FINAL REPORT

A COMPUTER BASED INFORMATION SYSTEM
FOR COUNTY EQUIPMENT COST RECORDS

HR-173

Submitted to the
Iowa Department of Transportation
Highway Division
and the
Iowa Highway Research Board
by

John D. Poyzer
and
John M. Liittschwager

Systems Division, College of Engineering
The University of Iowa
Iowa City, Iowa 52242

July 31, 1975
This appendix is divided into three sections. The first section contains abstracts of each of the eight computer programs in the system, instructions for keypunching the three input documents, and computer operating instructions pertaining to each program. The second section contains system flowcharts for the entire system as well as program flowcharts for each program. The last section contains PL/1 program listings of each program.

Program abstracts and computer operating instructions are given in this section for the eight computer programs which make up the information system for county equipment cost records. They are presented in the order in which the programs are to be executed: DIRECT, INDCOST, STCHNGE, UPDATEM, MAIN, CTYSUMRY, MFGAGE, and UPDATE. Keypunch instructions are also included for input data associated with programs DIRECT, INDCOST and STCHNGE, which are the only programs processing county supplied data.
Abstract for Program DIRECT

1. Purpose: To sort and edit the Direct Cost Summary Cards.

2. Procedure: Program DIRECT is a two-step job. The first step is a utility sort routine which sorts the input cards by county number and equipment number. The second step performs reasonableness checks of the input cards on the county field, the month field, the year field, and the remaining data fields taken collectively. Records without errors are written to a disk output file. Input records containing errors are grouped by county and recorded on printed output.

3. Configuration: IBM 370/145
   One Card Reader
   One Printer
   Six to Eight Cylinders of IBM 3330 Disk Space

4. Source Language: PL/1

5. Limitations: None

6. Running Time: Approximately five minutes CPU time

7. Additional Remarks: After running DIRECT, a check should be made of the error report to eliminate errors associated with keypunching and local data processing. If such errors are found, corrections should be made and DIRECT should be rerun prior to running MAIN. Error listings associated with county input are return-
ed to the counties with the other output from the system.

8. Subroutines: None
Keypunch Instructions for Program DIRECT

Input Document: Direct Cost Summary Form

<table>
<thead>
<tr>
<th>Card Columns</th>
<th>Field Name</th>
<th>Numeric</th>
<th>Alpha-numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>County Number</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4-11</td>
<td>Equipment Number</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>12-13</td>
<td>Month</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>14-15</td>
<td>Year</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>16-21</td>
<td>Fuel Cost</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>22-26</td>
<td>Lubricant Cost</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>27-30</td>
<td>Antifreeze Cost</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>31-36</td>
<td>Tires and Tubes Cost</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>37-42</td>
<td>Expendable Parts Cost</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>43-48</td>
<td>Repair Parts Cost</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>49-54</td>
<td>Labor Cost</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>55-59</td>
<td>Mileage or Hours</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>60-63</td>
<td>Down Time in Hours</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>64-65</td>
<td>Number of Times Repaired</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Special Instructions: Use left zero capability on all numeric fields when setting up drum card. A sample Direct Cost Summary Form appears on the next page.
# DIRECT COST SUMMARY FORM

<table>
<thead>
<tr>
<th>COUNTY NAME</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COUNTY EQUIPMENT NUMBER</th>
<th>DATE</th>
<th>FUEL</th>
<th>LUBR.</th>
<th>ANTI- FREEZE TUBES</th>
<th>EXPENDABLE PARTS</th>
<th>REPAIR PARTS</th>
<th>LABOR</th>
<th>MILES/HOURS</th>
<th>DOWN TIME</th>
<th>TIMES REPAIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12/14</td>
<td>16</td>
<td>22</td>
<td>27</td>
<td>31</td>
<td>37</td>
<td>43</td>
<td>49</td>
<td>55</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DCSF - No. 5
7-75
Computer Operating Instructions for Program DIRECT

Frequency of Run: Annually
Region: 100K
Source Language: PL/1
Run Time: Approximately five minutes CPU time
Job Description: Sorts and edits Direct Cost Summary Cards,
prints list of invalid input cards, writes
correct input data to disk file.
Printer Used: Yes    Punch Used: No
Printer Forms: Wide stock forms with standard alignment and
standard carriage control tape.
Sort Sequence: Card Positions 1-11, CH, A
Data Sets:

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>DSNAME</th>
<th>T/D/C/P</th>
<th>I/O</th>
<th>DISP</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORTIN</td>
<td></td>
<td>C</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORTOUT</td>
<td>&amp;&amp;DCOST</td>
<td>D</td>
<td>O</td>
<td>NEW, PASS</td>
<td>80</td>
</tr>
<tr>
<td>Step 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCOST</td>
<td>DCOST1</td>
<td>D</td>
<td>O</td>
<td>NEW, KEEP</td>
<td>65</td>
</tr>
<tr>
<td>SDCOST</td>
<td>*.STEP1.SORT.SORTOUT</td>
<td>D</td>
<td>I</td>
<td>OLD, DELETE</td>
<td>80</td>
</tr>
<tr>
<td>SYSIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSPRINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abstract for Program INDCOST

1. Purpose: To sort and edit the Indirect Cost Cards.

2. Procedure: Program INDCOST is a two-step job. The first step is a utility sort routine which sorts the input cards by county number. The second step performs reasonableness checks of the input cards on the county field and the remaining data fields taken collectively. Records without errors are written to a disk output file. Input records containing errors are listed on printed output.

3. Configuration: IBM 370/145
   - One Card Reader
   - One Printer
   - One to Two Tracks of IBM 3330 Disk Space

4. Source Language: PL/I

5. Limitations: None

6. Running Time: Approximately one minute CPU time

7. Additional Remarks: After running INDCOST, a check should be made of the error report to eliminate errors associated with keypunching and local data processing. If such errors are found, corrections should be made and INDCOST should be rerun prior to running MAIN. Error listings associated with county input are returned to the counties with the other output from the system.

8. Subroutines: None
Keypunch Instructions for Program INDCOST

Input Document: Indirect Cost Form

<table>
<thead>
<tr>
<th>Card Columns</th>
<th>Field Name</th>
<th>Numeric</th>
<th>Alpha-numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>County Number</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4-10</td>
<td>Salaries-Supervisory Employees</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>11-17</td>
<td>Salaries-Clerical Employees</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>18-24</td>
<td>Utilities</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>25-31</td>
<td>Building Depreciation, Maintenance, Rental</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>32-37</td>
<td>Shop Equipment Depreciation</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>38-43</td>
<td>Replacement of Expendable Shop Tools</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>44-49</td>
<td>Offices Supplies</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>50-56</td>
<td>Cost of Moving Equipment</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>57-63</td>
<td>Equipment Insurance</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Special Instructions: Use left zero capability on all numeric fields when setting up drum card. A sample Indirect Cost Form appears on the next page.
INDIRECT COST FORM

COUNTY NAME _______________________

DATE _______________________________

1. COUNTY NUMBER

4. SALARIES AND EXPENSES OF SUPERVISORY EMPLOYEES NOT DIRECTLY EMPLOYED IN SERVICING OR REPAIRING EQUIPMENT

11. SALARIES OF CLERICAL EMPLOYEES ENGAGED IN ACCOUNTING AND PREPARING REPORTS FOR EQUIPMENT.

SHOP STORAGE AND MISCELLANEOUS OVERHEAD COSTS RELATING TO EQUIPMENT CARE AND HANDLING:

18. a) UTILITIES

25. b) SHOP AND OFFICE BUILDING DEPRECIATION, MAINTENANCE AND/OR RENTAL

32. c) DEPRECIATION OF SHOP EQUIPMENT

38. d) REPLACEMENT COST OF EXPENDABLE SHOP TOOLS

44. e) OFFICE SUPPLIES

50. f) COST OF MOVING EQUIPMENT

57. g) EQUIPMENT INSURANCE

ICF - No. 6
7-75
Computer Operating Instructions for Program INDCOST

Frequency of Run: Annually
Region: 100K
Source Language: PL/1
Run Time: Approximately one minute CPU time
Job Description: Sorts and edits Indirect Cost Cards, prints list of invalid input cards, writes correct data to disk file.
Printer Used: Yes Punch Used: No
Printer Forms: Wide stock forms with standard alignment and standard carriage control tape.
Sort Sequence: Card Positions 1-3, CH, A

Data Sets:

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>DSNAME</th>
<th>T/D/C/P</th>
<th>I/O</th>
<th>DISP</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>SORTIN</td>
<td>*</td>
<td>C</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SORTOUT  &amp;ICOST</td>
<td>D</td>
<td>O</td>
<td>NEW,PASS</td>
<td>80</td>
</tr>
<tr>
<td>Step 2</td>
<td>ICOST</td>
<td>ICOST1</td>
<td>D</td>
<td>0</td>
<td>NEW,KEEP</td>
</tr>
<tr>
<td></td>
<td>SICOST   *,.STEP1.SORT.SORTOUT</td>
<td>D</td>
<td>I</td>
<td>OLD,DELETE</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>SYSIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSPRINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abstract for Program STCHNGE

1. Purpose: To sort and edit the Equipment Status Change Cards.

2. Procedure: Program STCHNGE is a two-step job. The first step is a utility sort routine which sorts the input cards by county number, equipment number, and card type. The second step first reads in from cards the 55 equipment class codes and descriptions. It then performs reasonableness checks on all three cards of each equipment status change record. Input records containing errors are grouped by county and listed on printed output. Input records without errors have the appropriate district number and equipment class description inserted in the record with the record then written to a disk output file.

3. Configuration: IBM 370/145
   One Card Reader
   One Printer
   Three to Four Cylinders of IBM 3330 Disk Space

4. Source Language: PL/1

5. Limitations: None

6. Running Time: Approximately two minutes CPU time

7. Additional Remarks: After running STCHNGE, a check should be made of the error report to eliminate errors
associated with keypunching and local data processing. If such errors are found, corrections should be made and STCHNGE should be rerun prior to running UPDATEM. Error listings associated with county input are returned to the counties with other output from the system.

8. Subroutines: None
**Keypunch Instructions for Program STCHNGE**

**Card 1**

Input Document: Equipment Status Change Form

<table>
<thead>
<tr>
<th>Card Columns</th>
<th>Field Name</th>
<th>Numeric</th>
<th>Alpha-numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>County Number</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4-11</td>
<td>Equipment Number</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>12</td>
<td>Card Type</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>13-14</td>
<td>District</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>15-16</td>
<td>Class Code</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>17-36</td>
<td>Class Description</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>37-39</td>
<td>Manufacturer Code Number</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>40-41</td>
<td>Year Equipment Manufactured</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>42-55</td>
<td>Make and Model Description</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>56-69</td>
<td>Manufacturer's Serial Number</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>70-72</td>
<td>Wheelbase</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>73-78</td>
<td>Date Purchased</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
**Keypunch Instructions for Program STCHNGE**

**Card 2**

*Input Document: Equipment Status Change Form*

<table>
<thead>
<tr>
<th>Card Columns</th>
<th>Field Name</th>
<th>Numeric</th>
<th>Alpha-numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>County Number</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4-11</td>
<td>Equipment Number</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Card Type</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Type of Engine</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>14-27</td>
<td>Engine Make &amp; Model Description</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>28-29</td>
<td>Engine Manufacturer Code</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>30-32</td>
<td>Rated Horsepower</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>33-34</td>
<td>Number of Cylinders in Engine</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Transmission Type</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>36-47</td>
<td>Dealer Purchased From</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>48-55</td>
<td>Original Purchase Cost</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>56-62</td>
<td>Salvage Value</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>63-70</td>
<td>Book Value</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>Sold, Junked or Traded Code</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>72-77</td>
<td>Date Sold, Junked or Traded</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
**Keypunch Instructions for Program STCHNGE**

**Card 3**

**Input Document: Equipment Status Change Form**

<table>
<thead>
<tr>
<th>Card Columns</th>
<th>Field Name</th>
<th>Numeric</th>
<th>Alphanumeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>County Number</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4-11</td>
<td>Equipment Number</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>12</td>
<td>Card Type</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>13-18</td>
<td>Miles or Hours-Life</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>19-25</td>
<td>Fuel Cost-Life</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>26-31</td>
<td>Lubricant Cost-Life</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>32-37</td>
<td>Tires and Tubes Cost-Life</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>38-43</td>
<td>Expendable Parts Cost-Life</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>44-48</td>
<td>Antifreeze Cost-Life</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>49-55</td>
<td>Parts Cost-Life</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>56-62</td>
<td>Labor Cost-Life</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>63-69</td>
<td>Indirect Cost-Life</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Special Instructions:** All three cards must be keypunched. Use left zero capability on all numeric fields when setting up drum card. Blank numeric fields may be skipped, rather than filled with all zeros. A sample Equipment Status Change Form appears on the next page.
## EQUIPMENT STATUS CHANGE FORM

<table>
<thead>
<tr>
<th>Card Type 1</th>
<th>Card Type 2</th>
<th>Card Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTY NUMBER</td>
<td>COUNTY NAME</td>
<td>COUNTY NUMBER</td>
</tr>
<tr>
<td>EQUIPMENT NUMBER</td>
<td>DATE</td>
<td>EQUIPMENT NUMBER</td>
</tr>
</tbody>
</table>

### Card Type 1
- CARD TYPE 1
- CARD NUMBER
- DISTRICT
- CLASS CODE
- CLASS DESCRIPTION
- MANUFACTURER CODE NUMBER
- YEAR EQUIPMENT MANUFACTURED
- MAKE AND MODEL DESCRIPTION
- MANUFACTURER'S SERIAL NUMBER
- WHEELBASE IN INCHES
- DATE PURCHASED-MMDDYY

### Card Type 2
- CARD TYPE 2
- TYPE OF ENGINE - GAS (G) OR DIESEL (D)
- ENGINE MAKE AND MODEL DESCRIPTION
- ENGINE MANUFACTURER CODE
- RATED HORSEPOWER
- NUMBER OF CYLINDERS IN ENGINE
- TRANSMISSION TYPE - AUTOMATIC (A) OR STANDARD (S)
- DEALER PURCHASED FROM
- ORIGINAL PURCHASE COST
- SALVAGE VALUE
- BOOK VALUE
- SOLD, JUNKED, OR TRADED CODE - (S) (J) (T)
- DATE SOLD, JUNKED OR TRADED - MMDDYY

### Card Type 3
- CARD TYPE 3
- MILES OR HOURS - LIFE
- FUEL COST
- LUBRICANT COST - LIFE
- TIRES AND TUBES COST - LIFE
- EXPENDABLE PARTS COST - LIFE
- ANTIFREEZE COST - LIFE
- REPAIR PARTS COST - LIFE
- LABOR COST - LIFE
- INDIRECT COST - LIFE
Computer Operating Instructions for Program STCHNGE

Frequency of Run: Annually
Region: 100K
Source Language: PL/1
Run Time: Approximately two minutes CPU time

Job Description: Sorts and edits Equipment Status Change Cards, prints list of invalid input cards, combines the three input cards per piece of equipment into one logical record, and writes correct input data to disk file. Also reads in list of class codes and descriptions from cards and places proper description in each equipment status change record.

Printer Used: Yes     Punch Used: No
Printer Forms: Wide stock forms with standard alignment and standard carriage control tape.

Sort Sequence: Card Positions 1-12, CH, A

Data Sets:

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>DSNAME</th>
<th>T/D/C/P</th>
<th>I/O</th>
<th>DISP</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORTIN</td>
<td>*</td>
<td>C</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORTOUT</td>
<td>&amp;change</td>
<td>D</td>
<td>O</td>
<td>NEW,PASS 80</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSC</td>
<td>*.STEP1.SORT.SORTOUT</td>
<td>D</td>
<td>I</td>
<td>OLD,DELETE 80</td>
<td></td>
</tr>
<tr>
<td>SCHANGE</td>
<td>SCHANGEL</td>
<td>D</td>
<td>O</td>
<td>NEW,KEEP 199</td>
<td></td>
</tr>
<tr>
<td>SYSIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSPRINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abstract for Program UPDATTEM

1. Purpose: To add new equipment records to the equipment master file, to correct master file records which have been found to contain incorrect data, to flag records of equipment disposed of, to calculate the total book value of all equipment for each county, and to print out an inventory listing of all equipment in each county.

2. Procedure: Records are read from the equipment master file (a tape file saved from the previous year's processing) and from the equipment status change record file created in program STCHNGE. Equipment status change records which do not match any master file record are added to the master file while matching records are corrected or flagged for later deletion.

While the program is adding records to the master file and updating others, it also calculates the total book value of all equipment for each county. A disk output file is constructed containing records for each county. These records contain the county number and the total book value of all county equipment recorded in the master file.

Printed output is also provided giving an inventory listing by county of each piece of equipment held in the master file.

3. Configuration: IBM 370/145

One Printer
One Tape Drive
Approximately Eight Cylinders of IBM
3330 Disk Space

4. Source Language: PL/1

5. Limitations: Cannot be run until after STCHNGE has been run. The first time the system is run, a dummy equip-
ment master file with only an EOF marker must have been created.

6. Running Time: Approximately five minutes CPU time.

7. Additional Remarks: Two copies of the printed output are to be produced for county use.

8. Subroutines: No external procedures.
Computer Operating Instructions for Program UPDATEM

Frequency of Run: Annually

Region: 100K

Source Language: PL/1

Run Time: Approximately five minutes CPU time

Job Description: Reads master file and equipment status change record file. Attempts to match records from these files by county number and equipment number. Matching records cause master file to be updated. Non-matching equipment status change records are added to master file. Total equipment book value is calculated by county and written to disk output file. A printed inventory listing of all equipment by county is produced.

Printer Used: Yes  Punch Used: No

Printer Forms: Wide stock forms with standard alignment and standard carriage control tape. Make two copies of printed output.

Data Sets:

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>DSNAME</th>
<th>T/D/C/F</th>
<th>I/O</th>
<th>DISP</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQMAST</td>
<td>EQMAST1</td>
<td>T</td>
<td>I</td>
<td>OLD,KEEP</td>
<td>300</td>
</tr>
<tr>
<td>NEQMAST</td>
<td>EQMAST2</td>
<td>D</td>
<td>O</td>
<td>NEW,KEEP</td>
<td>300</td>
</tr>
<tr>
<td>STCHNG</td>
<td>SCHANGE1</td>
<td>D</td>
<td>I</td>
<td>OLD,DELETE</td>
<td>199</td>
</tr>
<tr>
<td>BKVALU</td>
<td>BKVALU1</td>
<td>D</td>
<td>O</td>
<td>NEW,KEEP</td>
<td>13</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abstract for Program MAIN

1. Purpose: To update all master file records with the past year's direct and indirect cost data, to calculate cost data, to calculate the past year's and lifetime costs per mile/hour for each piece of equipment, and to produce a detailed printed output for all equipment by county and equipment classes showing direct costs, indirect costs, and cost per mile/hour figures for the past year and for the equipment's lifetime.

2. Procedure: MAIN is a three step job. The first step updates the master file, the second step is a utility sort to sort the master file into the appropriate sequence for the printed output, which is then produced in the third step.

The first step reads the records from the indirect cost record file and the total book value file created by INDCOST and UPDATEM, respectively, and places the contents of the records read into arrays. Next, records are read from the master file created in UPDATEM and the direct cost summary records file created in DIRECT. The program logic attempts to match records from these two files by county number and equipment number. All master file records which are matched by a corresponding direct cost summary record are updated. The past year's direct costs from the direct cost summary record are entered in-
to the master file record and the lifetime direct costs are updated. A portion of the county's total indirect equipment costs are allocated to each piece of equipment based upon the ratio of each piece's book value to the county's total equipment book value. The current book value of each piece of equipment is then updated through an appropriate depreciation calculation.

If a match does not occur between the master file record and the direct cost summary record, a printed listing of the nonmatching record is produced. Most of the listings in this output will stem from using an incorrect equipment number or county number on the direct cost summary form, or not submitting a direct cost summary form for a piece of equipment currently on the master file.

The second step of MAIN is a utility sort routine. The existing master file is sorted by county number and equipment number within the county number. The output desired from MAIN is classified by county, equipment class, and the equipment number. Therefore, the sort is necessary to get all pieces of equipment for each county into their proper equipment class so class totals can be calculated and printed along with the individual listings for each piece of equipment in each equipment class.
The third step of MAIN calculates total direct cost for each record for the past year and for the equipment's lifetime. Cost per mile/hour figures are then calculated.

Equipment class and county totals are accumulated, both for the past year's costs as well as for lifetime costs. Output is produced by county and equipment class showing detailed cost breakdowns for each piece of equipment for the past year's costs as well as for lifetime costs. Equipment class and county totals are also shown.

3. Configuration: IBM 370/145

   One Printer
   Six to Eight Cylinders of IBM 3330 Disk Space

4. Source Language: PL/1

5. Limitations: Cannot be run until DIRECT, INDCOST, and UPDATEM have been run.

6. Running Time: Approximately 30 minutes CPU time.

7. Additional Remarks: Two copies of the printed output are to be produced for county use.

8. Subroutines: No external procedures.
Computer Operating Instructions for Program MAIN

Frequency of Run: Annually
Region: 100K
Source Language: PL/1
Run Time: Approximately 30 minutes CPU time
Job Description: Reads equipment master file, direct cost summary file, indirect cost record file, total book value file, and updates master file records from matching direct cost summary records producing updated equipment master file. Sorts updated equipment master file, produces detailed printed cost figures on all equipment records in master file by county and equipment class.

