPER=PERID;

  DO WHILE(^EOF);
    /*************************************************************
    * IF THE CURRENT PERSON IS IN A NEW FAMILY THEN PROCESS       *
    * THE FAMILY IN THE ARRAYS.                                   *
    *************************************************************/
    IF LASTPER ^= PERID OR PERCOUNT = 32
      THEN DO;
        %PROCESS;
        %RESET;
      END;

    /*************************************************************
    * READ NEXT PERSON INTO ARRAYS                                *
    *************************************************************/
    SET DDIN.PEOPLE (FIRSTOBS=2) END=EOF;
    %READER;

  END;
  PERCOUNT = PERCOUNT + 1;
  %PROCESS;

  PUT RECREAD= RECWRIT=;
  STOP;


PROC PRINT DATA=PEOPLE (OBS=100);
  BY PERINDX;
```