Printer Used: Yes            Punch Used: No
Printed Forms: Wide stock forms with standard alignment and standard carriage control tape. Make two copies of printed output.
Sort Sequence: Record Positions 1-3, CH, A; 14-15, CH, A; 6-13, CH, A
### Data Sets:

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>DSNAME</th>
<th>T/D/C/P</th>
<th>I/O</th>
<th>DISP</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>SEQMAST</td>
<td>EQMAST2</td>
<td>D</td>
<td>I OLD,DELETE</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>TEQMAST</td>
<td>EQMAST3</td>
<td>D</td>
<td>O NEW,PASS</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>BKVALU</td>
<td>BKVALU1</td>
<td>D</td>
<td>I OLD,DELETE</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>INDCOST</td>
<td>ICOST1</td>
<td>D</td>
<td>I OLD,DELETE</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>DIRCOST</td>
<td>DCOST1</td>
<td>D</td>
<td>I OLD,DELETE</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>SYSPRINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>SORTIN</td>
<td>*STEP1.GO.TEQMAST</td>
<td>D</td>
<td>I OLD,KEEP</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>SORTOUT</td>
<td>&amp;SMAST</td>
<td>D</td>
<td>O NEW,PASS</td>
<td>300</td>
</tr>
<tr>
<td>Step 3</td>
<td>FEQMAST</td>
<td>*.STEP2.SORT.SORTOUT</td>
<td>D</td>
<td>I OLD,DELETE</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>YRDIR</td>
<td></td>
<td>P</td>
<td>O</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>YRTOT</td>
<td></td>
<td>P</td>
<td>O</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>LIFEDIR</td>
<td></td>
<td>P</td>
<td>O</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>LIFETOT</td>
<td></td>
<td>P</td>
<td>O</td>
<td>132</td>
</tr>
</tbody>
</table>
Abstract for Program CTYSUMRY

1. Purpose: To produce past year and lifetime cost per mile/hour averages for each equipment class by districts and counties.

2. Procedure: Program CTYSUMRY is a two-step job. The first step sorts the master file produced in MAIN by equipment class, district, and county number. The second step reads records from the sorted file produced in step one. It keeps running totals of total direct costs, pieces of equipment, and miles/hours. When the equipment class, district number, or county number of the record just read differs from the previous record, the appropriate average cost per mile/hour calculations are made, the averages are printed, and appropriate running totals are reset to zero.

3. Configuration: IBM 370/145
   One Printer
   Six to Eight Cylinders of IBM 3330 Disk Space

4. Source Language: PL/1

5. Limitations: Cannot be run until MAIN has produced an appropriate master file. One hundred copies must be produced if all counties are to receive output.

6. Running Time: Approximately 10 minutes CPU time.
7. Additional Remarks: Make 100 copies of printed output.
8. Subroutines: None
Computer Operating Instructions for Program CTYSUMRY

Frequency of Run: Annually
Region: 100K
Source Language: PL/1
Run Time: Approximately 10 minutes CPU time.
Job Description: Sorts master file, produces past year and lifetime cost per mile/hour averages for each equipment class by districts and counties.
Printer Used: Yes
Punch Used: No
Printer Forms: Wide stock forms with standard alignment and standard carriage control tape. Make 100 copies.
Sort Sequence: Record positions 14-15, CH, A; 4-5, CH, A; 1-3, CH, A

Data Sets:

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>DSNAME</th>
<th>T/D/C/P</th>
<th>I/O</th>
<th>DISP</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORTIN</td>
<td>EQMAST3</td>
<td>D</td>
<td>I</td>
<td>OLD,KEEP</td>
<td>300</td>
</tr>
<tr>
<td>SORTOUT</td>
<td>&amp;&amp;COUNTY</td>
<td>D</td>
<td>O</td>
<td>NEW,PASS</td>
<td>300</td>
</tr>
</tbody>
</table>

| Step 2  |                |         |       |             |       |
| UEMP    | *.STEP1.SORT.SORTOUT | D       | I     | OLD,DELETE  | 300   |
| OUT     |                | P       | O     |             | 132   |
Abstract for Program MFGAGE

1. Purpose: To produce past year and lifetime cost per mile/hour averages for five age groups in each equipment class by manufacturer.

2. Procedure: Program MFGAGE is a two-step job. The first step sorts the master file produced in MAIN by equipment class and manufacturer. The second step reads records from the sorted file produced in step one. This step keeps running totals of total direct costs, pieces of equipment, and miles/hours. After each record is read, the age of the piece of equipment contained on that record is calculated, and the direct costs and miles/ hours on the record are added to the appropriate age grouping. Cost per mile/hour calculations are made and output produced whenever the current and past records have differing equipment class codes or manufacturer codes.

3. Configuration: IBM 370/145
   One Printer
   Six to Eight Cylinders of IBM 3330 Disk Space

4. Source Language: PL/1

5. Limitations: Cannot be run until MAIN has produced an appropriate master file. One hundred copies must be produced if all counties are to receive output.
6. Running Time: Approximately 12 minutes CPU time.

7. Additional Remarks: Make 100 copies of printed output.

8. Subroutines: None
Computer Operating Instructions for Program MFGAGE

Frequency of Run: Annually
Region: 100K
Source Language: PL/1
Run Time: Approximately 12 minutes CPU time
Job Description: Sorts master file, produces past year and lifetime averages cost per mile/hour for five groups in each equipment class by manufacturer.

Printer Used: Yes        Punch Used: No
Printer Forms: Wide stock forms with standard alignment and standard carriage control tape. Make 100 copies.

Sort Sequence: Record Positions 14-15, CH, A; 18-20, CH, A

Data Sets:

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>DSNAME</th>
<th>T/D/C/P</th>
<th>I/O</th>
<th>DISP</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>SORTIN</strong></td>
<td><strong>EQMAST3</strong></td>
<td>D</td>
<td>I</td>
<td>OLD,KEEP</td>
</tr>
<tr>
<td><strong>SORTOUT</strong></td>
<td>&amp;MFGAGE</td>
<td><strong>D</strong></td>
<td>O</td>
<td>NEW,PASS</td>
<td>300</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>UEMF</strong></td>
<td>*.STEP1.SORT.SORTOUT</td>
<td>D</td>
<td>I</td>
<td>OLD,DELETE</td>
</tr>
<tr>
<td><strong>OUT</strong></td>
<td><strong>P</strong></td>
<td>O</td>
<td></td>
<td></td>
<td>132</td>
</tr>
</tbody>
</table>
Abstract for Program UPDATE

1. Purpose: To delete records of equipment disposed of in the past year from the master file and to produce a printed listing by county of the deleted records.

2. Procedure: Program UPDATE reads the master file created in MAIN. All records to be deleted were flagged for deletion by program UPDATEM by setting a flag in the disposal method field. All records with the disposal method field set to S, J, or T are removed from the master file and listed by county on the printer. The master file produced is written on magnetic tape and will be used as input in the year to follow.

3. Configuration: IBM 370/145

   One Printer
   One Tape Drive
   Three to Four Cylinders of IBM 3330
   Disk Space

4. Source Language: PL/1

5. Limitations: Cannot be run until MAIN, CTYSUMRY, and MFGAGE have been run.

6. Running Time: Approximately two minutes CPU time.

7. Additional Remarks: The new master tape file produced must be labeled with the current year to avoid confusing it with the previous year's master file. Make a back up magnetic tape copy of the new master file.

8. Subroutines: None
Computer Operating Instructions for Program UPDATE

Frequency of Run: Annually
Region: 100K
Source Language: PL/1
Run Time: Approximately two minutes CPU time
Job Description: Deletes records of equipment disposed of in the past year from the master file, produces printed output of deleted records, and writes new equipment master file to magnetic tape.

Printer Used: Yes  Punch Used: No
Printer Forms: Wide stock forms with standard alignment and standard carriage control tape.

Data Sets:

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>DSNAME</th>
<th>T/D/C/P</th>
<th>I/O</th>
<th>DISP</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLDMST</td>
<td>EQMAST3</td>
<td>D</td>
<td>OLD,DELETE</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>NEWMST</td>
<td>EQMAST1</td>
<td>T</td>
<td>NEW,KEEP</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>SYSPRINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This section of Appendix D contains the system flowcharts and the eight computer program flowcharts. The system flowcharts give the general system information flow. The computer program flowcharts show the specific flow of data processing in each computer program. These program flowcharts correspond to the PL/1 listings given in the next section.
Counties

Operator Forms

Shop Repair & Maintenance Forms

Statements from Commercial Repair Shops

Sort and Collate by Equipment No.

Done once per month. Put all forms for each piece of equipment together. Sort into numeric or alphabetic order by the equipment no.

Fuel Used

Transcribe Totals to Summary Forms
done once each month for each piece of equipment. Total all fuel used, multiply by rate to get dollars. Do likewise for all similar items. New forms started yearly. Good for 12 months.

Monthly Summary Forms

Summary Forms File, Update monthly, total at year end

Used to add new equip, delete old equip, or correct existing equip. records.

Complete Status Change Forms

Enter Indirect Costs on Ind. Cost Form, 1 per eqv.

Transcribe Totals to Direct Cost Summary Forms

One set of totals for each piece of equipment

Equipment Status Change Forms

Indirect Cost Form

Direct Cost Summary Forms

Each county sends completed forms to Highway Division, DOT, once per year, in January

Collect Forms from All Counties

Separate by type of form, and visually edit before being sent to DOT Data Processing Department

H1

H4

DOT

Indirect Cost Forms for All Counties

Direct Cost Summary Forms for All Counties

H5

DOT

Equipment Status Change Forms for All Counties

Indirect Cost Forms for All Counties

Direct Cost Summary Forms for All Counties

H2

H3

DOT

Data Processing Department

K1

K2

K3

K4

K5

K6

to SF2

to SF2

to SF2

to SF2
Start

Sort Input Cards by County

Explicitly Define Variables and Files Used in Program

Initialize Variables
LAST_CNTY_NO=0
ERROR=1

Read a Record

End of File

Yes

DONE?

ERROR=0?

No

Print "NO ERRORS" for Last County

End

LAST_CNTY_NO=Cnty No. of Record

Yes

No

Print Heading on New Page for County No. just read;
Set LAST_CNTY_NO to County Number of Record just read;
Set ERROR=0

ERROR=0?

Yes

No

Print "ERROR=0?" for previous County

G1

G2

G3

G4

G5

H1

H2

H3

H4

H5

J1

J2

J3

J4

J5

K1

K2

K3

K4

K5
Programmer: Rolf McHenry

Flowchart:

1. Set COUNTY_CODE = 1
2. County NUMBER < '001' or > '099'?
   - Yes: Set COUNTY_CODE = 0
   - No: All Remaining Columns = 0012345
3. Calculate Cord Column at First Illegal Character
4. Write to Disk File Record Just Read and Edited
5. COUNTY_CODE = 0?
   - No: Print out "RECORD HAS ILLEGAL COUNTY CODE"
   - Yes: Print out "ILLEGAL CHARACTER IN COLUMN" & Col. No.
6. Print out a Copy of Record Just Read In
7. Set ERROR = 1
8. to 2.1
9. to 2.1
from 3.3

Card Type on Record Just Read = 2?

Type of Eng. = G, D, or B?

Eng. Mfr. Code < COB = 2 or > 34?

Horsepower < COB = 1655 or > 999?

No. of Cylinders = 6b, 6c, 8, or 12?

Transmission Type = A, B, or S?

Chase Cost, Salvage Value & Book Value = All Digits or Blanks?

Status Change Code = T, 2, 3, 5, or 7?

Print "CARD TYPE NOT 2 ON SECOND CARD"

Set FLAG = 1

Print "ENGINE FUEL CODE NOT 6 OR 9"

Set FLAG = 1

Print "ENGINE MANUFACTURER CODE IS INVALID"

Set FLAG = 1

Print "HORSEPOWER VALUE IS INVALID"

Set FLAG = 1

Print "THE NUMBER OF CYLINDERS IS INVALID"

Set FLAG = 1

Print "THE TRANSMISSION CODE IS INVALID"

Set FLAG = 1

Print "INVALID CHARACTERS IN COST OR VALUE FIELDS"

Set FLAG = 1

Print "INVALID CODE IN JUNKED SOLD, OR TRADED CODE"

Print "CARD TYPE NOT 2 ON SECOND CARD"

Set FLAG = 1

Print "ENGINE FUEL CODE NOT 6 OR 9"

Set FLAG = 1

Print "ENGINE MANUFACTURER CODE IS INVALID"

Set FLAG = 1

Print "HORSEPOWER VALUE IS INVALID"

Set FLAG = 1

Print "THE NUMBER OF CYLINDERS IS INVALID"

Set FLAG = 1

Print "THE TRANSMISSION CODE IS INVALID"

Set FLAG = 1

Print "INVALID CHARACTERS IN COST OR VALUE FIELDS"

Set FLAG = 1

Print "INVALID CODE IN JUNKED SOLD, OR TRADED CODE"
Define Input and Output Files

Explicitly Declare Variables Used in the Program

Calculate Current Year Using Built-In Function

Initialize Variables & Arrays. Set
SCR.COUNTY_NUM='999';
MSTR.COUNTY_NUM='998';
LAST.CNTY_NUM='000';
TOTAL_BOOK_VALU=0;
SCDONE='0';
EMDONE='0';
Set CNTY_NAMES Array to the Names of the Counties

Read a Record from Equip. Master File (MSTR)

Set EMDONE='1'; MSTR.COUNTY_NUM='999';
MSTR.EQ_NUM='99999999'

Read a Record from the Status Change Records File (SCR)

Write a Record to the Book Value File

DONE:

END:

P

to 4.2

End
from 4.4

SCR. SALVAGE VALUE > 0
Yes
Set MSTR. SALVAGE VALUE = SCR. SALVAGE VALUE
No

SCR. BOOK VALUE > 0
Yes
Set MSTR. BOOK VALUE = SCR. BOOK VALUE
No

Set MSTR. DISPOSITION = SCR. DISPOSITION;
MSTR.DATE_DISPOSED = SCR.DATE_DISPOSED

SCR. MILES_HRS_LIFE = '000000'
Yes
Set MSTR. MILES_HRS_LIFE = SCR. MILES_HRS_LIFE
No

SCR. FUEL_COST_LIFE = '000000'
Yes
Set MSTR. FUEL_COST_LIFE = SCR. FUEL_COST_LIFE
No

SCR. LUBRICANTS_LIFE = '000000'
Yes
Set MSTR. LUBRICANTS_LIFE = SCR. LUBRICANTS_LIFE
No

SCR. TIRES_TUBES_LIFE = '000000'
Yes
Set MSTR. TIRES_TUBES_LIFE = SCR. TIRES_TUBES_LIFE
No

SCR. EXPEN_PARTS_LIFE = '000000'
Yes
Set MSTR. EXPEN_PARTS_LIFE = SCR. EXPEN_PARTS_LIFE
No

to 4.6
from 4.5

if SCR. ANTI FREEZE LIFE <= 0000000?

set MSTR. ANTI FREEZE LIFE = SCR. ANTI FREEZE LIFE

if SCR. PARTS COST LIFE > 00000000

set MSTR. PARTS COST LIFE = SCR. PARTS COST LIFE

if SCR. LABOR COST LIFE > 00000000

set MSTR. LABOR COST LIFE = SCR. LABOR COST LIFE

if SCR. IND. COST LIFE > 00000000

set MSTR. IND. COST LIFE = SCR. IND. COST LIFE

add BOOK VALUE = Amount from MSTR Record to TOTAL BOOK VALUE

write a record to new master file (NEQMAST) = MSTR Updated by SCR

print out the updated record

was equipment in record printed disposed of?

if yes, print a second line disposal method & date

else, to 4.7
from 4.6

Set LAST_CNTRY_NUM = MSTR_CNTRY_NUM

Read a Record from Equip Master File (MSTR)

End of File?

Yes

SCDONE = 'Y'?

No

DONE:

Write a Record to the Book Value File

Yes

Read a Record from the Status Change Records File (SCR)

End of File?

Yes

EMDONE = 'Y'?

No

DONE:

Write a Record to the Book Value File

Yes

Set EMDONE = 'Y', MSTR_CNTRY_NUM = '999', MSTR_EQ_NUM = '99999999'

No

Set SCDONE = 'Y', SCR_CNTRY_NUM = '999', SCR_EQ_NUM = '99999999'

End

No

to 4.2
from 4.2

LAST_CNTY_NUM=SCR.COUNTY_NUM?

No

Set THIS_CNTY_NUM=SCR.COUNTY_NUM

Set TOTAL_BOOK_VALUE=0

No

LAST_CNTY_NUM='000'

Yes

Print Heading on New Page for County Name in SCR.

Write a Record to the Book Value File

Move the Record Stored in the MSTR I/O Area to a Temporary Storage Area

Set MSTR to All Zeros, then, Move All Fields in SCR Area to Common Areas in MSTR.

Add BOOK_VALUE Amount from MSTR Record to TOTAL_BOOK_VALUE

Write a Record to New Master File (NEOMAST) New Record Added to File

Print Out the Record Just Added to the New Master File

Set LAST_CNTY_NUM=MSTR.COUNTY_NUM; Move Record in Temporary Area to MSTR I/O Area

1

4.9
Start

FMAIN:
Define Explicitly Variables and Arrays Used

Initialize County Name File; Set Depreciation Lifetimes for all Equip. Classes

Set End of File Conditions for all 4 Input Files; Open all Files

Read a record from the indirect cost file (IND_COST)

End of File?

Yes

No

Read a record from the book value file (BKVAL0)

End of File?

Yes

No

Calculate ratio of Total Indirect Cost to Total Book Value for each County and put in RATIO array

AA to 5.2
from 5.2

AF

Sort new master file (TEQMAST) by county, subunit, user class & requisites

SMAIN:

Define explicitly variables & arrays to be used. Define I/O areas to be used for input file

Initialize County Names Array; Initialize other variables to 0

Set up output print formats; Calculate current year from built-in function

Set ENDFILE and ENDPAGE conditions; open input & output files

Read a record from the master file (TEQMAST)

End of FILE

Yes

No

Set LAST_CNTY_NUM = COUNTY_NUM from record read

A H

to 5.5

Yes

No

J5 + --

H5 + --

F5 + --

E5 + --

D5 + --

E4 + --

D4 + --

E3 + --

D3 + --

E2 + --

D2 + --

E1 + --

D1 + --

A F

from 5.2

Sort new
master file
(TEDMAST)
by

county,
su

unit,

user
class &
req

u

ites

Define explicitly
variables & arrays
to be used. Define
I/O areas to be
used for input file

Initialize County
Names Array; Initialize other
variables to 0

Set up output
print formats; Calculate
current year from
built-in function

Set ENDFILE
and ENDPAGE
conditions; open
input & output
files

Read a record
from the
master file
(TEDMAST)

End of FILE

Set LAST_CNTY_NUM = COUNTY_NUM from record read

A H

to 5.5

Yes

No

J5 + --

H5 + --

F5 + --

E5 + --

D5 + --

E4 + --

D4 + --

E3 + --

D3 + --

E2 + --

D2 + --

E1 + --

D1 + --

A F
Add all appropriate data for this record to running totals for this class;

Set 
LAST_EQ_NUM = EQ_NUM;
LAST_CLASS = CLASS_CODE;
LAST_CNTY_NUM = CNTY_NUM

Read a record from the master file (FEQMAST)

End of File? 
Yes → AG
No → F3

CNTY_NUM from record just read, LAST_CNTY_NUM
No → Call Procedure UPONTYT (which adds class totals to county running cost totals)
Yes → Call Procedure UPONTYT

Call Procedure PRCLASSST (which calculates class CPM/CPH and print totals)

Print County (grand) totals on current vs. output files

Set all class running total variables and all county running total var. to zero

See logic for PRCLASSST on sheet 5.7
Procedure PRCLASST

Start

CLASS_MHRSL = 0 ?
Yes
Set the whole class direct cost, direct + indirect cost, & direct + indirect + depreciation CPM/CPH to 0

No
Calculate Class direct, direct + indirect, & direct + indirect + deprec. CPM/CPH for the current year

CLASS_MHRS1 = 0 ?
Yes
Set the class direct cost, direct + indirect cost, & direct + indirect + deprec. cost CPM/CPH life figures to 0

No
Calculate class direct + indirect, & direct + indirect + deprec. CPM/CPH for the life figures

Print Class Totals and CPM/CPH on all four output print files

End

from 54 85.6
DONE:

Call Procedure UPCNTY

Print County Grand Totals on current yr. output files

End

Call Procedure PRCLASST
Start

Sort the updated equipment master file by eq. class, district no, and city no.

ECCLSUM:

Define input file area, county name. Array B declare explicitly variables used - initialize to 0.

Calculate current year using built-in function.

Print heading using ON ENDPAGE capability.

REPEAT:

Read a record from sorted master file (UEMF).

End of File?

Yes

No

CLASS CODE = 0?

Yes

No

Print equipment class description heading and district number.

AP to 6.2

Yes

No

AN to 6.4

JS to 6.5

GO to 6.1
Add Current Year & Life Miles or Hours to running totals for this county

Add total direct costs to running totals by county

Add 1 to the no. of pieces of eqpt. for this county, Set OLD.COUNTY = COUNTY_NO of record read, OLD.DISTRICT = DISTRICT_NO of record read, OLD.EQUIP_CLASS = CLASS_CODE of record read

Read a record from sorted master file (UEMF)

End of File?

Calculate current year and life CPM/CFH for OLD.COUNTY

Add current year & life county total direct costs and miles or hours to district running totals, add county equipment total to district equip. running total.

to 6.3
Program No.: 6
Chart ID: 6.4
Chart Name: CTYSUMRY

Diagram:

1. From 6.3 AT
2. OLD_EQUIP_CLASS
3. OLD_EQUIP_CLASS
4. Yes
   - Calculate current year & life CPM/CPH for OLD_EQUIP_CLASS
5. No
   - Print CPH/CPM for current year & life and no. of pieces of equip. for OLD_EQUIP_CLASS
6. Set the state running totals variables for direct cost currently & life, miles or hours current year & life & pieces of equip. to zero
7. Print heading for new equip class and district
8. To 6.2
9. From 6.1 & 6.2
10. AN
11. DONE:
12. Calculate current year & life CPM/CPH for OLD_COUNTY
13. Add current year and life county total direct costs and miles or hours to district running totals. Add county equip. total to district running total
14. To 6.3
Calculating CPM/CPH for OLD_DISTRICT:

1. From 6.4, AV
2. Calculate current and life CPM/CPH for OLD_DISTRICT
3. Add current year and life district total direct costs and miles or hours to state running totals.
4. Add district equip total to state running total.
5. Calculate current and life CPM/CPH for OLD_EQUIP_CLASS
6. Print out CPM/CPH for current yr. & life for county district, & state total no. of pieces of equip.

End
Start


MFRAGE
Define Explicitly Variables Used and Initialize to O; Set Up I/O Area

Calculate Current Year Using Built-In Function—Calculate Past Year

Get First Record from MFR File—No. of Mfr Names to be Read

Allocate Space to Mfr Name Array

Set up ON ENDPAGE Condition to Print Headings

Print Heading Using ON ENDPAGE Capability

Get a Record from MFR File (mfr Code & Mfr Name)

HERE:

from 7.2

AX

to 7.2

AW

K4

K5

JS
from 7.1

End of File?

No

Yes

Next:

Read a Sorted Record from Equip Master File (UEMP)

End of File?

Yes

No

Place Mfr. Name Just Read in MFR_NAMES Array

CLASS CODE=2?

No

Yes

Print Heading: Equipment Class Description

Calculate Age Group of Piece of Equip on Record Just Read

Add 1 to No. of Vehicles in Age Group Calculated Above; Add Miles or Hours from Record Read, Both Life & Current Year to Approp. Age Group; Calculate Total Direct Costs, Life & Current Yr., and Add to Appropriate Age Group

Loop

from 7.3

BA

to 7.3

NEXT:

AX

to 7.1

EN
from 7.2

Set OLD_EQUIP_CLASS = CLASS_CODE & OLD_MFR_CODE = MFR_CODE

Read a Sorted Record from Equip Master File (UEMF)

End of File?
Yes

to 7.4

OLD_MFR_CODE = MFR_CODE or
OLD_EQUIP_CLASS < CLASS_CODE

Yes
Calculate Current Year and Life CPM/CPH for All Age Groups in OLD_MFR_CODE

No

Print Out OLD_MFR_CODE Name & CPW/CPH Year & Life Figures for Age Groups Repres.

Zero Out Arrays of Running Totals for Mfr. Year & Life Costs, Year & Life Miles or Hours, & No. of Vehicles (All Age Gps)

Print Out Heading with New Equip Class Description

OLD_EQUIP_CLASS < CLASS_CODE

Yes

No

to 7.2

BA
from
Z2 or 7.3

DONE:

Calculate Current
Year and Life
CPM/CPH for All
Age Groups in
OLD_MFR_CODE

Print Out OLD_
MFR_CODE Name
& CPM/CPH Year
and Life Figures
for All Age
Groups Represented

END
COMPUTER PROGRAM LISTINGS

This section contains the computer program listing of each of the eight computer programs used in this system. All programs are written in the PL/1 programming language. DIRECT, INDCOST, STCHNGE, CTYSUMRY, and MFGAGE are two-step jobs, the first step being a sort performed by a utility sort program. MAIN is a three step job, the first step and third step are programs written in PL/1, with the second step being a utility sort program. The sort routine portions of these programs are not given in these listings, however, the necessary sort sequences are given in each of the Computer Operating Instructions in the first section of this Appendix.
Listing of Program DIRECT
STAT LEVEL NEST
1  ECCR: PROC OPTIONS(MAIN):

/*
**                      PROGRAM DIRECT
**                      */
**                      */
**                      */
**                      THIS PROGRAM READS ALL DIRECT COST SUMMARY FORM CARS,
**                      */
**                      CHECKS THEM FOR ERRORS, AND OUTPUTS THE CORRECT RECORDS
**                      TO THE DIRECT COST FILE. ERRORS FOR EACH COUNTY ARE
**                      LISTED.
**                      */
**                      */
**                      THIS PROGRAM IS ONE OF EIGHT COMPUTER PROGRAMS
**                      */
**                      WHICH FORM THE "COMPUTER BASED INFORMATION SYSTEM
**                      FOR COUNTY EQUIPMENT COST RECORDS."
**                      */
**                      */
**                      WRITTEN BY
**                      */
**                      */
**                      SYSTEMS DIVISION
**                      */
**                      COLLEGE OF ENGINEERING
**                      */
**                      THE UNIVERSITY OF IOWA
**                      */
**                      IOWA CITY, IOWA
**                      */
**                      JULY, 1975
**                      */
ECCR: PROC OPTIONS(MAIN);

SYMT LEVEL NEST

2 1 DECLARE SDCOST FILE RECORD SEQUENTIAL;
3 1 DECLARE ERROr FIXED DEC(5) INIT(0);
4 1 DECLARE FLAG FIXED DEC(5) INIT(2);
5 1 DECLARE LAST_CNTY_NO CHAR(3) INIT('AAA');

6 1 DCL 1 DIRECT_COST_SUMMARY_RECORDS,
    2 COUNTY_NUMBER CHAR(3),
    2 EQUIPMENT_NUMBER CHAR(8),
    2 C_CATE,
    3 MONTH CHAR(2),
    3 YEAR CHAR(2),
    2 DIRECT_COSTS CHAR(50);

7 1 DECLARE CCOST FILE RECORD SEQUENTIAL;
8 1 DECLARE COLUMN REAL FIXED BIN(15,0);
9 1 DECLARE 1 C_DATE,
    2 C_YEAR CHAR(2),
    2 MCN_DAY CHAR(4);

10 1 ON EOF(FILE(SDCOST)) GO TO DONE;

12 1 OPEN FILE(SDCOST) INPUT;
12 1 OPEN FILE(DCCOST) OUTPUT;
14 1 C_DATE = DATE;
STMT LEVEL NEST

15 1 NEXT:
    READ FILE(DOCOST) INTO (DIRECT_COST_SUMMARY_RECORDS);

16 1 IF LAST_CNTY_NO = COUNTY_NUMBER
17 1 THEN DO:
18 1 1 IF FLAG = 0
19 1 1 THEN PUT SKIP(3) EDIT('NO ERRORS FOUND') (X(5), A):
20 1 1 PUT PAGE EDIT('ERROR LISTING FOR DIRECT COST SUMMARY FORMS',
    ' - COUNTY CODE *COUNTY_NUMBER')(CCL(32), A, A, A(3));
21 1 1 PUT SKIP(3)
22 1 1 LAST_CNTY_NO = COUNTY_NUMBER;
23 1 1 FLAG = 0;
24 1 1 END:

25 1 IF COUNTY_NUMBER < '001' OR COUNTY_NUMBER > '099'
26 1 THEN DO:
27 1 1 PUT SKIP EDIT('THE FOLLOWING RECORD HAS AN ILLEGAL COUNTY CODE.')
    (X(5), A);
28 1 1 ERROR = 1;
29 1 1 END;

30 1 IF MONTH > '12'
31 1 THEN DO:
32 1 1 PUT SKIP EDIT('THE FOLLOWING RECORD HAS AN INVALID MONTH VALUE')
    (X(5), A);
33 1 1 ERROR = 1;
34 1 1 END;

35 1 IF YEAR > C_YEAR
36 1 THEN DO:
37 1 1 PUT SKIP EDIT('THE FOLLOWING RECORD HAS AN INVALID YEAR VALUE')
    (X(5), A);
38 1 1 ERROR = 1;
39 1 1 END;

40 1 COLUMN = VERIFY(DIRECT_COSTS, '0123456789') + 15;

41 1 IF COLUMN = 15
42 1 THEN DO:
43 1 1 PUT SKIP EDIT('THE FOLLOWING RECORD HAS AN ILLEGAL CHARACTER',
    ' AT POSITION *COLUMN*') (X(5), A, A, F(2), A);
44 1 1 ERROR = 1;
45 1 1 END;

46 1 IF ERROR = 1
47 1 THEN DO:
48 1 1 PUT SKIP(2) EDIT (DIRECT_COST_SUMMARY_RECORDS) (X(3), (5) A);
49 1 1 PUT SKIP(2);
50 1 1 FLAG = 1;
51 1 1 ERROR = 0;
52 1 1 END;

53 1 ELSE WRITE FILE(DOCOST) FROM(DIRECT_COST_SUMMARY_RECORDS);
54 1 GO TO NEXT;
STAT LEVEL NEST

55   1   DONE: IF FLAG = 0 THEN PUT SKIP(3) EDIT(*NO ERRORS*)\{X(5),A\};
57   1   END EDGR;
Listing of Program INDCOST
/* EDIT INDIRECT COST RECORDS */

/* EDIT INDIRECT COST RECORDS */
EICR: CRCC OPTIONS(MAIN);

/* PROGRAM INDIRECT */
/* */
/* THIS PROGRAM READS ALL INDIRECT COST FORMS, CHECKS */
/* THEM FOR ERRORS, AND OUTPUTS THE CORRECT RECORDS TO */
/* THE INDIRECT COST FILE. ERRORS FOR EACH COUNTY ARE */
/* LISTED. */
/* */
/* THIS PROGRAM IS ONE OF EIGHT COMPUTER PROGRAMS */
/* WHICH FORM THE "COMPUTER BASED INFORMATION SYSTEM */
/* FOR COUNTY EQUIPMENT COST RECORDS."
/* */
/* WRITTEN BY */
/* */
/* SYSTEMS DIVISION */
/* COLLEGE OF ENGINEERING */
/* THE UNIVERSITY OF IOWA */
/* IOWA CITY, IOWA */
/* JULY, 1975 */
DECLARE SICGST FILE RECORD SEQUENTIAL;
  1 INDIRECT_COST_RECORD;
  2 COUNTY_NUMBER CHAR(3);
  2 INDIRECT_COST CHAR(60);
DECLARE ICGST FILE RECORD SEQUENTIAL;
DECLARE COUNTY_CODE REAL FIXED BIN(15,0);
DECLARE LAST_CNTY_NC CHAR(3) INIT('AAA');
DECLARE ERROR FIXED DEC(5) INIT(1);
ON ENDFILE(SICGST) GG TO CCNE;
OPEN FILE(SICGST) INPUT;
OPEN FILE(ICGST) OUTPUT;
.ecit indirect cost records */

next:
read file(sicst) into (indirect_cost_record);

if last_cnty_no = county_number
then do:
  if error = 0 then put skip ecit ('no errors')(x(5),a);
  put page edit ('error listing for indirect cost form',
                 'countv code ',county_number)(col32,a,a,a(2));
  put skip(3);
  error = 0;
  last_cntv_no = county_number;
end:

if county_number < '001' or county_number > '099'
then county_code = 0;
else county_code = 1;
column = verify(indirect_cost,'0123456789') + 3;

if column = 3
then if county_code = 0
  then do:
    put skip edit('the following record has an illegal ',
                    'county code')(x(5),a,a);
    put skip edit(indirect_cost_record)(x(10),a,a);
    error = 1;
  end:
else do:
  write file(icost) from (indirect_cost_record);
end:

else if county_code = 0
  then do:
    put skip edit('the following record has an illegal county ',
                    'code and an illegal character at position ',
                    'column')(x(5),a,a,f(2));
    put skip edit(indirect_cost_record)(x(10),a,a);
    error = 1;
  end:
else do:
  put skip edit('the following record has an illegal ',
                    'character at position ',column)(x(5),a,a,f(2));
  put skip edit(indirect_cost_record)(x(10),a,a);
  error = 1;
end:

end if;

go to next;

done:
if error=0 then put skip ecit ('no errors')(x(5),a);
end ecit;
Listing of Program STCHNGE
DECLARE THE STRUCTURES TO HOLD THE INPUT CARDS

DECLARE TSC FILE RECORD SEQUENTIAL,
   1 CARD1,
   2 COUNTY_NUMBER CHAR(3),
   2 EQUIPMENT_NUMBER CHAR(8),
   2 CARD_TYPE1 CHAR(11),
   2 DISTRICT CHAR(2),
   2 CLASS_CODE CHAR(2),
   2 CLASS_DESCRIPTION CHAR(20),
   2 MANUFACTURER_CODE CHAR(3),
   2 YEAR_EQ_MANUFACTURED CHAR(2),
   2 MAKE_MODEL_SERIAL_NO CHAR(28),
   2 WHEELBASE CHAR(3),
   2 DATE_PURCHASED CHAR(2),
   3 MONTH CHAR(2),
   3 DAY CHAR(2),
   3 YEAR CHAR(2),
   2 FILLER CHAR(2),

DECLARE TSC FILE RECORD SEQUENTIAL,
   1 CARD2,
   2 CTY_EQ_NO CHAR(11),
   2 CARD_TYPE2 CHAR(1),
   2 TYPE_OF_ENGINE CHAR(1),
   2 ENGINE_MAKE_MODEL_DESC CHAR(14),
   2 ENG_MAF_CODE CHAR(2),
   2 RATED_SPEED_WATERSPEED CHAR(3),
   2 NO_CYLINDERS CHAR(2),
   2 TRANS_TYPE CHAR(1),
   2 COMPANY_PUR_FROM CHAR(12),
   2 COST_VALUE CHAR(23),
   2 CHNG_STATUS_CODE CHAR(1),
   2 DATE_CHNG_STATUS CHAR(2),
   3 MONTH CHAR(2),
   3 DAY CHAR(2),
SCR: PROC OPTICAS(MAIN):

STMT LEVEL NEST

3 YEAR
2 FILLER  CHAR(2),
          CHAR(3):

4 1
DECLARE
  1 CARD3,
  2 CTY_NO_EC_NO2  CHAR(11),
  2 CAPD_TYPE3  CHAR(11),
  2 LIFE_COST  CHAR(57),
  2 FILLER  CHAR(11):
DECLARE THE STRUCTURE TO HOLD THE OUTPUT RECORD

DECLARE SCHEMA FILE RECORD SEQUENTIAL,
1 STATUSCHANGE_RECORD,
  2 COUNTY_NUMBER CHAR(3),
  2 EQUIPMENT_NUMBER CHAR(8),
  2 DISTRICT CHAR(2),
  2 CLASS_CODE CHAR(12),
  2 CLASS_DESCRIPTION CHAR(20),
  2 MANUFACTURED_CODE CHAR(3),
  2 YEAR_EQ_MANUFACTURED CHAR(2),
  2 MAKE_CCCEL_SERIAL_NO CHAR(28),
  2 WHEELBASE CHAR(3),
  2 DATE_PURCHASED,
  3 MONTH CHAR(2),
  3 DAY CHAR(2),
  3 YEAR CHAR(2),
  2 TYPE_OF_ENGINE CHAR(1),
  2 ENGINE_MAKE_MODEL_DSC CHAR(14),
  2 ENG_MFG_CODE CHAR(2),
  2 RATE_HOREPOWER CHAR(3),
  2 NO_CYLINDERS CHAR(2),
  2 SPANS_TYPE CHAR(1),
  2 COMPANY_PUR_FROM CHAR(12),
  2 COST_VALUE CHAR(23),
  2 CHANGES_STATUS_CODE CHAR(1),
  2 DATE_CHANGE_STATUS,
  3 MONTH CHAR(2),
  3 DAY CHAR(2),
  3 YEAR CHAR(2),
  2 LIFE_COST CHAR(57);
SCR: PROC OPTIONS(MAIN):

<table>
<thead>
<tr>
<th>Stmt Level</th>
<th>Nest</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1</td>
<td>OPEN FILE(SCHANGE) OUTPUT;</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>ON EOFFILE(TSC) GO TO DONE;</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>ON EOFFILE(SYSIN) GO TO NEXT;</td>
</tr>
</tbody>
</table>
/* READ IN CLASS DESCRIPTIONS FROM CARDS */
22 1 FIRST: GET EDIT(CODES,DESCRIPTIONS)(COL(0),F(2),A(20));
23 1 CL_DESC(CODES) = DESCRIPTIONS;
24 1 GO TO FIRST;

/* READ AND EDIT THE FIRST CARD OF AN EQUIPMENT STATUS */
/* CHANGE RECORD. */
25 1 NEXT: READ FILE(SC1) INTO(CARD1);
26 1 IF NEW_COUNTRY = CARD1.COUNTY_NUMBER THEN DO:
27 1 IF CNT = 0 THEN PUT SKIP EDIT(*ALL DATA SUBMITTED BY COUNTY 
28 1 NEW_COUNTRY. HAS BEEN EDITED AND NO ERRORS WERE FOUND.*) 
29 1 (COL(20),A,A,A);
30 1 ELSE CNT = 0;
31 1 PUT PAGE EDIT(*ERROR LISTING OF STATUS CHANGE RECORDS FOR COUNTY 
32 1 CARD1.COUNTY_NUMBER)(COL(35),A,A);
33 1 NEXT:
34 1 END:

35 1 IF (CARD1.COUNTY_NUMBER < '000') | (CARD1.COUNTY_NUMBER > '099') 
36 1 THEN DO:
37 1 FLAG = 1;
38 1 PUT SKIP EDIT(*THE COUNTY CODE IS INVALID*)(X(5),A);
39 1 END:

40 1 IF CARD_TYPE1 = '1' THEN DO:
41 1 FLAG = 1;
42 1 PUT SKIP EDIT(*CARD TYPE NOT 1 ON FIRST CARD.*)(X(5),A);
43 1 END:

44 1 IF (CARD1.CLASS_CODE < '00') & (CARD1.CLASS_CODE = ' ') & 
45 1 (CARD1.CLASS_CODE > '54') THEN DO:
46 1 FLAG = 1;
47 1 PUT SKIP EDIT(*CLASS CODE IS INVALID.*)(X(5),A);
48 1 END:

49 1 IF (CARD1.YEAR_EQ_MANUFACTURED < 'C_YEAR') | 
50 1 ((CARD1.YEAR_EQ_MANUFACTURED < '00') &
51 1 (CARD1.YEAR_EQ_MANUFACTURED = ' ')) THEN DO:
52 1 FLAG = 1;
53 1 PUT SKIP EDIT(*YEAR EQUIPMENT MANUFACTURED IS INVALID.*)(X(5),A);
54 1 END:

55 1 IF (CARD1.MANUFACTURER_CODE < '000') & (CARD1.MANUFACTURER_CODE = ' ') & 
56 1 (CARD1.MANUFACTURER_CODE > '135') THEN DO:
57 1 FLAG = 1;
58 1 PUT SKIP EDIT(*THE MANUFACTURER CODE NUMBER IS INVALID.*)(X(5),A);
STMT LEVEL NEST

55 1 1 END;

60 1 IF VERIFY(CARD1.WHEELBASE, '1234567890') ^= 0

61 1 THEN DO;

62 1 1 FLAG = 1;

63 1 1 PUT SKIP EDIT('THE WHEELBASE VALUE IS INVALID.')\$(5),A);

64 1 1 END;

65 1 IF ((CARD1.DATE_PURCHASED.MONTH < '01') & (CARD1.DATE_PURCHASED.MONTH ^= '')) | (CARD1.DATE_PURCHASED.MONTH > '12')

66 1 THEN DO;

67 1 1 FLAG = 1;

68 1 1 PUT SKIP EDIT('THE MONTH OF THE DATE PURCHASED IS INVALID.')\$(5),A);

69 1 1 END;

70 1 IF ((CARD1.DATE_PURCHASED.DAY < '01') & (CARD1.DATE_PURCHASED.DAY ^= '')) | (CARD1.DATE_PURCHASED.DAY > '31')

71 1 THEN DO;

72 1 1 FLAG = 1;

73 1 1 PUT SKIP EDIT('THE DAY OF THE DATE PURCHASED IS INVALID.')\$(5),A);

74 1 1 END;

75 1 IF (CARD1.DATE_PURCHASED.YEAR > C.YEAR) | 

76 1 IF (CARD1.DATE_PURCHASED.YEAR < '00') & (CARD1.DATE_PURCHASED.YEAR ^= ''))

77 1 THEN DO;

78 1 1 FLAG = 1;

79 1 1 PUT SKIP EDIT('THE YEAR OF THE DATE PURCHASED IS INVALID.')\$(5),A);

80 1 THEN DO;

81 1 IF CARD_TYPE1 ^= '1'

82 1 THEN DO:

83 1 1 CARD2 = BLANKS;

84 1 1 CARD3 = BLANKS;

85 1 1 GO TO TEST;

86 1 1 END;
START LEVEL NEXT

/* READ AND EDIT THE SECOND CARD OF AN EQUIPMENT STATUS */
/* CHANGE RECORD */

86 1 READ FILE(TSNC) INTO(CARD2);
87 1 IF CARD_TYPE2 = '2'
88 1 THEN DO;
89 1 1 FLAG = 1;
90 1 1 PUT SKIP EDIT('THE CARD TYPE FOR THE SECOND CARD IS NOT TYPE 2.')
91 1 1 (X(5),A);
92 1 1 END;
93 1 IF VERIFY(CARD2.TYPE_OF_ENGINE, 'G0') = 0
94 1 THEN DO;
95 1 1 FLAG = 1;
96 1 1 PUT SKIP EDIT('THE ENGINE FUEL CODE IS NOT A **G** OR **C**.')
97 1 1 END;
98 1 IF ((CARD2.ENG_MAF_CODE < '00') & (CARD2.ENG_MAF_CODE = '01')
99 1 THEN DO;
100 1 1 FLAG = 1;
101 1 1 PUT SKIP EDIT('THE ENGINE MANUFACTURER CODE IS INVALID.')
102 1 THEN END;
103 1 IF ((CARD2.RATED_HORSEPOWER < '000') &
104 1 (CARD2.RATED_HORSEPOWER = '04')
105 1 (CARD2.RATED_HORSEPOWER > '999'))
106 1 THEN DO;
107 1 1 FLAG = 1;
108 1 1 PUT SKIP EDIT('THE RATED HORSEPOWER IS INVALID.')
109 1 THEN END;
110 1 IF ((CARD2.NO_CYLINDERS < '02') & (CARD2.NO_CYLINDERS = '04')
111 1 (CARD2.NO_CYLINDERS < '06') & (CARD2.NO_CYLINDERS = '08') &
112 1 (CARD2.NO_CYLINDERS < '12') & (CARD2.NO_CYLINDERS = '16') &
113 1 (CARD2.NO_CYLINDERS < '32') & (CARD2.NO_CYLINDERS = '48')
114 1 THEN DO;
115 1 1 FLAG = 1;
116 1 1 PUT SKIP EDIT('THE NUMBER OF CYLINDERS IS INVALID.')
117 1 THEN END;
118 1 IF VERIFY(CARD2.TRANS_TYPE, 'AS') = 0
119 1 THEN DO;
120 1 1 FLAG = 1;
121 1 1 PUT SKIP EDIT('THE TRANSMISSION CODE IS INVALID.')
122 1 THEN END;
123 1 IF VERIFY(CARD2.COST_VALUE, '1234567890') = 0
124 1 THEN DO;
125 1 1 FLAG = 1;
126 1 1 PUT SKIP EDIT('THE ORIGINAL PURCHASE PRICE, SALVAGE VALUE, OR ')
127 1 'BOOK VALUE CONTAINS INVALID CHARACTER(S).')
128 1 THEN END;
IF VERIFY(CARD2.CHNG_STATUS_CODE='JST') = 0 THEN DO;
   FLAG = 1;
   PUT SKIP EDIT('THE SOLO, JUNKED, OR TRADED CODE FIELD CONTAINS AN ',
   'INVALID CHARACTER.')('X(5),A,A');
END;

IF (CARD2.DATE_CHNG_STATUS.MONTH < '01') &
   (CARD2.DATE_CHNG_STATUS.MONTH = '31') &
   (CARD2.DATE_CHNG_STATUS.MONTH > '12') THEN DO;
   FLAG = 1;
   PUT SKIP EDIT('THE MONTH OF DATE SOLD, JUNKED, OR TRADED IS ',
   'INVALID.')('X(5),A,A');
END;

IF (CARD2.DATE_CHNG_STATUS.DAY < '01') &
   (CARD2.DATE_CHNG_STATUS.DAY = '31') &
   (CARD2.DATE_CHNG_STATUS.DAY > '31') THEN DO;
   FLAG = 1;
   PUT SKIP EDIT('THE DAY OF DATE SOLD, JUNKED, OR TRADED IS ',
   'INVALID.')('X(5),A,A');
END;

IF (CARD2.DATE_CHNG_STATUS.YEAR > C.YEAR) &
   (CARD2.DATE_CHNG_STATUS.YEAR < '00') &
   (CARD2.DATE_CHNG_STATUS.YEAR = '11') THEN DO;
   FLAG = 1;
   PUT SKIP EDIT('THE YEAR OF THE DATE SOLD, JUNKED, OR TRADED IS ',
   'INVALID.')('X(5),A,A');
END;

IF CARD0.TYPE2 = 'Z' THEN DO;
   CARD3 = BLANKS;
   GO TO TEST;
END;
/* READ AND EDIT THE THIRD CARD OF AN EQUIPMENT STATUS */
/* CHANGE RECORD */

READ FILE(TSC) INTO(CARD3);

IF CARD_TYPE3 = '3'
THEN DO:

FLAG = 1;
PLT SKIP EDIT('THE CARD TYPE FOR THE THIRD CARD IS INVALID.')
(X(5),A);
END:

IF VERIFY(CARD3.LIFE_COST,'1234567890') = 0
THEN DO:

FLAG = 1;
PUT SKIP EDIT('THE LIFE INFORMATION CONTAINS AT LEAST ONE INVALID.')
('CHARACTER.')
(X(5),A,A);
END:

TEST: IF FLAG = 0

THEN DO: /* WRITE A RECORD WITH NO ERRORS TO OUTPUT FILE. */

CARD.DISTRICT = DISTRICT(CARD3,COUNTY_NUMBER);

IF CARD1.CLASS_CODE = '1' THEN

CARD1.DESCRIPTION = CL_DESC(CARD1,CLASS_CODE);

STATUS_CHANGE.RECORD = CARD1, BY NAME;

STATUS CHANGE.RECORD = CARD2, BY NAME;

STATUS CHANGE.RECORD = CARD3, BY NAME;

WRITE FILE(SCHANGE) FROM(STATUS CHANGE RECORD);

GO TO NEXT;

ELSE DO: /* PRINT A RECORD CONTAINING ERRORS ON THE PRINTER. */

PUT SKIP(2) EDIT(CARD1) (X(1),13)A;

PUT SKIP(2) EDIT(CARD2) (X(1),14)A;

PUT SKIP(2) EDIT(CARD3) (X(1),3,A);

COUNT = 1;
FLAG = 0;
GO TO NEXT;
END:

DONE: IF COUNT = 0

THEN PUT SKIP EDIT('ALL DATA SUBMITTED BY COUNTY * NEW COUNTY* HAS BEEN EDITED AND NO ERRORS WERE FOUND. ')
(COL(30),A,A,A);

END SCR;
Listing of Program UPDATEM
* * PROGRAM UPDATE
* * THIS PROGRAM INPUTS NEW STATUS CHANGE RECORDS AND
* * UPDATES EXISTING RECORDS WITH NEW DATA OR ADDS NEW
* * RECORDS TO THE EQUIPMENT MASTER FILE. TOTAL COUNTY
* * EQUIPMENT BOOK VALUES ARE CALCULATED. THE EQUIPMENT
* * INVENTORY LIST FOR THE PREVIOUS YEAR IS OUTPUT.
* * THIS PROGRAM IS ONE OF EIGHT COMPUTER PROGRAMS
* * WHICH FORM THE "COMPUTER BASED INFORMATION SYSTEM
* * FOR COUNTY EQUIPMENT COST RECORDS."
* * WRITTEN BY
* *
* * SYSTEMS DIVISION
* * COLLEGE OF ENGINEERING
* * THE UNIVERSITY OF IOWA
* *
* * IOWA CITY, IOWA
* *
* * JULY, 1975
**STAT LEVEL NEST**

**DCL STCHNC FILE RECORD SEQUENTIAL:**

2 1 DCL STCHNC 1 SCR;

2 COUNTY_NUM CHAR(3) INIT('999'),

2 FC_NUM CHAR(8),

2 DISTRICT CHAR(2),

2 CLASS_CODE CHAR(2),

2 CLAC.DESC CHAR(20),

2 MFR_CODE CHAR(3),

2 YEAR_MFD CHAR(2),

2 MAKE_Model CHAR(14),

2 SERIAL_NUM CHAR(14),

2 WHEELBASE CHAR(3),

2 DATE_PURCHASED CHAR(1),

3 PMCN CHAR(2),

3 FDAY CHAR(2),

3 PYR CHAR(2),

2 ENGINE_TYPE CHAR(1),

2 ENGINE_DESC CHAR(14),

2 ENGINE_MFR_CODE CHAR(2),

2 FATED_LP CHAR(3),

2 NUM_CYL CHAR(2),

2 TRMSTRY_TYPE CHAR(13),

2 CP_PURCHASED_FROM CHAR(12),

2 ORIG_COST PIC '9999999999',

2 SALVAGE_VALUE PIC '9999999999',

2 BCK.VALUE PIC '9999999999',

2 DISPOSITION CHAR(11),

2 DATE_DISPOSED CHAR(2),

3 PMCN CHAR(2),

3 YYR CHAR(2),

2 MILES_HRS.LIFE CHAR(6),

2 FUEL_COST_LIFE CHAR(7),

2 LUBRICANTS_LIFE CHAR(6),

2 TIRES_TUBES.LIFE CHAR(6),

2 EXPN.PARTS.LIFE CHAR(6),

2 ANTIFREEZE.LIFE CHAR(5),

2 PARTS_COST.LIFE CHAR(7),

2 LABOR_COST.LIFE CHAR(7),

2 IND.COST.LIFE CHAR(7);
DCL ECHAST.FILE RECORD SEQUENTIAL.
1 MSTR:
  2 COUNTY.NUM   CHAR(3) INIT(*998).
  2 DISTRICT    CHAR(2),
  2 FC_NUM      CHAR(8),
  2 CLASS_CODE  CHAR(2),
  2 YEAR_POD    CHAR(2),
  2 MFR_CODE    CHAR(3),
  2 ENGINE_MFR_CODE  CHAR(2),
  2 NUM_CARS    CHAR(2),
  2 ENGINE_TYPE  CHAR(1),
  2 TRANSM_TYPE  CHAR(1),
  2 PROG_COST   PIC S9(5)999999999.
  2 SALVAGE_VALUE PIC S9(5)9999999999.
  2 BOOK_VALUE  PIC S9(5)9999999999.
  2 CURRENT_COSTS CHAR(12),
  2 MILES_HRS_LIFE  CHAR(6),
  2 FUEL_COST_LIFE  CHAR(6),
  2 LUBRICANTS_LIFE  CHAR(6),
  2 TIRES_TUBES_LIFE  CHAR(6),
  2 EXPNS_PANTS_LIFE  CHAR(6),
  2 ANT_FREEZE_LIFE  CHAR(5),
  2 PARTS_COST_LIFE  CHAR(7),
  2 LMOD_COST_LIFE  CHAR(7),
  2 TRAN_COST_LIFE  CHAR(7),
  2 CLASS_DESC  CHAR(20),
  2 MAKE_MODEL  CHAR(14),
  2 SERIAL_NUM         CHAR(14),
  2 ENGINE_DESC  CHAR(14),
  2 CC_PURCHASED_FRC  CHAR(12),
  2 DATES_PURCHASED,
    3 PMN        CHAR(2),
    3 PDAY       CHAR(2),
    3 PYR        CHAR(2),
  2 DATES_DISPOSED,
    3 DMDN       CHAR(2),
    3 CDAY       CHAR(2),
    3 CYR        CHAR(2),
  2 DISPOSITION   CHAR(1),
  2 WHEELBASE    CHAR(3),
  2 REPAIRS     CHAR(13),
  2 RATED_HP     CHAR(3),
  2 BLANKS      CHAR(24);
STMT LEVEL NEST

5 1   DCL 1 CNTY_BOOKVALU
     2 LAST_CNTY_NUM   CHAR(3) INIT('000');
     2 TOTAL_BOOKVALU   PIC 9(9999999999) INIT((10)*0');
6 1   DCL DISP  CHAR(6) VARYING;
7 1   DCL LETTERS  FIXED DEC(5);
8 1   DCL THIS_CNTY_NUM  CHAR(3);
9 1   DCL SCOCONE  BIT(1) INIT('0*8');
10 1   DCL ENDONE  BIT(1) INIT('0*8');
11 1   DCL 1 TODAYS_DATE;
     2 C_YEAR   CHAR(2);
     2 C_MON_DAY  CHAR(4);
12 1   TODAYS_DATE = DATE;
13 1   DCL LAST_YEAR   FIXED DEC(4);
14 1   LAST_YEAR = 1899 + C_YEAR;
15 1   DCL 1 SAVSTR  LIKE MSTR;
16 1   DCL NECKAST  FILE  RECORD  SEQUENTIAL;
17 1   DCL BKVL1U  FILE  RECORD  SEQUENTIAL;
18 1   EQ_INV_LIST:  FORMAT  (COL(3), A, COL(13), A, COL(28), A, COL(43), A,
                   COL(57), A, COL(63), A, COL(69), A, COL(74), A, COL(78), A, COL(82), A,
                   COL(103), A, COL(112), P*ZZZZZZ*99, A, COL(122), P*ZZZZZZ*99*);
15 1   DCL CNTY_NAMES(99)  CHAR(13) VARYING
     INIT('ADFORD', 'ADAMS', 'ALLAMAKEE', 'APPANOOSA', 'AUDUBON', 'BENTON',
      'BLACK HAWK', 'BCEME', 'BREMNER', 'BUCHANAN', 'BUENA VISTA', 'BUTLER',
       'CABARRUS', 'CAROLINA', 'CASS', 'CEDAR', 'CERCRO', 'CHEROKEE',
       'CHICKASAW', 'CLAY', 'CLAYTON', 'CLINTON', 'CRAWFORD', 'DALLAS',
       'DAVIS', 'DECATUR', 'DELWARE', 'DES MINES', 'DICKINSON', 'DLBURKE',
       'EFMET', 'FAYETTE', 'FLOYD', 'FRANKLIN', 'FRANKLIN', 'GREENE', 'GRundy',
       'GUTHRIE', 'HAMILTON', 'HANCOCK', 'HAWK', 'HARISON', 'HANRY',
       'HARRISON', 'HUMBOLDT', 'IDA', 'IDADA', 'JACKSON', 'JAPET', 'JEFFERSON',
       'JHYLSON', 'JONES', 'KANSAS', 'KANSAS', 'KENNEDY', 'LICK', 'LAWRENCE',
       'LYNN', 'MACISCH', 'MADISON', 'MARION', 'MARSHALL', 'MILLS', 'MITCHELL',
       'MONDA', 'MCNINNE', 'MONTGOMERY', 'MUSCATINE', 'C*BRIEN', 'OSCEOLA',
       'PAIGE', 'PAOL ALTO', 'PLYMOUTH', 'POCAHONTAS', 'PCL', 'POTAWATOMIE',
       'PKERSHIEK', 'RINGGOLD', 'SAC', 'SCEY', 'SHELBY', 'SIOUX', 'SUGAR',
       'TAMA', 'TAYLOR', 'UNION', 'VALE BURES', 'WAPELLO', 'WARREN',
       'WASHINGTON', 'WAYNE', 'WEBSTER', 'WINNEBAGO', 'WINNEBOSI', 'WOODBURY',
       'WORTH', 'WRIGHT');
ON FILE(ECMAST) BEGIN:
    IF SCOCNE = '1'B
    THEN GO TO DONE;
    ELSE DO:
        SCOCNE = '1'B;
        MSTR.COUNTY_NUM = '999':
        MSTR.EQ_NUM = '995999999';
        END;
    END:
ON FILE(StCHNG) BEGIN:
    IF SCOCNE = '1'B
    THEN GO TO DONE;
    ELSE DO:
        SCOCNE = '1'B;
        SCR.COUNTY_NUM = '999';
        SCR.EQ_NUM = '995999999';
        END;
    END:
OPEN FILE(ECMAST) INPUT;
OPEN FILE(STCHNG) INPUT;
OPEN FILE(NEQMAST) OUTPUT;
OPEN FILE(BKVAL) OUTPUT;
OPEN FILE(SYSPRINT) LINESIZE(132) PAGESIZE(60);
READ FILE(ECMAST) INTO (MSTR);
READ FILE(STCHNG) INTO (SCR);
NEXT:

IF SCR.COUNTY_NUM > MSTR.COUNTY_NUM OR SCR. EC_NUM > MSTR. EC_NUM
THEN DO: /* COPY MASTER FILE RECORD */

IF LAST_CNTY_NUM ^= MSTR.COUNTY_NUM
THEN DO:
  THIS_CNTY_NUM = MSTR.COUNTY_NUM;
  CALL HEADING;
  IF LAST_CNTY_NUM ^= '000'
  THEN WRITE FILE(BKVALU) FROM(CNTY_BOOK_VALUE);
  ELSE:
  TOTAL_BOOK_VALUE = TOTAL_BOOK_VALUE + MSTR.BOOK_VALUE;
  ELSE:
  TOTAL_BOOK_VALUE = TOTAL_BOOK_VALUE + MSTR.BOOK_VALUE;

WRITE FILE(NEXTAST) FCSH (*MSTR*);

PUT SKIP EDIT(*MSTR.EC_NUM, MSTR.MAKE_MODEL, *MSTR.SERIAL_NUM*,
MSTR.ViewChild, *MSTR.YEAR_MFD, MSTR.WHEELBASE, MSTR.XMISSION_TYPE, MSTR.NUM_CYL, MSTR.ENGINE_TYPE,
MSTR. RATED HP, MSTR.ENGINE_DESC, MSTR.DATE_PURCHASED, *MON*, '-1',
MSTR. DATE_PURCHASED, *DAY*, '-1', MSTR.CATE_PURCHASED, *PYR*,
MSTR.ORIG_COSY, MSTR.BOOK_VALUE)(RLEC_INV_LIST);

LAST_CNTY_NUM = MSTR.COUNTY_NUM;
READ FILE(ECMAST) INTO (MSTR);
GO TO NEXT;
END:
ELSE IF SCR.COUNTY_NUM = MSTR.COUNTY_NUM & SCR.ENG_NUM = MSTR.ENG_NUM THEN DO: /* UPDATE EXISTING RECORD ON MASTER FILE */
IF LAST_CNTY_NUM = SCR.COUNTY_NUM THEN DO:
   THIS_CNTY_NUM = SCR.COUNTY_NUM;
   CALL HEADING;
   IF LAST_CNTY_NUM = 'OO'
   THEN WRITE FILE(SKVALU) FROM(CNTY_BOOK Valu);
   WRITE FILE(SKVALU) FROM(CNTY_BOOK Valu);
   TOTAL_BOOK_VALU = (10)0';
   END;
ELSE:
   IF SCR.CLS_CODE >= '00'
   THEN DO:
      MSTR.CLS_CODE = SCR.CLS_CODE;
      MSTR.CLS_DESC = SCR.CLS_DESC;
      END;
   ELSE:
      IF SCR.MFR_CODE > 'OO'
      THEN MSTR.MFR_CODE = SCR.MFR_CODE;
   ELSE:
      IF SCR.YEAR_MFC > '00'
      THEN MSTR.YEAR_MFC = SCR.YEAR_MFC;
   ELSE:
      IF SCR.MAKE.Model > (14)'
      THEN MSTR.MAKE.Model = SCR.MAKE.Model;
      IF SCR.SERIAL_NUM > (14)'
      THEN MSTR.SERIAL_NUM = SCR.SERIAL_NUM;
      IF SCR.WHEELBASE > '000'
      THEN MSTR.WHEELBASE = SCR.WHEELBASE;
      IF SCR.DATE_PURCHASED.PKON > '00'
      THEN MSTR.DATE_PURCHASED.PKON = SCR.DATE_PURCHASED.PKON;
      IF SCR.DATE_PURCHASED.PAY > '00'
      THEN MSTR.DATE_PURCHASED.PAY = SCR.DATE_PURCHASED.PAY;
      IF SCR.DATE_PURCHASED.PYR > '00'
      THEN MSTR.DATE_PURCHASED.PYR = SCR.DATE_PURCHASED.PYR;
      IF SCR.ENGINE_TYPE > '
      THEN MSTR.ENGINE_TYPE = SCR.ENGINE_TYPE;
      IF SCR.ENGINE_DESC > (14)'
      THEN MSTR.ENGINE_DESC = SCR.ENGINE_DESC;
      IF SCR.ENGINE_MFR_CODE > '00'
      THEN MSTR.ENGINE_MFR_CODE = SCR.ENGINE_MFR_CODE;
      IF SCR.RATED_HP > '000'
      THEN MSTR.RATED_HP = SCR.RATED_HP;
      IF SCR.NUM_CYLS > '00'
      THEN MSTR.NUM_CYLS = SCR.NUM_CYLS;
      IF SCR.TRANSMISSION_TYPE > '000'
**PRCC OPTIONS (MAIN):**

STAP LEVEL NEST

106  1  1  THEN MSTR.XMISION_TYPE = SCR.XMISION_TYPE;
107  1  1  IF SCR.CC_PURCHASED_FROM > '121' THEN MSTR.CC_PURCHASED_FROM = SCR.CC_PURCHASED_FROM;
108  1  1  IF SCR.ORIG_COST > 0 THEN MSTR.CROR.COST = SCR.ORIG_COST;
109  1  1  IF SCR.SALVAGE_VALUE > 0 THEN MSTR.SALVAGE_VALUE = SCR.SALVAGE_VALUE;
110  1  1  IF SCR.BOOK_VALUE > 0 THEN MSTR.BOOK_VALUE = SCR.BOOK_VALUE;
111  1  1  MSTR.DISPOSITION = SCR.DISPOSITION;
112  1  1  MSTR.DATE_DISP_SEC = SCR.DATE_DISP_SEC;
113  1  1  IF SCR.MILES_HRS_LIFE > '000000' THEN MSTR.MILES_HRS_LIFE = SCR.MILES_HRS_LIFE;
114  1  1  IF SCR.FUEL_COST_LIFE > '000000' THEN MSTR.FUEL_COST_LIFE = SCR.FUEL_COST_LIFE;
115  1  1  IF SCR.LUBRICANTS_LIFE > '000000' THEN MSTR.LUBRICANTS_LIFE = SCR.LUBRICANTS_LIFE;
116  1  1  IF SCR.TIRES_TUBES_LIFE > '000000' THEN MSTR.TIRES_TUBES_LIFE = SCR.TIRES_TUBES_LIFE;
117  1  1  IF SCR.EXPEN_PARTS_LIFE > '000000' THEN MSTR.EXPEN_PARTS_LIFE = SCR.EXPEN_PARTS_LIFE;
118  1  1  IF SCR.ANTIFREEZE_LIFE > '000000' THEN MSTR.ANTIFREEZE_LIFE = SCR.ANTIFREEZE_LIFE;
119  1  1  IF SCR.PARTS_COST_LIFE > '000000' THEN MSTR.PARTS_COST_LIFE = SCR.PARTS_COST_LIFE;
120  1  1  IF SCR.LABOR_COST_LIFE > '000000' THEN MSTR.LABOR_COST_LIFE = SCR.LABOR_COST_LIFE;
121  1  1  IF SCR.IND_COST_LIFE > '000000' THEN MSTR.IND_COST_LIFE = SCR.IND_COST_LIFE;
122  1  1  TOTAL_BOOK_VALUE = TOTAL_BOOK_VALUE + MSTR.BOOK_VALUE;
123  1  1  WRITE FILE(<MSTR.ENG_NAME>) FROM (MSTR);
124  1  1  PUT SKIP ECIT(MSTR.MAKE, MSTR.MODEL, MSTR.SERIAL_NUM, MSTR.CO_PURCHASED_FROM, '19', MSTR.YEAR_MFG, MSTR.WHEELBASE, MSTR.XMISION_TYPE, MSTR.NUM_CYL, MSTR.ENGINE_TYPE, MSTR.RATE_HP, MSTR.ENGINE_DESC, MSTR.DATE_PURCHASED, MSTR.POM, '19', MSTR.DATE_PURCHASED, MSTR.PEDAY, '19', MSTR.DATE_PURCHASED, MSTR.PYR, MSTR.ORIG_COST, MSTR.BOOK_VALUE) (REQ_INV_LIST):
DATA: PROC OPTIONS(MAIN);

START LEVEL NEST.

136 1 1 THEN DO:
140 1 2 IF MSTR.DISPOSITION = 'J'
141 1 2 THEN DISP = 'JUNKED';
142 1 2 IF MSTR.DISPOSITION = 'T'
143 1 2 THEN DISP = 'TRADED';
144 1 2 IF MSTR.DISPOSITION = 'S'
145 1 2 THEN DISP = 'SOLD';
146 1 2 PUT SKIP EDIT ('**',MSTR.EO_NUM,'WAS 'DISP,' CN ')
147 1 2 MSTR.DATE_DISPOSED.DAY,'MSTR.DATE_DISPOSED.DAY,' MSTR.DATE_DISPOSED.DAY (CCL(2),(10)A);
148 1 2 END;
149 1 1 LAST_CNTY_NUM = MSTR.COUNTY_NUM;
150 1 1 READ FILE(ECMAST) INTO (MSTR);
151 1 1 READ FILE(SICHING) INTO (SCR);
152 1 1 GO TO NEXT;
153 1 1 END;

PAGE 10
133  1  ELSE DO: /* ADD NEW RECORD TO MASTER FILE */
134  1 1  IF LAST_CNTY_NUM <> SCR.COUNTY_NUM
135  1 1  THEN DO:
136  1 2  THIS_CNTY_NUM = SCR.COUNTY_NUM;
137  1 2  CALL HEADING;
138  1 2  IF LAST_CNTY_NUM = 000
139  1 2  THEN WRITE FILE(BKVALU) FROM(CNTY_BOOK_VALUE);
140  1 2  TOTAL_BOOK_VALUE = (10)'0';
141  1 2  END;
142  1 1  ELSE:
143  1 1  SAVSTR = MSTR, BY NAME;
144  1 1  MSTR = (300)'0';
145  1 1  MSTR = SCR, BY NAME;
146  1 1  TOTAL_BOOK_VALUE = TOTAL_BOOK_VALUE + MSTR.BOOK_VALUE;
147  1 1  WRITE FILE (NEOMAST) FROM (MSTR);
148  1 1  PUT SKIP EDIT(MSTR.EQ_NUM,MSTR.MAKE_Model,MSTR.SERIAL_NUM,
149  1 2  MSTR.CO_PURCHASED_FROM,MSTR.YEAR_MFD,
150  1 2  MSTR.WHRESSALE,MSTR.TRANSMISSION_TYPE,MSTR.NUM_CYLS,
151  1 2  MSTR.ENGINE_TYPE,MSTR.RATED_HP,
152  1 2  MSTR.ENGINE_SEC,MSTR.DATE_PURCHASED,PMON,'-',
153  1 2  MSTR.DATE_PURCHASED, PYR,'-', MSTR.DATE_PURCHASED, PDR,
154  1 2  MSTR.Orig_Cost,MSTR.BOOK_VALUE)F(R(EQ_INV_LIST));
155  1 1  IF MSTR.DISPOSITION = 'J'
156  1 2  THEN DISP = 'JUNKED';
157  1 2  IF MSTR.DISPOSITION = 'T'
158  1 2  THEN DISP = 'TRADED';
159  1 2  IF MSTR.DISPOSITION = 'S'
160  1 2  THEN DISP = 'SOLD';
161  1 2  PUT SKIP EDIT (**MSTR.EQ_NUM,'HAS','DISP',' CN ','
162  1 2  MSTR.DATE_DISPOSED,PMON,'-',MSTR.DATE_DISPOSED,DAY,
163  1 2  MSTR.DATE_DISPOSED, YYYY(CCL(2),(10)A));
164  1 2  END:
165  1 1  LAST_CNTY_NUM = MSTR.COUNTY_NUM;
166  1 1  MSTR = SAVMSTR, BY NAME;
167  1 1  READ FILE(SYCNCG) INTO (SCR);
168  1 1  GO TO NEXT;
169  1 1  END:
HEADING: PROC;

LETTERS = LENGTH(CNTY_NAMES THIS_CNTY_NUM);

PUT PAGE EDIT(CNTY_NAMES THIS_CNTY_NUM CNTY)
       (COL(125 LETTERS/2) A A);

PUT SKIP(2) EDIT (LAST_YEAR ' EQUIPMENT INVENTORY LIST)
       (COL(51) F(4) A);

PUT SKIP(3) EDIT 'EQUIPMENT' 'YEAR WPB TRAN' 'ENGINE MAKE'
       'PURCHASE PURCHASE BOOK' (COL(3) A COL(57) COL(88) A)

PUT SKIP EDIT (NUMBER MAKE MODEL SERIAL NUMBER CEJLER)
       'XRDO INS TYPE CYL. FUEL HP AND MODEL'
       'DATE COST VALUE' (COL(4) A COL(57) A COL(105) A);

PUT SKIP(2);

RETURN;

END HEADING;

DOCE:

WRITE FILE(BKVALU) FROM (CNTY_BOOK VALU):

END UPDATE;
Listing of Program MAIN

Step 1
/* MAIN: PROC OPTIONS(MAIN): */

/*
  PROGAM MAIN
  /*
  THIS PROGRAM CALCULATES COST/MILE OR COST/HOUR FOR ALL
  PIECES OF EQUIPMENT, BOTH FOR PAST YEAR AND LIFETIME
  /*
  COSTS.
  /*
  OUTPUT CONSISTS OF YEARLY AND LIFETIME DIRECT
  OPERATING COSTS AND YEARLY AND LIFETIME TOTAL
  OPERATING COSTS.
  /*
  THIS PROGRAM IS ONE OF EIGHT COMPUTER PROGRAMS
  WHICH FORM THE "COMPUTER BASED INFORMATION SYSTEM"
  FOR COUNTY EQUIPMENT COST RECORDS."
  /*
  WRITTEN BY
  /*
  SYSTEMS DIVISION
  COLLEGE OF ENGINEERING
  THE UNIVERSITY OF IOWA
  IOWA CITY, IOWA
  JULY, 1975
  */
*/
PROC CPTCNS(MAIN):

STMT LEVEL NEST

DCL SEQMAST FILE RECORD SEQUENTIAL,

COUNTY_NUM PIC '999'
DISTRICT CHAR(2)
EQ_NUM CHAR(8)
CLASS_CODE PIC '99'
MISC CHAR(11)
ORIG_COST PIC '999999999'
SALVAGE_VALUE PIC '999999999'
BOOK_VALUE PIC '999999999'
DEPRECIATION PIC '999999999'
MILES_HRS_YR PIC '999999'
FUEL_COST_YR PIC '999999'
LUBRICANTS_YR PIC '999999'
TIMES_TUBES_YR PIC '999999'
EXPEND_PARTS_YR PIC '999999'
ANTIFREEZE_YR PIC '999999'
PARTS_COST_YR PIC '999999'
LABOR_COST_YR PIC '999999'
IND_COST_YR PIC '999999'
MILES_HRS_LIFE PIC '999999'
FUEL_COST_LIFE PIC '999999'
LUBRICANTS_LIFE PIC '999999'
TIMES_TUBES_LIFE PIC '999999'
EXPEND_PARTS_LIFE PIC '999999'
ANTIFREEZE_LIFE PIC '999999'
PARTS_COST_LIFE PIC '999999'
LABOR_COST_LIFE PIC '999999'
IND_COST_LIFE PIC '999999'
CLASS_DESC CHAR(2)
MAKE_MODEL CHAR(14)
SERIAL_NUM CHAR(14)
MISC2 CHAR(42)
DOWN_TIME_YR PIC '9999'
DOWN_TIME_LIFE PIC '9999'
NUM_REPAIRS_YR PIC '999'
NUM_REPAIRS_LIFE PIC '9999'
RATED_HP CHAR(3)
BLANKS CHAR(24)
DCL INOCOST FILE RECORD SEQUENTIAL,
  1 IND,
     2 COUNTY_NUM       PIC'99999',
     2 SUP_SALARIES     PIC'99999V999',
     2 CLER_SALARIES    PIC'99999V999',
     2 UTILITIES        PIC'99999V999',
     2 ALDS_COSTS       PIC'99999V999',
     2 EQ_DEPR          PIC'99999V999',
     2 EXP_TCOGS         PIC'99999V999',
     2 SUPPLIES         PIC'99999V999',
     2 MOVING_COSTS     PIC'99999V999',
     2 EQ_INS           PIC'99999V999';
*/MAIN: PROC OPTIONS(*MAIN);*

SMT LEVEL NEST

4 1  DCL CIRCOST  FILE RECORD SEQUENTIAL,
     1 DIR,
     2 COUNTY_NUM  PIC '9999' ,
     2 EQ_NUM  CHAR(4),
     2 DATE  CHAR(4),
     2 FUEL_COST_YR  PIC '999999' ,
     2 LUBRICANTS_YR  PIC '999999' ,
     2 ANTI_FREEZE_YR  PIC '999999' ,
     2 TIRES_TUBES_YR  PIC '999999' ,
     2 EXPENSE_PARTS_YR  PIC '999999' ,
     2 PARTS_COST_YR  PIC '999999' ,
     2 LABOR_COST_YR  PIC '999999' ,
     2 MILES_HRS_YR  PIC '999999' ,
     2 DOWN_TIME_YR  PIC '999999' ,
     2 NUM_REPAIRS_YR  PIC '999';
DCL BKVALU FILE RECORD SEQUENTIAL,
   1 BKVL,
   2 CNTY_NUM PIC '9999' VALUE PIC '9999999999';

DCL CNTY_NAMES(99) CHAR(13) VARYING
   INIT('ACORN', 'ADAMS', 'APPLETON', 'APPANOOSA', 'AURORA', 'BENTON',
       'BLACK HAWK', 'Benton', 'Buchanan', 'BUENA VISTA', 'Butler',
       'CALHOUN', 'CARROLL', 'CASS', 'CEDAR', 'CEPRO GORD', 'CHEYENNE',
       'CHICKASAW', 'CLAY', 'CLAYTON', 'CLINTON', 'CRAWFORD', 'DALLAS',
       'DAVIS', 'DECATUR', 'DELWARE', 'DES MOINES', 'DICKINSON', 'DLBQUE',
       'EMMET', 'FAYETTE', 'FLOYD', 'FRANKLIN', 'FRANKLIN', 'GREENE', 'GRundy',
       'GUTHRIE', 'HAMILTON', 'HANCOCK', 'HAMPDEN', 'HARPERS', 'HIN',
       'HOWER', 'HUMBOLDT', 'IOWA', 'JACKSON', 'JASPER', 'JEFFERSON',
       'JOHNSON', 'JONES', 'KEOKUK', 'KOSUTH', 'LEE', 'LINN', 'LOUISA', 'Lucas',
       'LYON', 'MADISON', 'MAHASKA', 'MARION', 'MARSHALL', 'MILLS', 'MIDDLE',
       'MONONA', 'MONROE', 'MONTGOMERY', 'MUSCATINE', 'O'BRIEN', 'OSCEOLA',
       'PAGE', 'PARK', 'PELHAM', 'Pocahontas', 'POLK', 'POTTAWATAMIE',
       'POWES', 'RACINE', 'RINGGOLD', 'SAC', 'SCOTT', 'SHELBY', 'SIoux', 'STORY',
       'TAMA', 'TAYLOR', 'TEN', 'VAN BUREN', 'WAPANOC', 'WARRN',
       'WASHINGTON', 'HAYNE', 'WEBSTER', 'WINNEBAGO', 'WINNESHIEK', 'WOODBURY',
       'WORTH', 'WRIGHT');
PROC OPTIONS(MAIN):
ON ENCFILE(TEQMST) BEGIN:

EMDSEE = '1'B;

IF EECHE = '1'B

THEN GO TO DCEM;

ELSE DO:

46 2 1

NOEQM:

IF DIT..COUNTY_NUM = LAST_EPR_CNTY

THEN DO:

47 2 1

PUT PAGE EDIT ('ERROR LISTING FOR ',

COUNTY_NAMES(DIT..COUNTY_NUM) , ' COUNTY') (COL(30),A,A,A);

49 2 2

LAST_EPR_CNTY = DIT..COUNTY_NUM;

50 2 2

END;

51 2 1

PUT SKIP(2) EDIT ('THE EQUIPMENT NUMBER IN THE FOLLOWING DIRECT ',

'COST RECORD WAS NOT FOUND IN THE MASTER FILE') (A,A);

52 2 1

PUT SKIP EDIT (DIT)(A);

52 2 1

READ FILE(DIRCOST) INTO (DIT);

54 2 1

END:

55 2 1

END;

56 2

DCL TEQMST FILE RECORD SEQUENTIAL:

57 1

OPEN FILE(SEQMST) INPUT;

58 1

OPEN FILE(SYSPRINT) LINESIZE(132) PAGESIZE(60);

59 1

OPEN FILE(INDCOST) INPUT;

60 1

OPEN FILE(DIRCOST) INPUT;

61 1

OPEN FILE(TEQMAST) OUTPUT;

62 1

OPEN FILE(TEQMAST) OUTPUT;
READ FILE(INCOST) INTO (INC);

DO WHILE (IND_CK);
   /* FIND TOTAL INDIRECT COST FOR COUNTY */
   TOT_IND_COST(INC,COUNTY_NUM) =
      SUP_SALARIES + CLER_SALARIES
      + UTILITIES + BLDG_COSTS + EQ_DEPR
      + EXP_TOLS + SUPPLIES + MOVING_COSTS
      + EC_INS;
   READ FILE(INCOST) INTO (INC);
END;

READ FILE(BKVALU) INTO (BKVAL);

DO WHILE (BKVAL_CK);
   TOT_BK_VAL(BKVAL,CNTY_NUM) = BKVAL.VALUE;
   PREAD FILE(BKVALU) INTO (BKVAL);
END;

INDEX = 1;
/* THE MAX VALUE OF INDEX MUST BE CHANGED AS CITIES ARE ADDED */

DO WHILE (INDEX <= 69);
   IF TOT_BK_VAL(INDEX) <= (10)**0;
      THEN RATIO(INDEX) = TOT_IND_COST(INDEX)/TOT_BK_VAL(INDEX);
   ELSE RATIO(INDEX) = (10)**0;
   END;
   INDEX = INDEX + 1;
END;
**STAT LEVEL NEST**

```
81 1 NEXT:
    READ FILE(SFQMST) (MT) (MSTR):
82 1 READ FILE(DIRCOST) INTO (CIR):
83 1 NEXT2:
    IF MSTR.COUNTY_NUM = DIR.COUNTY_NUM & MSTR.REGION_NUM = CIR.REGION_NUM
    THEN DO: /* UPDATE LIFE FIGURES FOR ALL EQUIPMENT */
84 1 1 MSTR.INI.IND_COST_YR = MSTR.BOOK_VALUE * RATIO(MSTR.COUNTY_NUM):
85 1 1 MSTR = DIR, BY NAME:
86 1 1 MSTR.MILES_HRS_LIFE = MSTR.MILES_HRS_LIFE + MSTR.MILES_HRS_YR:
87 1 1 MSTR.FUEL_COST_LIFE = MSTR.FUEL_COST_LIFE + MSTR.FUEL_COST_YR:
88 1 1 MSTR.LUBRICANTS_LIFE = MSTR.LUBRICANTS_LIFE + MSTR.LUBRICANTS_YR:
89 1 1 MSTR.TIRES_TUBES_LIFE = MSTR.TIRES_TUBES_LIFE + MSTR.TIRES_TUBES_YR:
90 1 1 MSTR.EXPEN_PARTS_LIFE = MSTR.EXPEN_PARTS_LIFE + MSTR.EXPEN_PARTS_YR:
91 1 1 MSTR.ANTIFREEZE_LIFE = MSTR.ANTIFREEZE_LIFE + MSTR.ANTIFREEZE_YR:
92 1 1 MSTR.PARTS_COST_LIFE = MSTR.PARTS_COST_LIFE + MSTR.PARTS_COST_YR:
93 1 1 MSTR.LABOR_COST_LIFE = MSTR.LABOR_COST_LIFE + MSTR.LABOR_COST_YR:
94 1 1 MSTR.DWN_TIME_LIFE = MSTR.DWN_TIME_LIFE + MSTR.DWN_TIME_YR:
95 1 1 MSTR.NUM_REPAIRS_LIFE = MSTR.NUM_REPAIRS_LIFE + MSTR.NUM_REPAIRS_YR:
96 1 1 MSTR.IND_COST_LIFE = MSTR.IND_COST_LIFE + MSTR.IND_COST_YR:
97 1 1 MSTR.DEPREC = (MSTR.ORIG_COST - MSTR.SALVAGE_VALUE) / DEPRECIATION_YRS(MSTR.CLASS_CODE):
98 1 1 TEMP_BK_VAL = MSTR.BOOK_VALUE - MSTR.DEPREC:
99 1 1 IF TEMP_BK_VAL > MSTR.SALVAGE_VALUE THEN MSTR.BOOK_VALUE = TEMP_BK_VAL;
100 1 1 ELSE MSTR.BOOK_VALUE = MSTR.SALVAGE_VALUE:
101 1 1 WRITE FILE(TEQMST) FRM (MSTR):
102 1 1 GO TO NEXT:
103 1 1 END:
```
FMAIN: PROC OPTIONS(MAIN):

START LEVEL NEST

116 1 IF MSTR.COUNTY_NUM < DIR.COUNTY_NUM |
       (MSTR.COUNTY_NUM = DIR.COUNTY_NUM & MSTR.EQ_NUM < DIR.EQ_NUM)
       THEN DO:
117 1  11
118 1  12
119 1  13
120 1  14
121 1  15
122 1  16
123 1  17
124 1  18
125 1  19
126 1  20
127 1  21
128 1  22
129 1  23
130 1  24

DONE:
END FMAIN;

/* IF MSTR.COUNTY_NUM > DIR.COUNTY_NUM | MSTR.EQ_NUM > DIR.EQ_NUM */

ELSE DO:

120 1  1
121 1  2
122 1  3
123 1  4
124 1  5
125 1  6
126 1  7
127 1  8
128 1  9
129 1 10
130 111

DONE:
END FMAIN;
Listing of Program MAIN

Step 3
STMT LEVEL NEST

1
1

DCL 1 MSTP,

2 County_num PIC '9999';
2 district char(2);
2 eq_num char(8);
2 class_code PIC '999';
2 year_mfd char(2);
2 mfr_code char(3);
2 engine_mfr_code char(2);
2 num_cyls char(2);
2 engine_type char(1);
2 xmission_type char(1);
2 orig_cost PIC '999999999';
2 salvage_value PIC '9999999999';
2 book_value PIC '9999999999';
2 depreciation PIC '9999999999';
2 miles_hrs_yr PIC '99999999';
2 fuel_cost_yr PIC '9999999999';
2 lubricants_yr PIC '9999999999';
2 tires_tubes_yr PIC '9999999999';
2 expn_parts_yr PIC '9999999999';
2 antifreeze_yr PIC '9999999999';
2 parts_cost_yr PIC '9999999999';
2 lmrp_cost_yr PIC '9999999999';
2 ind_cst_yr PIC '9999999999';
2 hrs_life PIC '9999999999';
2 fuel_cost_life PIC '9999999999';
2 lubricants_life PIC '9999999999';
2 tires_tubes_life PIC '9999999999';
2 expn_parts_life PIC '9999999999';
2 antifreeze_life PIC '9999999999';
2 parts_cost_life PIC '9999999999';
2 lmrp_cost_life PIC '9999999999';
2 ind_cst_life PIC '9999999999';
2 class_desc char(20);
2 make_model char(14);
2 serial_num char(14);
2 engine_desc char(14);
2 cc_purchased_pcm char(12);
2 date_purchased,
3 day char(2);
3 pay char(2);
3 yr char(2);
2 date_disposed,
3 day char(2);
3 pay char(2);
3 yr char(2);
2 disposition char(2);
2 wheelbase char(3);
2 down_time_yr PIC '999999';
2 down_time_life PIC '999999';
2 num_repairs_yr PIC '9999999';
2 num_repairs_life PIC '9999999';
2 rated_hp char(3);
2 blanks char(24);
START LEVEL 1000
3 1 DCL CNTY_NAMES(99) CHAR(13) VARYING
INITIAL,'AIR', 'AMOS', 'ALLA PARK', 'APP, NO', 'BARTON',
'BEE, ', 'BRENNER', 'BROOKE', 'BUENA VISTA', 'CUTLER',
'CALHOUN', 'CARROLL', 'CASSET', 'CEDAR', 'CERRO', 'CGCC', 'CHEYENNE',
'CHICAGO', 'CLARK', 'CLAY', 'CLAYTON', 'CLINTON', 'CLARKFOOT', 'DALLAS',
'DAVIE', 'DECATUR', 'DELAFIELD', 'DELAWARE', 'DICKINSON', 'DUBLIN',
'EAGLE', 'FAYETTE', 'FLOYD', 'FRANKLIN', 'FREMONT', 'GREENE', 'GRUNDY',
'GREEN', 'HAMILTON', 'HAZZARD', 'HAYDEN', 'HARRISON', 'HENRY',
'HUNT', 'IDA', 'IDON', 'JACOB', 'JAMES', 'JASPER', 'JEFFERSON',
'JOHNSTOWN', 'JONES', 'KEOKUK', 'KOSSE', 'LEE', 'LINN', 'LOUIS', 'LUCAS',
'LYNCH', 'MACON', 'MAHASKA', 'MARION', 'MARSHALL', 'MILLS', 'MCCULLOUGH',
'MONTANA', 'MONROE', 'MONTGOMERY', 'MUSCATING', 'MC BRIEN', 'OSCEOLA',
'PAGE', 'PARK', 'PACIFIC', 'PACIFIC', 'PACODY', 'POLL', 'PORTADIENT',
'POWELL', 'RESER', 'SAC', 'SCOTT', 'SHEPHERD', 'SIoux', 'SKY',
'TAYLOR', 'UNION', 'VAN BUREN', 'WINFIELD', 'WALPOLE', 'WARREN',
'WASHINGTON', 'WAYNE', 'WEBSTER', 'WINNEBAGO', 'WINNEBEAK', 'WOODBURY',
'WORTH', 'WRIGHT');
4 1 DCL CEPREN_H_LIFE PIC '99999999V99';
5 1 DCL CEPREN_L1 PIC '99V99';
6 1 DCL CEPREN_L2 PIC '99V99';
7 1 DCL CEPREN_L3 PIC '99V99';
8 1 DCL CEPREN_L4 PIC '99V99';
9 1 DCL CEPREN_L5 PIC '99V99';
10 1 DCL CEPREN_L6 PIC '99V99';
11 1 DCL CEPREN_L7 PIC '99V99';
12 1 DCL CEPREN_L8 PIC '99V99';
13 1 DCL CEPREN_L9 PIC '99V99';
14 1 DCL CEPREN_L10 PIC '99V99';
15 1 DCL LAST_CNTY_NUM PIC '999';
16 1 DCL LAST_CLASS PIC '999';
17 1 DCL LAST_EQ_NUM CHAR(8);
MACRO: PROC OPTIONS(MAIN):  

| ST *1 LEVEL NEST | 18 | 1 | DCL CLASS_MIL_KRS | PIC '99999999' | INIT((7) '0'); |
| 15 | 1 | DCL CLASS_FUEL | PIC '99999999' | INIT((8) '0'); |
| 20 | 1 | DCL CLASS_LUB | PIC '99999999' | INIT((6) '0'); |
| 21 | 1 | DCL CLASS_ANTIFZ | PIC '99999999' | INIT((6) '0'); |
| 22 | 1 | DCL CLASS_TIRES | PIC '99999999' | INIT((8) '0'); |
| 23 | 1 | DCL CLASS_EXPL_PARTS | PIC '99999999' | INIT((7) '0'); |
| 24 | 1 | DCL CLASS_REPAIR_PARTS | PIC '99999999' | INIT((7) '0'); |
| 25 | 1 | DCL CLASS_REPAIR_LABOR | PIC '99999999' | INIT((7) '0'); |
| 26 | 1 | DCL CLASS_TOT_DIR | PIC '99999999' | INIT((8) '0'); |
| 27 | 1 | DCL CLASS_TOT_TIME | PIC '99999999' | INIT((8) '0'); |
| 28 | 1 | DCL CLASS_HUM_REPAIRS | PIC '99999999' | INIT((8) '0'); |
| 29 | 1 | DCL CLASS_TOT_INC | PIC '99999999' | INIT((8) '0'); |
| 30 | 1 | DCL CLASS_DEPRECA | PIC '99999999' | INIT((8) '0'); |
| 31 | 1 | DCL CLASS_COST | PIC '99999999' | INIT((8) '0'); |
| 32 | 1 | DCL CLASS_DIR_CPH | PIC '99999999' | INIT((9) '0'); |
| 33 | 1 | DCL CLASS_D1_CPH | PIC '99999999' | INIT((9) '0'); |
| 34 | 1 | DCL CLASS_D2_CPH | PIC '99999999' | INIT((9) '0'); |
| 35 | 1 | DCL CLASS_M1_HRS | PIC '99999999' | INIT((9) '0'); |
| 36 | 1 | DCL CLASS_FUEL | PIC '99999999' | INIT((6) '0'); |
| 37 | 1 | DCL CLASS_LUB | PIC '99999999' | INIT((6) '0'); |
| 38 | 1 | DCL CLASS_ANTIFZ | PIC '99999999' | INIT((6) '0'); |
| 39 | 1 | DCL CLASS_TIRES | PIC '99999999' | INIT((6) '0'); |
| 40 | 1 | DCL CLASS_EXPL_PARTS | PIC '99999999' | INIT((6) '0'); |
| 41 | 1 | DCL CLASS_REPAIR_PARTS | PIC '99999999' | INIT((6) '0'); |
| 42 | 1 | DCL CLASS_REPAIR_LABOR | PIC '99999999' | INIT((6) '0'); |
| 43 | 1 | DCL CLASS_TOT_DIR | PIC '99999999' | INIT((6) '0'); |
| 44 | 1 | DCL CLASS_TOT_TIME | PIC '99999999' | INIT((6) '0'); |
| 45 | 1 | DCL CLASS_DEPRECA | PIC '99999999' | INIT((6) '0'); |
| 46 | 1 | DCL CLASS_COST | PIC '99999999' | INIT((6) '0'); |
| 47 | 1 | DCL CLASS_DIR_CPH | PIC '99999999' | INIT((6) '0'); |
| 48 | 1 | DCL CLASS_D1_CPH | PIC '99999999' | INIT((6) '0'); |
| 49 | 1 | DCL CLASS_D2_CPH | PIC '99999999' | INIT((6) '0'); |
| 50 | 1 | DCL CNTY_FUEL | PIC '99999999' | INIT((6) '0'); |
| 51 | 1 | DCL CNTY_LUB | PIC '99999999' | INIT((6) '0'); |
| 52 | 1 | DCL CNTY_ANTIFZ | PIC '99999999' | INIT((6) '0'); |
| 53 | 1 | DCL CNTY_TIRES | PIC '99999999' | INIT((6) '0'); |
| 54 | 1 | DCL CNTY_EXPL_PARTS | PIC '99999999' | INIT((8) '0'); |
| 55 | 1 | DCL CNTY_REPAIR_PARTS | PIC '99999999' | INIT((8) '0'); |
| 56 | 1 | DCL CNTY_REPAIR_LABOR | PIC '99999999' | INIT((8) '0'); |
| 57 | 1 | DCL CNTY_TOT_DIR | PIC '99999999' | INIT((8) '0'); |
| 58 | 1 | DCL CNTY_TOT_TIME | PIC '99999999' | INIT((8) '0'); |
| 59 | 1 | DCL CNTY_DEPRECA | PIC '99999999' | INIT((8) '0'); |
| 60 | 1 | DCL CNTY_TOT_INC | PIC '99999999' | INIT((8) '0'); |
| 61 | 1 | DCL CNTY_COST | PIC '99999999' | INIT((8) '0'); |
**STMT LEVEL DATA**

62  1  

**DIR_FORM:** FORMAT(CCL(3),A,CCL(15),A,CCL(16),P'ZZZZZ9',COL(39)),P'ZZZZZ9',COL(45),P'ZZZZZ9',COL(51),P'ZZZZZ9',COL(57),P'ZZZZZ9',COL(63),P'ZZZZZ9',COL(69),P'ZZZZZ9',COL(75);  

64  1  

**TOT_FORM:** FORMAT(CCL(3),A,CCL(12),A,CCL(27),P'ZZZZZ9',COL(36),P'ZZZZZ9',COL(43),P'ZZZZZ9',COL(50),P'ZZZZZ9',COL(56),P'ZZZZZ9',COL(62),P'ZZZZZ9',COL(69),P'ZZZZZ9',COL(76),P'ZZZZZ9',COL(83),P'ZZZZZ9',COL(90),P'ZZZZZ9',COL(97),P'ZZZZZ9',COL(104));  

65  1  

**GRAND_DIR:** FORMAT(CCL(38),P'ZZZZZ9',COL(48),P'ZZZZZ9',COL(54),P'ZZZZZ9',COL(60),P'ZZZZZ9',COL(66),P'ZZZZZ9',COL(73),P'ZZZZZ9',COL(80),P'ZZZZZ9',COL(86),P'ZZZZZ9',COL(93),P'ZZZZZ9',COL(100),P'ZZZZZ9',COL(107));  

66  1  

**GRAND_TOT:** FORMAT(CCL(51),P'ZZZZZ9',COL(63),P'ZZZZZ9',COL(69),P'ZZZZZ9',COL(75),P'ZZZZZ9',COL(81),P'ZZZZZ9',COL(88),P'ZZZZZ9',COL(95),P'ZZZZZ9',COL(102));  

67  1  

**DCL LAST_YEAR**  
**FIXED DEC(4);**  

68  1  

**DCL 1 TCDAYS_CATE;**  
**2 C_YEAR CHAR(2);**  
**2 C_MON_DAY CHAR(4);**  

69  1  

**TCDAYS_DATE = DATE;**  

70  1  

**LAST_YEAR = 1899 + C_YEAR;**
STAT LEVEL NEST

71 1 DCL YRDIR PRINT FILE;
72 1 DCL YRTOT PRINT FILE;
73 1 DCL LIFEDIR PRINT FILE;
74 1 DCL LIFETOT PRINT FILE;
75 1 DCL FEQMAST FILE RECORD SEQUENTIAL:
76 1 ON ENDFILE(FEQMAST) GO TO DCOME;
77 1 ON ENDPAGE(YRDIR) BEGIN;
80 2 PRINT HEADING;
81 2 END;
82 1 OPEN FILE(FEQMAST) INPUT;
83 1 OPEN FILE(YRDIR) LINESIZE(132) PAGESIZE(60);
84 1 OPEN FILE(YRTOT) LINESIZE(132) PAGESIZE(60);
85 1 OPEN FILE(LIFEDIR) LINESIZE(132) PAGESIZE(60);
86 1 OPEN FILE(LIFETOT) LINESIZE(132) PAGESIZE(60);
STMT LEVEL TEST

87 1 READ FILE(FRECAST) INTO (MSTR);
88 1 LAST_CNTY_NUM = COUNTY_NUM;
89 1 NEXTCCNTY:
   CALL HEADINGS;
90 1 NEXTCLASS:
   PUT SKIP(2) FILE(YRDIR) EDIT('EQUIPMENT CLASS - ',CLASS_DESC)
      (X(3),A*1);
91 1 PUT SKIP(2) FILE(YSTCT) EDIT('EQUIPMENT CLASS - ',CLASS_DESC)
      (X(3),A*1);
92 1 PUT SKIP(2) FILE(LIFECIR) EDIT('EQUIPMENT CLASS - ',CLASS_DESC)
      (X(3),A*1);
93 1 PUT SKIP(2) FILE(LIFETYOT) EDIT('EQUIPMENT CLASS - ',CLASS_DESC)
      (X(3),A*1);
94 1 NEXTDIR:
   TOT_DIR = FUEL_COST_YR + LUBRICANTS_YR + TIRES_TUBES_YR +
             EXPEN_PARTS_YR + ANTIFREEZE_YR + PARTS_COST_YR +
             LABC_COST_YR;
95 1 TOT_DIR_L = FUEL_COST_LIFE + LUBRICANTS_LIFE + TIRES_TUBES_LIFE +
              EXPEN_PARTS_LIFE + ANTIFREEZE_LIFE + PARTS_COST_LIFE +
              LABC_COST_LIFE;
96 1 DEPRECA_LIFE = ORIG_COST - ECCK_VALUE;
97 1 TOT_COST = TOT_DIR + IND_COST_YR + DEPRECIATION;
98 1 TOT_COST_L = TOT_DIR_L + IND_COST_LIFE + DEPREC_LIFE;
99 1 IF MILES_HRS_YR = (6)'0'
   THEN DO:
100 1   DIR_CPMH = YCT_DIR / MILES_HRS_YR;
101 1   DIR_IND_CPMH = (TOT_DIR + IND_COST_YR) / MILES_HRS_YR;
102 1   DEPR_D_I_CPMH = TOT_COST / MILES_HRS_YR;
103 1   END;
104 1 ELSE DO:
105 1   DIR_CPMH = (5)'0';
106 1   DIR_IND_CPMH = (5)'0';
107 1   DEPR_D_I_CPMH = (5)'0';
108 1   END;
109 1 IF MILES_HRS_LIFE = (6)'0'
110 1 THEN DO:
111 1   DIR_CPMH_L = TOT_DIR_L / MILES_HRS_LIFE;
112 1   DIR_IND_CPMH_L = (TOT_DIR_L + IND_COST_LIFE) / MILES_HRS_LIFE;
113 1   DEPR_D_I_CPMH_L = TOT_COST_L / MILES_HRS_LIFE;
114 1   END;
115 1 ELSE DO:
116 1   DEPR_D_I_CPMH_L = (5)'0';
117 1   DIR_IND_CPMH_L = (5)'0';
118 1   DIR_CPMH_L = (5)'0';
119 1   END;
120 1 PUT FILE(YRDIR) EDIT('EQ_NUM,MAKE_MODEL,MILES_HRS_YR,FUEL_COST_YR,
LUBRICANTS_YR,ANTI_FREEZE_YR, TIRES_TUBES_YR, EXPEN_PARTS_YR,
PARTS_COST_YR,LABC_COST_YR,TOT_DIR,DIR_CPMH)(R(DIR_FORM));
STAT LEVEL NEST

122 1 PUT FILE(LIFECYCLE) EDITEQ.NUM,MAKE_MODE, MILES_HRS_LIFE,
    FUEL_COST_LIFE,LUBRICANTS_LIFE,ANTIFREEZE_LIFE,
    TIRES_TUBES_LIFE,EXPN_PARTS_LIFE, PARTS_COST_LIFE,
    LABOR_COST_LIFE,TOT_DIR_L,DIR_CPMH_L)(DIR_FORM);

123 1 PUT FILE(YRDTOT) EDITEQ.NUM,MAKE_MODE,MILES_HRS_YR,DOWN_TIME_YR,
    NUM_REPAIRS_YR,TOT_DIR_YR,IND_CST_YR,DEPRECIATION,TOT_COST,
    DIR_CPMH,TIR_IND_CPMH,DEPR_DIR_CPMH)(YRDTOT_FORM);

124 1 PUT FILE(LIFECYCLE) EDITEQ.NUM,MAKE_MODE,MILES_HRS_LIFE,
    DWN_TIME_LIFE,NUM_REPAIRS_LIFE,TOT_DIR_L,IND_CST_LIFE,
    DEPRECIATION,LCD_COST_L,DIR_CPMH_L,DIR_IND_CPMH_L;

125 1 CLASS_MH_HRS = CLASS_MH_HRS + MILES_HRS_YR;
126 1 CLASS_MHPRS = CLASS_MHLRS.L + MILES_HRS_LIFE;
127 1 CLASS_FUEL = CLASS_FUEL + FUEL_COST_YR;
128 1 CLASS_FUEL_L = CLASS_FUEL_L + FUEL_COST_LIFE;
129 1 CLASS_LUB = CLASS_LUB + LUBRICANTS_YR;
130 1 CLASS_LUB_L = CLASS_LUB_L + LUBRICANTS_LIFE;
131 1 CLASS_ANTFZ = CLASS_ANTFZ + ANTIFREEZE_YR;
132 1 CLASSlicants = CLASSlicants + ANTIFREEZE_LIFE;
133 1 CLASS_TIRES = CLASS_TIRES + TIRES_TUBES_YR;
134 1 CLASS_TIRES_L = CLASS_TIRES_L + TIRES_TUBES_LIFE;
135 1 CLASS_EXPAN_PARTS = CLASS_EXPAN_PARTS + EXPAN_PARTS_YR;
136 1 CLASS_EXPAN_PARTS_L = CLASS_EXPAN_PARTS_L + EXPAN_PARTS_LIFE;
137 1 CLASS_REPAIR_PARTS = CLASS_REPAIR_PARTS + PARTS_COST_YR;
138 1 CLASS_REPAIR_PARTS_L = CLASS_REPAIR_PARTS_L + PARTS_COST_LIFE;
139 1 CLASS_REPAIR_LABR = CLASS_REPAIR_LABR + LABCST_YR;
140 1 CLASS_REPAIR_LABR_L = CLASS_REPAIR_LABR_L + LABCST_LIFE;
141 1 CLASS_TOT_DIR = CLASS_TOT_DIR + TOT_DIR;
142 1 CLASS_TOT_DIR_L = CLASS_TOT_CIP_L + TOT_DIR_L;
143 1 CLASS_DWN_TIME = CLASS_DWN_TIME + DWN_TIME_YR;
144 1 CLASS_DWN_TIME_L = CLASS_DWN_TIME_L + DWN_TIME_LIFE;
145 1 CLASS_NUM_REPAIRS = CLASS_NUM_REPAIRS + NUM_REPAIRS_YR;
146 1 CLASS_NUM_REPAIRS_L = CLASS_NUM_REPAIRS_L + NUM_REPAIRS_LIFE;
147 1 CLASS_TOT_IND = CLASS_TOT_IND + IND_COST_YR;
148 1 CLASS_TOT_IND_L = CLASS_TOT_IND_L + IND_COST_LIFE;
149 1 CLASS_DEPREC = CLASS_DEPREC + DEPRECIATION;
150 1 CLASS_DEPREC_L = CLASS_DEPREC + DEPRECIATION_LIFE;
151 1 CLASS_COST = CLASS_COST + TOT_COST;
152 1 CLASS_COST_L = CLASS_COST_L + TOT_COST_L;
153 1 LAST_EQ_NUM = EQ_NUM;
154 1 LAST_CLASS = CLASS_CODE;
155 1 LAST_CTY_NUM = COUNTRY_NUM;
START

LEVEL NEST

156 1 1 NEXTRECORD:
157 1 1 READ FILE(EGCHART) INTO (MSTR):
158 1 1 IF CNTY_NUM = LAST_CNTY_NUM
159 1 1 THEN GOTO:
160 1 1 CALL UPCODE:
161 1 1 CALL PCLASS:
162 1 1 PUT SKIP(2) FILE(YRDIR) EDIT ('GRAND TOTALS',CNTY_FUEL,CNTY_LUB,
163 1 1 CNTY_ANTIFZ,CNTY_TIRES,CNTY_EXPEN_PARTS,CNTY_REPAIR_PARTS,
164 1 1 CNTY_REPAIR_LABOR,CNTY_TOT_DIR) (X(6),A,R(GRANE_TOT));
165 1 1 PUT SKIP(2) FILE(YRDLC) EDIT ('GRAND TOTALS',CNTY_TOT_DIR,
166 1 1 CNTY_TOT_IND,CNTY_DEPRECN,CNTY_COST) (X(6),A,R(GRANE_TOT));
167 1 1 CLASS_MT_HRS = (7) '0';
168 1 1 CLASS_FUEL = (8) '0';
169 1 1 CLASS_LUB = (6) '0';
170 1 1 CLASS_ANTIFZ = (6) '0';
171 1 1 CLASS_TIRES = (8) '0';
172 1 1 CLASS_EXPEN_PARTS = (7) '0';
173 1 1 CLASS_REPAIR_PARTS = (7) '0';
174 1 1 CLASS_REPAIR_LABOR = (7) '0';
175 1 1 CLASS_TOT_DIR = (8) '0';
176 1 1 CLASS_DOWN_TIME = (5) '0';
177 1 1 CLASS_NUM_REPAIRS = (6) '0';
178 1 1 CLASS_TOT_IND = (8) '0';
179 1 1 CLASS_DEPRECN = (8) '0';
180 1 1 CLASS_COST = (9) '0';
181 1 1 CLASS_DIR_CPMH = (5) '0';
182 1 1 CLASS_D_O_CPMH = (5) '0';
183 1 1 CLASS_D_D_O_CPMH = (5) '0';
184 1 1 CLASS_MT_HRS_L = (7) '0';
185 1 1 CLASS_FUEL_L = (8) '0';
186 1 1 CLASS_LUB_L = (6) '0';
187 1 1 CLASS_ANTIFZ_L = (6) '0';
188 1 1 CLASS_TIRES_L = (8) '0';
189 1 1 CLASS_EXPEN_PARTS_L = (7) '0';
190 1 1 CLASS_REPAIR_PARTS_L = (7) '0';
191 1 1 CLASS_REPAIR_LABOR_L = (7) '0';
192 1 1 CLASS_TOT_DIR_L = (8) '0';
193 1 1 CLASS_DOWN_TIME_L = (5) '0';
194 1 1 CLASS_KIP_RePAIRS_L = (4) '0';
195 1 1 CLASS_TOT_IND_L = (9) '0';
196 1 1 CLASS_DEPRECN_L = (8) '0';
197 1 1 CLASS_COST_L = (9) '0';
198 1 1 CLASS_DIR_CPMH_L = (5) '0';
199 1 1 CLASS_D_O_CPMH_L = (5) '0';
200 1 1 CLASS_D_D_O_CPMH_L = (5) '0';
201 1 1 CNTY_FUEL = (9) '0';
202 1 1 CNTY_LUB = (7) '0';
203 1 1 CNTY_ANTIFZ = (6) '0';
204 1 1 CNTY_TIRES = (8) '0';
205 1 1 CNTY_EXPEN_PARTS = (4) '0';
206 1 1 CNTY_REPAIR_PARTS = (8) '0';
207 1 1 CNTY_REPAIR_LABOR = (8) '0';
208 1 1 CNTY_TOT_DIR = (9) '0';
209 1 1 CNTY_TOT_IND = (9) '0';
210 1 1 CNTY_DEPRECN = (9) '0';
SMAIN: PROC OPTIONS(MAIN);

STAT LEVEL NEST

207  1  1  CNTY_COST = (9)*D*
208  1  1  GO TO NEXTCOUNTY;
209  1  1  END;
IF LAST_CLASS = CLASS_CODE
THEN IF ED_NUM = LAST_EQ_NUM
THEN GO TO NEXTRECORD;
ELSE GO TO NEXTEQ;
ELSE:
CALL PCLASSST;
CALL UPCODE;
CLASS_HI_HRS = (7)'0';
CLASS_FUEL = (8)'0';
CLASS_LUB = (6)'0';
CLASS_ANTIFZ = (6)'0';
CLASS_TIRES = (8)'0';
CLASS_SKFV_PARTS = (7)'0';
CLASS_REPAIR_PARTS = (7)'0';
CLASS_REPAIR_LABOR = (7)'0';
CLASS_TOT_DIP = (8)'0';
CLASS_TOT_TIME = (5)'0';
CLASS_NUM_REPAIRS = (4)'0';
CLASS_TOT_INC = (8)'0';
CLASS_D_I_CPMH = (9)'0';
CLASS_COST = (9)'0';
CLASS_D_I_CPMH_L = (9)'0';
CLASS_LUB_L = (6)'0';
CLASS_ANTIFZ_L = (6)'0';
CLASS_TIRES_L = (8)'0';
CLASS_SKFV_PARTS_L = (7)'0';
CLASS_REPAIR_PARTS_L = (7)'0';
CLASS_REPAIR_LABOR_L = (7)'0';
CLASS_TOT_DIR_L = (8)'0';
CLASS_D_I_CPMH_L = (5)'0';
CLASS_D_D_I_CPMH_L = (5)'0';
CLASS_D_D_I_CPMH_L = (5)'0';
GO TO NEXTCLASS;
GO TO NEXTCLASS;
START LEVEL NEST

252  1  PRCLASS:  PRCC;

253  2  CLASS_DIR:  FORMTY(CCL(20),P'ZZZZZ9',CCL(38),P'ZZZZ9V.99',CCL(49),
                  P'ZZZZ9V.99',CCL(58),P'ZZZZ9V.99',CCL(67),P'ZZZZV.99',
                  CCL(77),P'ZZZZV.99',CCL(87),P'ZZZZV.99',CCL(97),
                  P'ZZZZ9V.99',CCL(108),P'ZZZZ9V.99',CCL(123),P'Z9V.999');

254  2  CLASS_TOT:  FORMTY(CCL(20),P'ZZZZZ9',CCL(38),P'ZZZZ9',
                  CCL(51),P'ZZZZ9V.99',CCL(64),P'ZZZZ9V.999',CCL(76),
                  P'ZZZZV.99',CCL(88),P'ZZZZV.99',CCL(101),P'Z9V.999',
                  CCL(112),P'Z9V.999',CCL(123),P'Z9V.999');

255  2  IF  CLASS_MI_HRS = (7)'0'
256  2  THEN DO;
257  2  1  CLASS_DIR_CPHM = CLASS_TOT_DIR / CLASS_MI_HRS;
258  2  1  CLASS_D_I_CPHM = (CLASS_TOT_DIR+CLASS_TOT_IND) / CLASS_MI_HRS;
259  2  1  CLASS_D_I_CPHM = CLASS_COST / CLASS_MI_HRS;
260  2  1  END;
261  2  ELSE DO;
262  2  1  CLASS_D_I_CPHM = (5)'0';
263  2  1  CLASS_D_I_CPHM = (5)'0';
264  2  1  END;

266  2  IF  CLASS_MI_HRS_L = (7)'0'
267  2  THEN DO;
268  2  1  CLASS_DIR_CPHM_L = CLASS_TOT_DIR_L / CLASS_MI_HRS_L;
269  2  1  CLASS_D_I_CPHM_L = (CLASS_TOT_DIR_L + CLASS_TOT_IND_L) / CLASS_MI_HRS_L;
270  2  1  CLASS_D_I_CPHM_L = CLASS_COST_L / CLASS_MI_HRS_L;
271  2  1  END;
272  2  ELSE DO;
273  2  1  CLASS_D_I_CPHM_L = (5)'0';
274  2  1  CLASS_D_I_CPHM_L = (5)'0';
275  2  1  END;

277  2  PUT SKIP(2) FILE(YRDIR) EDIT('CLASS TOTALS',CLASS_MI_HRS,CLASS_FUEL,
                  CLASS_LUB,CLASS_ANTIFZ,CLASS_TRES,CLASS_EXPEN_PARTS,
                  CLASS_REPAIR_PARTS,CLASS_REPAIR_LABOR,CLASS_TOT_DIR,
                  CLASS_DIR_CPHM)
                  (X(16),A,PRCLASS_CIP));

278  2  PUT SKIP(2) FILE(YRTC) EDIT('CLASS TOTALS',CLASS_MI_HRS,
                  CLASS_CCWN_TIME,CLASS_NUM_REPAIRS,CLASS_TOT_DIR,
                  CLASS_TOT_IND,CLASS_DEPENCE,CLASS_CCST,CLASS_DIR_CPHM,
                  CLASS_D_I_CPHM,CLASS_D_I_CPHM_L)
                  (X(16),A,PRCLASS_TCT));

275  2  PUT SKIP(2) FILE(LIFEDIR) EDIT('CLASS TOTALS',CLASS_MI_HRS_L,
                  CLASS_FUEL_L,CLASS_LUB_L,CLASS_ANTIFZ_L,CLASS_TRES_L,
                  CLASS_EXPEN_PARTS_L,CLASS_REPAIR_PARTS_L,
                  CLASS_REPAIR_LABOR_L,CLASS_TOT_DIR_L,CLASS_DIR_CPHM_L)
                  (X(16),A,PRCLASS_DIR));
SMAIN: PROC OPTIONS(MAIN);

START LEVEL NEST

280 2 PUT SKIP(2) FILE(LIFETOT) EDIT ('CLASS TOTALS',CLASS_MI_HRS_L,
CLASS_DC_HRS_L,CLASS_NUM_REPAIRS_L,CLASS_TOT_DIR_L,
CLASS_TOT_IND_L,CLASS_DEPRECN_L,CLASS_COST_L,
CLASS_DIR_CPMH_L,CLASS_D_I_CPMH_L,CLASS_D_D_I_CPMH_L)
(X(6),A,R(CLASS_TCT));

END PRCLASST;

PAGE 14
MAIN: PROC CPTICNS(MAIN):

PAGE 15

4. MAIN: PROC CPTICNS(MAIN):

PAGE 15

BEGIN LEVEL NEXT

282 1 HEADING:

283 2 PAGE 15

284 2 COLUMN_ STARY = INT(LENGTH(CNTY NAMES(COUNTY_NUM))/2);

285 2 PUT PAGE FILE(YORID) EDIT (CNTY NAMES(COUNTY_NUM),"COUNTY")

286 2 (COL(COLUMN_ STARY),A,A);

287 2 PUT PAGE FILE(YORID) EDIT (CNTY NAMES(COUNTY_NUM),"COUNTY")

288 2 (COL(COLUMN_ STARY),A,A);

289 2 PUT SKIP(2) FILE(YORID) EDIT (LAST_YEAR,

290 2 "EQUIPMENT DIRECT OPERATING COST$") (COL(48),F(4),A);

291 2 PUT SKIP(2) FILE(YORID) EDIT

292 2 ("LIFETIME EQUIPMENT DIRECT OPERATING COST$") (COL(46),A);

293 2 PUT SKIP FILE(YORID) EDIT (LAST_YEAR,

294 2 "EQUIPMENT TOTAL OPERATING COST$") (COL(49),F(4),A);

295 2 PUT SKIP FILE(YORID) EDIT

296 2 ("LIFETIME EQUIPMENT TOTAL OPERATING COST$") (COL(47),A);

297 2 PUT SKIP FILE(YORID) EDIT ("EQUIPMENT","MILES","'ANT"

298 2 "EXTRLE REPAIR REPAIR","DIRECT

299 2 "COST"

300 2 PUT SKIP FILE(YORID) EDIT ("EQUIPMENT","MILES","'ANT"

301 2 "EXTRLE REPAIR REPAIR","DIRECT

302 2 PUT SKIP FILE(YORID) EDIT ("EQUIPMENT","MILES","'ANT"

303 2 "EXTRLE REPAIR REPAIR","DIRECT

304 2 END HEADING;

305 2
STMT LEVEL NEST

305  1       UPCNTYI:  PROC;
306  2     CNTY_FUEL = CNTY_FUEL + CLASS_FUEL;
307  2     CNTY_LUB = CNTY_LUB + CLASS_LUB;
308  2     CNTY_ANTIFZ = CNTY_ANTIFZ + CLASS_ANTIFZ;
309  2     CNTY_TIRES = CNTY_TIRES + CLASS_TIRES;
310  2     CNTY_EXPEN_PARTS = CNTY_EXPEN_PARTS + CLASS_EXPEN_PARTS;
311  2     CNTY_REPAIR_PARTS = CNTY_REPAIR_PARTS + CLASS_REPAIR_PARTS;
312  2     CNTY_REPAIR_LABOR = CNTY_REPAIR_LABOR + CLASS_REPAIR_LABOR;
313  2     CNTY_TOT_DIR = CNTY_TOT_DIR + CLASS_TOT_DIR;
314  2     CNTY_TOT_IND = CNTY_TOT_IND + CLASS_TOT_IND;
315  2     CNTY_DEPRECN = CNTY_DEPRECN + CLASS_DEPRECN;
316  2     CNTY_COST = CNTY_COST + CLASS_COST;
317  2     END UPCNTYI;
STMT LEVEL NEST

318 1 DONE:
    CALL UPCTY:
    CALL FRCALST:

319 1
    CALL FRCALST;

320 1
    PUT SKIP(2) FILE(YRDIR) EDIT ("'GRAND TOTALS',CNTY_FUEL,CNTY_LUB,
        CNTY_ANTIFZ,CNTY_TIRES,CNTY_EXPEN_PARTS,CNTY_REPAIR_PARTS,
        CNTY_REPAIR_LABOR,CNTY_TOT_DIR)(X(6),A,R('GRAND_DIR'));

321 1
    PUT SKIP(2) FILE(YRTOT) EDIT ("'GRAND TOTALS',CNTY_TOT_DIR,
        CNTY_TOT_IND,CNTY_CEPRECN,CNTY_COST)(X(6),A,R('GRAND_TOT'));

322 1 END SMAIN:
Listing of Program CTYSUMRY
STMT LEVEL: 1

1  PROC OPTIONS(MAIN):
   / *  PROGRAM CIYSUMRY  */
   / *  THIS PROGRAM LISTS PAST YEAR AND LIFETIME CPN/CPH  */
   / *  AVERAGES FOR EACH COUNTY, DISTRICT, AND FOR THE WHOLE  */
   / *  STATE, WITHIN EACH EQUIPMENT CLASS.  */
   / *  THIS PROGRAM IS ONE OF EIGHT COMPUTER PROGRAMS WHICH FORM  */
   / *  THE "COMPUTER BASED INFORMATION SYSTEM FOR COUNTY EQUIPMENT  */
   / *  COST RECORDS".  */
   / *  WRITTEN BY  */
   / *  SYSTEMS DIVISION  */
   / *  COLLEGE OF ENGINEERING  */
   / *  THE UNIVERSITY OF IOWA  */
   / *  IOWA CITY, IOWA  */
   / *  JULY 1975  */

2  DCL UEMF FILE RECORD SEQUENTIAL:  */
3  DCL OUT FILE PRINT:  */
   / *  DECLARE THE STRUCTURE FOR THE EQUIPMENT COST RECORDS.  */
4  DCL 1 EQUIP_MSTR_FILE,  */
   2 COUNTY_NG  PIC'999',  */
   2 DISTRICT_ND  PIC'999',  */
   2 EQ_NU4  CHAR(8),  */
   2 CLASS_CODE  PIC'999',  */
   2 UNUSED_DATA_1  CHAR(12),  */
   2 MILEJS_HRS_YEAR  PIC'999999999',  */
   2 FUEL_COST_YEAR  PIC'999999999',  */
   2 LUBRICATIONS_YEAR  PIC'999999',  */
   2 TIRES_TIMES_YEAR  PIC'999999',  */
   2 EXPEN_PARTS_YEAR  PIC'999999',  */
   2 ANTIFREEZE_YEAR  PIC'999999',  */
   2 PARTS_COST_YEAR  PIC'999999999',  */
   2 LABOR_COST_YEAR  PIC'999999999',  */
   2 INDIRECT_COST_YEAR  PIC'999999999',  */
   2 MILES_HRS_LIFE  PIC'999999999',  */
   2 FUEL_COST_LIFE  PIC'999999999',  */
   2 LUBRICATIONS_LIFE  PIC'999999',  */
   2 TIRES_TIMES_LIFE  PIC'999999',  */
   2 EXPEN_PARTS_LIFE  PIC'999999',  */
   2 ANTIFREEZE_LIFE  PIC'999999',  */
   2 PARTS_COST_LIFE  PIC'999999',  */
   2 LABOR_COST_LIFE  PIC'999999999',  */
   2 INDIRECT_COST_LIFE  PIC'999999999',  */
   2 CLASS_DESC  CHAR(20),  */
   2 UNUSED_DATA_2  CHAR(24),  */
   2 FILLER  */

5  DCL CNTY_NAMES(99) CHAR(13) VARYING  */
   INIT('ADAMS' , 'ALLAMAKEE' , 'APPANOOSE' , 'AUDUBON' , 'BENTON' ,  */
   'BLACK HAWK' , 'BUTLER' , 'CASS' , 'CEDAR' , 'CERRO CORDO' , 'CHEROKEE' ,  */
   'CALHOUN' , 'CARROLL' , 'CRAWFORD' , 'DAKOTA' , 'DLOVER' , 'DES MOINES' ,  */
   'EASTERN IOWA' , 'EDDY' , 'EUROPEAN' , 'FLOYD' , 'FRANKLIN' , 'FREDERICK' ,  */
   'FREMONT' , 'GALLATIN' , 'GRANTS' , 'GREAT WESTERN' , 'GREGORY' , 'GRIMES' ,  */
   'GREER' , 'GREEN' , 'GREENWOOD' , 'GRIFFIN' , 'HAMILTON' , 'HANCOCK' ,  */
   'HAWKINS' , 'HICKORY' , 'HAMILTON' , 'HOLLOWAY' , 'HUMBOLDT' , 'HURON' ,  */
   'JACKSON' , 'JACKSONVILLE' , 'JASPER' , 'JEFFERSON' , 'JOHNSON' , 'JONES' ,  */
   'JUDEA' , 'KEOKUK' , 'KIOWA' , 'LAWRENCE' , 'LEE' , 'LEWIS & CLARK' ,  */
   'LAUREL' , 'LEWIS' , 'LUCAS' , 'MADISON' , 'MACAY' , 'MASON' , 'MEADOWS',  */
   'MERCER' , 'MERRILL' , 'METCALFE' , 'MINER' , 'MOUNT VERNON' , 'MUSKOKA',  */
   'NEAL' , 'NEWTON' , 'NORTHWESTERN IOWA' , 'O'BRIEN' , 'O'BRIEN' , 'OSSIAN',  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' ， 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' , 'PACIFIC' ,  */
   'PACIF...
STAT LEVEL NEST

*CHICKASAW* 'CLARKE'S 'CLAY'S 'CLAYTON'S 'CLINTON'S 'CRAWFORD'S 'DALLAS'S
*CAVI'S 'DEACON'T'S 'DELAWARE'S 'DES MOINES'S 'DICKINSON'S 'DOUGLAS'
*EMERY'S 'FAYETTE'S 'FLOYD'S 'FRANKLIN'S 'FRANKLYN'S 'GREEN'S 'GRANDY'
*GUTHRIE'S 'HAMMOND'S 'HANCOCK'S 'HARRINGTON'S 'HENRY'S
*HOOVER'S 'HUMBOLDT'S 'IDA'S 'IDOWA'S 'JACKSON'S 'JASPER'S 'JEFFERSON'
*JONES'S 'KANSAS'S 'KEOKUK'S 'KOSSES'TH'S 'LEE'S 'LINN'S 'LOUIS'S 'LUCAS'
*LYNET'S 'MADISON'S 'MANHATA'S 'MARIETTA'S 'MARSHALL'S 'MILLS'S 'MITCHELL'S
*MONTANA'S 'MOYER'S 'MONTGOMERIES 'MUSCATINE'S 'CHURCH'S 'OSCEOLA'S
*PAGES'S 'PALO ALTO'S 'PLATYOUTH'S 'PROMENTAS'S 'POLK'S 'POTASSIAMIE'S
*POWESHEIK'S 'RINGGOLD'S 'SCOTT'S 'SHELBYS 'SIGUYS 'STORY'S
*TAMA'S 'TAYLOR'S 'UNION'S 'VAN BURENS 'WAPELLO'S 'WARRENS
*WASHINGTON'S 'WAYNE'S 'WEBSTER'S 'WINNEBAGO'S 'WINNIESHEIK'S 'WOODBURY'S
*WORTH'S 'WRIGHT'S:

6 1
DCL (DIP_COST_YR,DIP_COST_LIFE) FIXED DEC (9) INIT (0):
7 1
DCL (TOT_COST_CTY_YR,CTY_DIP_COST_CTY_LIFE) FIXED DEC (9)
INIT (0):
8 1
DCL (DIST_COST_YR,DIST_COST_LIFE) FIXED DEC (9) INIT (0):
9 1
DCL (STATE_COST_YR,STATE_COST_LIFE) FIXED DEC (10) INIT (0):
10 1
DCL (M_CTRY_YR,M_CTRY_LIFE) FIXED DEC (9) INIT (0):
11 1
DCL (M_CTRY_YR,M_CTRY_LIFE) FIXED DEC (9) INIT (0):
12 1
DCL (VTR_CTRY_YR,VTR_CTRY_LIFE) FIXED DEC (9) INIT (0):
13 1
DCL (CTY_CP_YR,CTY_CP_LIFE,CTY_PAST_YR,DIST_CP_YR,DIST_CP_LIFE,ST_CP_YR,
ST_CP_LIFE) FIXED DEC (6,3) INIT (0):
14 1
DCL (COUNTY,YEAR) FIXED DEC (3):
15 1
DCL (CLO_DISTRICT, CLO_CLASS) FIXED DEC (2):
16 1
DCL (CLO_EQUIP_CLASS) FIXED DEC (2):
17 1
DCL 1 TODAY'S_DATE, C_YEAR, C_MONTH, DAY CHAR(2),
18 1
DCL C_Y FIXED DEC (2):
19 1
DCL PAST_YEAR FIXED DEC (4):
20 1
TODAYS_DATE = DATE:
21 1
C_Y = C_YEAR:
22 1
PAST_YEAR = 1900 + C_Y - 1:
23 1
ON ENDFILE (UEMF) GO TO DONE:
24 1
ON ENDPAGE (OUT) BEGIN:
25 2
PLT FILE (OUT) PAGE:
26 2
PUT FILE (OUT) SKIP (2) EDIT ('DIRECT OPERATING COSTS') (COL(56), A):
27 2
PUT FILE (OUT) SKIP (2) EDIT ('COST PER MILE OR HOUR BY EQUIPMENT')
(COL(56), A):
28 2
PUT FILE (OUT) SKIP (2) EDIT ('COUNTY', 'PAST_YEAR', 'LIFE', 'PIECES OF')
(COL(23), A, COL(43), F(4), COL(53), A, COL(60), A):
29 2
PUT FILE (OUT) SKIP (2) EDIT ('CODE', 'NAME', 'CPH/CPH', 'CPH/CPH',
'EQUIPMENT')
(COL(20), A, COL(28), A, COL(41), A, COL(51), A, COL(60), A):
30 2
END:
31 1
OPEN FILE (OUT) PAGESIZE (60) LINESIZE (132) OUTPUT:
32 1
OPEN FILE (UEMF) INPUT:
36 1 SIGNAL ENDPAGE(CUT*);
37 1 REPEAT: READ FILE(U5WF) INTO EQUIP_MSTR_FILE;
38 1 IF EQUIP_MSTR_FILE..CLASS_CODE = 0
39 1 THEN GOTO REPEAT;
40 1 PUT FILE(CUT) SKIP(2) EDIT('EQUIPMENT' CLASS = 'CLASS DESC') 
41 1 (C1L(11), A, (20));
42 1 PUT FILE(CUT) SKIP(2) EDIT('DISTRICT ' DISTRICT_NO') 
43 1 (C1L(14), A, (2));
44 1 LOC1: M_H_CTV_YR = M_H_CTV_YR + MILES_HRS_YEAR;
45 1 M_H_CTV_LIFE = M_H_CTV_LIFE + MILES_HRS_LIFE;
46 1 DIR_COST_YR = FUEL_COST_YEAR + LUBRICANTS_YEAR + TIRES_TUBES_YEAR + 
47 1 EXPENSE_PARTS_YEAR + ANTIFREEZE_YEAR + PARTS_COST_YEAR + 
48 1 LABOR_COST_YEAR;
49 1 DIR_COST_LIFE = FUEL_COST_LIFE + LUBRICANTS_LIFE + TIRES_TUBES_LIFE + 
50 1 EXPENSE_PARTS_LIFE + ANTIFREEZE_LIFE + PARTS_COST_LIFE + 
51 1 LABOR COST_LIFE;
52 1 TCT_DIR_COST_CTY_YR = TCT_DIR COST_CTY_YR + DIR_COST_YR;
53 1 TCT_DIR_COST_CTY_LIFE = TCT_DIR COST_CTY_LIFE + DIR_COST_LIFE;
54 1 VEH_CTY = VEH_CTY + 1;
55 1 OLC_COUNTY = COUNTY_NO;
56 1 OLC_EQUIP_CLASS = CLASS_CODE;
57 1 READ FILE(U5WF) INTO EQUIP_MSTR_FILE;
58 1 IF (OLC_COUNTY = COUNTY_NO) | 
59 1 (OLC_EQUIP_CLASS < CLASS_CODE)
60 1 THEN DO:
61 1 1 IF M_H_CTV_YR = 0
62 1 1 1 THEN CTV_CPM_YR = TCT_DIR COST_CTY_YR/M_H_CTV_YR;
63 1 1 1 ELSE CTV_CPM_YR = 0;
64 1 1 1 IF M_H_CTV_LIFE = 0
65 1 1 1 THEN CTV_CPM_LIFE = TCT_DIR COST_CTY_LIFE/M_H_CTV_LIFE;
66 1 1 1 ELSE CTV CPM LIF E = 0;
67 1 1 1 DIST_COST_YR = DIST COST_YR + TCT_DIR COST_CTY_YR;
68 1 1 1 DIST_COST_LIFE = DIST COST_LIFE + TCT_DIR COST_CTY_LIFE;
69 1 1 1 TCT_DIR COST_CTY_YR = TCT_DIR COST CTV_LIFE + 0;
70 1 1 1 M_HDIST_YR = M_H DIST_YR + M H CTV_YR;
71 1 1 1 M_HDIST LIF E = M H DIST_LIFE + M H CTV_LIFE;
72 1 1 1 VEH_CTY = VEH_CTY + 1;
73 1 1 1 PUT FILE(CUT) SKIP EDIT(CITY, CPM_YR, CITY, CPM_LIFE, VEH_CTY) 
74 1 1 1 (C(20), F(3), C(25), A, C(42), F(6, 3, -2), C(52), F(6, 3, -2), 
75 1 1 1 C(63), F(2));
76 1 1 1 VEH_CTY = 0;
77 1 1 1 END;
78 1 1 IF (OLC_DISTRICT = DISTRICT_NO) | (OLC_EQUIP_CLASS < CLASS_CODE)
79 1 1 THEN DO:
80 1 1 1 IF M_H DIST_YR = 0
81 1 1 1 THEN DIST_CPM_YR = DIST COST_YR/M_H DIST_YR:
ELSE DIST_CPW_YR = 0;
76 1 1 IF M_H_DIST_LIFE = 0 THEN DIST_CPW_LIFE = DIST_COST_LIFE/M_H_DIST_LIFE;
77 1 1 ELSE DIST_CPW_LIFE = 0;
78 1 1 STATE_COST_YR = STATE_COST_YR + DIST_COST_YR;
79 1 1 STATE_COST_LIFE = STATE_COST_LIFE + DIST_COST_LIFE;
80 1 1 DIST_COST_YR,DIST_COST_LIFE = 0;
81 1 1 M_H_ST_YR = M_H_ST_YR + M_H_DIST_YR;
82 1 1 M_H_ST_LIFE = M_H_ST_LIFE + M_H_DIST_LIFE;
83 1 1 M_H_DIST_YR,M_H_DIST_LIFE = 0;
84 1 1 VEH_STATE = VEH_STATE + VEH_DIST;
85 1 1 PUT FILE(OUT) SKIP EDIT('AVERAGE FOR DISTRICT ',OLD_DISTRICT,
86 1 1 DIST_CPW_YR,DIST_CPW_LIFE,VEH_DIST)
87 1 1 (COL(17),A,F(2),COL(42),F(63,-2),CCL(52),F(63,-2),COL(62),
88 1 1 F(3));
89 1 1 IF OLD_EQUIP_CLASS = CLASS_CODE
90 1 1 THEN PUT FILE(OUT) SKIP(2) EDIT('DISTRICT ',DISTRICT_NO)
91 1 1 (COL(14),A,F(2));
92 1 1 END;
93 1 1 IF COL_EQUIP_CLASS < CLASS_CODE
94 1 1 THEN DO;
95 1 1 IF M_H_ST_YR = 0 THEN ST_CPW_YR = STATE_COST_YR/M_H_ST_YR;
96 1 1 ELSE ST_CPW_YR = 0;
97 1 1 IF M_H_ST_LIFE = 0 THEN ST_CPW_LIFE = STATE_COST_LIFE/M_H_ST_LIFE;
98 1 1 ELSE ST_CPW_LIFE = 0;
99 1 1 STATE_COST_YR,STATE_COST_LIFE = 0;
100 1 1 M_H_ST_YR,M_H_ST_LIFE = 0;
101 1 1 PUT FILE(OUT) SKIP(2) EDIT('AVERAGE COST FOR STATE',ST_CPW_YR,
102 1 1 ST_CPW_LIFE,VEH_STATE) (COL(14),A,COL(42),F(63,-2),CCL(52),
103 1 1 F(63,-2),COL(61),F(4));
104 1 1 VEH_STATE = 0;
105 1 1 PUT FILE(OUT) SKIP(2) EDIT('EQUIPMENT CLASS ',CLASS_CODE)
106 1 1 (COL(14),A,F(2));
107 1 1 END;
108 1 1 GC TO LOOP;
109 1 1 DONE: IF M_H_CTY_YR = 0 THEN CTY_CPW_YR = TOT_DIR_COST_CTY_YR/M_H_CTY_YR;
110 1 1 ELSE CTY_CPW_YR = 0;
111 1 1 IF M_H_CTY_LIFE = 0 THEN CTY_CPW_LIFE = TOT_DIR_COST_CTY_LIFE/M_H_CTY_LIFE;
112 1 1 ELSE CTY_CPW_LIFE = 0;
113 1 1 DIST_COST_YR = DIST_COST_YR + TOT_DIR_COST_CTY_YR;
114 1 1 DIST_COST_LIFE = DIST_COST_LIFE + TOT_DIR_COST_CTY_LIFE;
115 1 1 M_H_CTY_YR = M_H_CTY_YR + M_H_CTY_TYR;
116 1 1 M_H_CTY_LIFE = M_H_CTY_LIFE + M_H_CTY_LIFE;
117 1 1 VEH_CTY = VEH_DIST + VEH_CTY;
118 1 1 IF M_H_DIST_YR = 0
S-W LEVEL NEST

119  1  THEN DIST_CPM_YR = DIST_COST_YR/M_H_DIST_YR;
120  1  ELSE DIST_CPM_YR = 0;
121  1  IF M-H_DIST_LIFE = 0;
122  1  THEN DIST_CPM_LIFE = DIST_COST_LIFE/M_H_DIST_LIFE;
123  1  ELSE DIST_CPM_LIFE = 0;
124  1  STATE_COST_YR = STATE_COST_YR + DIST_COST_YR;
125  1  STATE_COST_LIFE = STATE_COST_LIFE + DIST_COST_LIFE;
126  1  M-H_ST_YR = M-H_ST_YR + M-H_DIST_YR;
127  1  M-H_ST_LIFE = M-H_ST_LIFE + M-H_DIST_LIFE;
128  1  VEH_STATE = VEH_STATE + VEH_DIST;
129  1  IF M-H_ST_YR = 0;
130  1  THEN ST_CPM_YR = STATE_COST_YR/M-H_ST_YR;
131  1  ELSE ST_CPM_YR = 0;
132  1  IF M-H_ST_LIFE = 0;
133  1  THEN ST_CPM_LIFE = STATE_COST_LIFE/M-H_ST_LIFE;
134  1  ELSE ST_CPM_LIFE = 0;
135  1  PUT FILE (OUT) SKIP EDIT (CLD_COUNTY, CNTY_NAMES (CLD_COUNTY),
136  1  CTV_CPM_YR, CTV_CPM_LIFE, VEH_CTY) (COL (24), F (3), COL (25), A,
137  1  COL (42), F (6, 3, -2), COL (52), F (6, 3, -2), COL (63), F (2));
138  1  END ECCLSUM;

D-138
Listing of Program MFGAGE
/*FRAGE: PROC OPTIONS(MAIN);*/

PROC OPTIONS(MAIN):

/*
** PROGRAM OPTIONS
*/

/
** THIS PROGRAM LISTE LAST YEAR AND LIFETIME CPM/CPH */
** AVERAGES FOR VARIOUS EQUIPMENT AGE GROUPINGS BY EQUIPMENT */
** MANUFACTURERS WITHIN EACH EQUIPMENT CLASS. */
** THIS PROGRAM IS ONE OF EIGHT COMPUTER PROGRAMS WHICH FORM */
** THE "COMPUTER BASED INFORMATION SYSTEM FOR COUNTY EQUIPMENT */
** COST RECORDS". */
**
** WRITTEN BY */
** SYSTEMS DIVISION */
** COLLEGE OF ENGINEERING */
** THE UNIVERSITY OF IOWA */
** IOWA CITY, IOWA */
** JULY 1975 */

DCL UEMP FILE RECORD SEQUENTIAL;
DCL MFR FILE;
DCL OUT FILE PRINT;

/*
** DECLARE THE STRUCTURE FOR THE EQUIPMENT COST RECORDS. */

DCL 1 EQUIP_MSTR_FILE;
  2 COUNTY_NO PIC '999',
  2 DISTRICT_NO PIC '99',
  2 EQ_NUM PIC '999',
  2 CLASS_CODE CHAR(8),
  2 MFR_CODE PIC '9999',
  2 MILES_HRS_YEAR PIC '99999999',
  2 FUEL_COST_YEAR PIC '99999999',
  2 LUBRICANTS_YEAR PIC '99999999',
  2 TIRES_TUBES_YEAR PIC '99999999',
  2 EXPENSE_PARTS_YEAR PIC '99999999',
  2 ANTI_FREEZE_YEAR PIC '99999999',
  2 PARTS_COST_YEAR PIC '99999999',
  2 LADO_COST_YEAR PIC '99999999',
  2 INDIRECT_COST_YEAR PIC '99999999',
  2 MILES_HRS_LIFE PIC '99999999',
  2 FUEL_COST_LIFE PIC '99999999',
  2 LUBRICANTS_LIFE PIC '99999999',
  2 TIRES_TUBES_LIFE PIC '99999999',
  2 EXPENSE_PARTS_LIFE PIC '99999999',
  2 ANTI_FREEZE_LIFE PIC '99999999',
  2 PARTS_COST_LIFE PIC '99999999',
  2 LADO_COST_LIFE PIC '99999999',
  2 INDIRECT_COST_LIFE PIC '99999999',
  2 CLASS_DESC CHAR(20),
  2 UNUSED_DATA_2 CHAR(84),
  2 FILLER CHAR(261);

DCL ARRAY_ELEM FIXED DEC(3);
**START LEVEL NEST**

```
DCL NAMES CHA(23);
DCL NO_MFR FIXED DEC(3) INIT(20);
DCL MFR_NAMES(*) CHAR(23) VARYING CONTROLLER;
DCL MFR_VEH(5,3) FIXED DEC(10) INIT(15) 01;
DCL C_INFL(5,3) FIXED DEC(10) INIT(10) 01;
DCL TOTALS(5,2) FIXED DEC(4,3) INIT(10) 01;
DCL CLO_EQUP_CLASS FIXED CEC(2);
DCL MFR_CODE FIXED DEC(3);
DCL AGE(5) CHAR(7) VARYING
   INIT(0,2,"4-6","7-9","10-12","OVER 12");
DCL TODAYS_DATE CHAR(6);  
DCL C_YEAR CHAR(2) DEFINE TODAYS_DATE POS(1);
DCL PAST_YEAR FIXED DEC(4);
TODAYS_DATE = DATE;
PAST_YEAR = 1900 + C_YEAR - 1;
OPEN FILE(UINF) INPUT;
OPEN FILE(MFR) INPUT;
OPEN FILE(IN) PAGESIZE(60) LINESIZE(132) OUTPUT;
ON ENDFILE(MFR) GO TO NEXT;
ON ENDFILE(UINF) GO TO DCHE;
GET FILE(MFR) EDIT(NO_MFR) (COL(1),F(3));
ALLOCATE MFR_NAMES(0:NO_MFR);
ON ENDPAGE(OUT) BEGIN;
   PUT FILE(OUT) PAGE;
   PUT FILE(OUT) SKIP(2) EDIT('DIRECT OPERATING COSTS')
      (COL(56),A);
   PUT FILE(OUT) SKIP(2) EDIT('COST PER MILE OR HOUR BY,
      'EQUIPMENT CLASS, MANUFACTURER', AND, AGE')
      (COL(34),A);
   PUT FILE(OUT) SKIP(2) EDIT('MANUFACTURER',PAST_YEAR, 'LIFE')
      (COL(14),A,COL(35),A,COL(44),A,COL(45),A);
   PUT FILE(OUT) SKIP EDIT('AGE','CPH/CMP','CPM/CMH','NUMBER')
      (COL(21),A,COL(33),A,COL(43),A,COL(53),A);
END;
SIGNAL ENDPAGE(OUT);
/ * READ IN A LIST OF THE MANUFACTURER NAMES. */
HERE: GET FILE(MFR) EDIT(ARRAY_ELEM, NAMES)
      (COL(1),F(3),A(23));
MFR_NAMES(ARRAY_ELEM) = NAMES;
GO TO HERE;
/ * READ A RECORD FROM THE MASTER FILE. */
NEXT: READ FILE(UINF) INTO (EQUIP_MSTR_FILE);
IF CLASS_CODE = 0
   THEN GO TO NEXT;
   PUT FILE(OUT) SKIP(2) EDIT('EQUIPMENT CLASS = ',CLASS_DESC)
```
STMT LEVEL SGEST

(CCL(11), A, A(20));

LOOP: IF C_YEAR - YR_MFR <= 3
  THEN I = 1;
ELSE IF C_YEAR - YR_MFR <= 6
  THEN I = 2;
ELSE IF C_YEAR - YR_MFR <= 9
  THEN I = 3;
ELSE IF C_YEAR - YR_MFR <= 12
  THEN I = 4;
ELSE I = 5;

MI_FR_VEH(I, 3) = MI_FR_VEH(I, 3) + 1;
MI_HR_VEH(I, 1) = MI_HR_VEH(I, 1) + MILES_HRS_YEAR;
MI_HR_VEH(I, 2) = MI_HR_VEH(I, 2) + MILES_HRS_LIFE;

COSTS_YR_LIFE(I, 1) = COSTS_YR_LIFE(I, 1) + FUEL_COST_YEAR +
LUBRICANTS_YEAR + TIRES_TUBES_YEAR + EXPEN_PARTS_YEAR +
ANTIFREEZE_YEAR + PARTS_COST_YEAR + LABOR_COST_YEAR;
COSTS_YR_LIFE(I, 2) = COSTS_YR_LIFE(I, 2) + FUEL_COST_LIFE +
LUBRICANTS_LIFE + TIRES_TUBES_LIFE + EXPEN_PARTS_LIFE +
ANTIFREEZE_LIFE + PARTS_COST_LIFE + LABOR_COST_LIFE;

OLD_EQUIP_CLASS = CLASS_CODE;
OLD_MFR_CODE = MFR_CODE;

/* READ A RECORD FROM THE MASTER FILE. */

READ FILE(USEMF) INTO (EQUIP_MSTR_FILE);

IF(CLO_MFR_CODE = MFR_CODE) I
  (OLD_EQUIP_CLASS < CLASS_CODE)
  THEN DO:
    CC I = 1 TO 5;
    DC J = 1 TO 2;
    IF MI_HR_VEH(I, J) = 0
      THEN TOTALS(I, J) = COSTS_YR_LIFE(I, J)/MI_HR_VEH(I, J);
    ELSE TOTALS(I, J) = 0;
    END;
EN:
END;

PUT FILE(OUT) SKIP(2) EDIT(MFR_NAMES(OLD_MFR_CODE))
  (CCL(14), A);
DO I = 1 TO 5;
IF MI_FR_VEH(I, 3) > 0
  THEN PUT FILE(CUT) EDIT(AGE(I),
    (TOTALS(I, J) DC J = 1 TO 2), MI_FR_VEH(I, 3))
    (COL(23), A, COL(34), F(6, 3), COL(44), F(6, 3), COL(53), F(41));
END:
END;

MI_FR_VEH = 0;
COSTS_YR_LIFE = 0;
END:

IF OLD_EQUIP_CLASS < CLASS_CODE
THEN DO:
MFRAGE: PROC OPTIONS(MAIN):

START LEVEL NEST

83 11 PUT FILE(OUT) SKIP(2) EDIT('EQUIPMENT CLASS = ', CLASS_DESC(COL(11),A,A(20))):
84 11 END:
85 11 GO TO LOOP:
86 11 DONE: DO I = 1 TO 5:
87 11 DO J = 1 TO 2:
88 11 IF MI_HR_VEH(I,J) = 0
89 11 THEN TOTALS(I,J) = COSTS_YR_LIFE(I,J)/MI_HR_VEH(I,J);
90 11 ELSE TOTALS(I,J) = 0;
91 11 ENC:
92 11 ENC:
93 11 PUT FILE(OUT) SKIP(2) EDIT(MFR_NAMES(Old_MFR_CCDE))
94 11 (COL(14),A):
95 11 DO I = 1 TO 5:
96 11 IF MI_HR_VEH(I,3) > 0
97 11 THEN PUT FILE(OUT) SKIP (AGE(I),
98 11 (Y)TOTALS(I,J) DC J = 1 TO 2,MI_HR_VEH(I,3))
99 11 (COL(23),A,COL(34),F(6,3),COL(44),F(6,3),COL(53),F(4));

END MFRAGE:
Listing of Program UPDATE
DECLARE OLD_MST FILE RECORD SEQUENTIAL;
   1 UPDATED_EQ_MSTP_FILE,
      2 COUNTY_NUMBER PIC 99999.
      2 DISTRICT_NUMBER CHAR(2),
      2 EQUIPMENT_NUMBER CHAR(8),
      2 CLASS_CODE CHAR(2),
      2 OTHER_INFO1 CHAR(93),
      2 LIFETIME PIC '9999999',
      2 FUEL_COST PIC '9999999',
      2 LUB_COST PIC '9999999',
      2 TIRES_YUBES_COST PIC '9999999',
      2 EXP_PRTS_COST PIC '9999999',
      2 ANTI_FREEZE_COST PIC '999999',
      2 PART_COST PIC '9999999',
      2 LABOR_COST PIC '9999999',
      2 INDIRECT_COST PIC '9999999',
      2 CLASS_DESCRIPTION CHAR(20),
      2 MAKE_MODEL_DESCRIPTION CHAR(14),
      2 OTHER_INFO2 CHAR(46),
      2 DATE_SOLD,
         3 MCNTH CHAR(2),
         3 DAY CHAR(2),
         3 YR CHAR(2),
      2 DISPOSAL_METHOD CHAR(14),
      2 OTHER_INFO3 CHAR(43);
**STATE LEVEL NEST**

*CAVIS*, *DECATUR*, *DELAWARE*, *DFS MUNES*, *DICKINSON*, *DLRQUE*,
*EMMET*, *FAYETTE*, *FLOYD*, *FRANKLIN*, *FREEMONT*, *GREENE*, *GRUNDY*,
*GUTHRIE*, *HAMILTON*, *HANCOCK*, *HARDDIN*, *HAFRISON*, *HARL*,
*HAYWOOD*, *HUMBROL*, *IDA*, *IDAHO*, *JACKSON*, *JASPER*, *JEFFERSON*,
*JEFFERSON*, *JENES*, *KOKUK*, *KOSKUT*, *LEC*, *LINN*, *LOUISA*, *LUCAS*,
*LICK*, *LINDON*, *LICK*, *MADISON*, *MAHASKA*, *MARION*, *MARMON*, *MARIAN*, *MILLS*, *MICH*,
*MORGAN*, *MINCE*, *MONTGOMERY*, *MUSCATINE*, *COLUMBIA*, *DANIEL*, *DINGMAN*,
*PASSEY*, *PLAT ALT*, *PLUMCOUT*, *POCAHONTAS*, *POLK*, *PUITTA*, *ATTAMIE*,
*PKESHEK*, *RINGGOLD*, *SAC*, *SCOTT*, *SHELBY*, *SIDWAY*, *STORY*,
*TAYLOR*, *TAYLOR*, *TAYLOR*, *VAN BUREN*, *WAPHEL*, *WARREN*,
*WASHINGTON*, *WAYNE*, *WEBSTER*, *WINNEBAGO*, *WINNESHIEK*, *WOODBURY*,
*WORTH*, *WRIGHT*;

8 1  DECLARE DISP_METH CHAR(6) VAR;
9 1 DECLARE NEW_COUNTY PIC'999' INIT(0);
10 1 OPEN FILE(OLDMST) INPUT;
11 1 OPEN FILE(NEWMST) OUTPUT;
12 1 ON ENDFILE(OLDMST) GO TO DONE:
/* READ A RECORD FROM THE MASTER FILE, AND CHECK TO SEE IF IT */
/* IS FLAGGED FOR DELETION. */

14 1 NEXT: READ FILE(OLDST) INTO (UPDATED_EQ_MSTR_FILE);
15 1 IF DISPOSAL_METHOD = 'T' THEN DISP_METH = 'TRADED';
16 1 ELSE IF DISPOSAL_METHOD = 'J' THEN DISP_METH = 'JUNKED';
17 1 ELSE DISP_METH = 'SOLD';
18 1 IF NEW_COUNTRY ^= COUNTY_NUMBER THEN DO:
19 2 LETTERS = LENGTH(COUNTY_NAME(COUNTY_NUMBER));
20 2 PUT PAGE:
21 2 PUT SKIP EDIT(COUNTY_NAME(COUNTY_NUMBER), * COUNTY)
22 2 (CPL((125-LETTERS)/2),A); PUT SKIP(2) EDIT('LIST OF EQUIPMENT DISPOSED OF IN 19*,YEAR)
23 2 (CPL(47),A); PUT SKIP(2) EDIT('TOTAL') (COL(30),A);
24 2 PUT SKIP EDIT(* EQUIPMENT MILES/DIRECT *'
25 2 'DISPOSAL DISPOSAL','LIFE') (A,A,X(6),A);
26 2 PUT SKIP EDIT('NUMBER MAKE & MODEL HOURS COSTS *'
27 2 'DATE METHOD','CPH/CPH') (A,A,X(5),A);
28 2 PUT SKIP(1):
29 2 NEW_COUNTRY = COUNTY_NUMBER;
30 2 END:
31 1 TOTAL_DIRECT_COST = FUEL_COST + LUB_COST + TIPES_TUBES_COST +
32 1 EXP_PARTS_COST + ANTIFREEZE_COST + PART_COST + LABOR_COST;
33 1 IF LIFE_M_H ^= 0 THEN CPM = TOTAL_DIRECT_COST/(100*LIFE_M_H);
34 1 ELSE CPM = 0;
35 1 PUT SKIP EDIT(EQUIPMENT_NUMBER,MAKE_MODEL_DESCRIPTION,LIFE_M_H,
36 1 TOTAL_DIRECT_COST,MONTH,DAY,YEAR,DISP_METH,CPM)
37 1 (CPL(3),A,X(2),A,X(2),F(7,0),F(2),F(9,2),2,X(13),A,A,A,A,A,
38 1 X(7),A,X(6),F(6,3));
39 1 GO TO NEXT;
40 1 DONE: END UMF;