Spring 5-1-2004

Defense Finance & Accounting Service Suspense Account Reconciliation

Linda Campbell
Dakota State University

Defense Finance & Accounting Service

Suspense Account Reconciliation

Linda Campbell

A project submitted in partial fulfillment of the requirements for the

Master of Science in Information Systems, Dakota State University

2004

DAKOTA STATE dsu UNIVERSITY

MSIS
PROJECT APPROVAL FORM

Student Name: Linda Campbell

Expected Graduation Date: May 2005

Master's Project Title: Defense Finance and Accounting Service Suspense Account Reconciliation

Date Project Plan Approved: _____

Date Project Coordinator Notified and Grade Submitted: _____

Approvals/Signatures:

Student: Linda Campbell Date: Dec 2, 2004

Faculty supervisor: [signature] Date: 12/4/04

Committee member: [signature] Date: 12/4/04

Committee member: [signature] Date: 12/6/04

To Ensure Certification of Completion:
Student must bring or send the original to the Graduate Programs Office.
Copies on acid free paper go to the library with the reports for binding.

Original to Graduate Programs Office
Acid-free copies with written reports to library
Copies to: Project supervisor and committee and to MSIS Coordinator

Revised 1/28/03

Abstract

Defense Finance & Accounting Services (DFAS) Indianapolis is required to work a Suspense Account Reconciliation (SAR). For this process, each individual Army installation, Activity, or State National Guard Headquarters submits an Excel workbook to the installation's accounting center. The assigned accounting center then rolls up the worksheets into one submission to Headquarters.

The Excel workbook submitted by each installation is designed as a template in report format. Each workbook has a detail sheet and a summary sheet. The detail sheet has 79 rows (fund categories) and 63 columns (aging) of data. The data reported is a breakdown of Absolute value, Net Value, and transaction count by fund and age.

The existing SAR process took one person from 3 to 5 days monthly and used an Excel macro to compile the numerous submissions into one file. Once all of the data was rolled up, it was necessary to "cut-and-paste" data into other specific report formats.

The section working this task requested an improved method to complete the process quicker and allow data collected to remain available for other computations. The resulting project allowed for a more time-efficient and user-friendly process. All data collected remains in tact and available for on-the-spot data calls.

This is a new application that mimics results previously submitted to the higher headquarters. The application is a form driven executable database application loaded on the individual users desktop.

Table of Contents

List of Figures

Introduction

Defense Finance & Accounting Services-Indianapolis (DFAS) currently works a Suspense Account Reconciliation (or SAR) as required by Higher Headquarters. For this process, each individual Army installation, National Guard Camp, or other Army Activity submits a two-sheet Excel workbook in template form as defined by Arlington (Headquarters) to the installation's next higher accounting center (DFAS-Indianapolis). The Excel workbook is designed as a template in more of a report format than traditional spreadsheet format. Each workbook has two worksheets, the "Summary" tab is a summary sheet and the "Suspense_Template" tab is a detail sheet. This Excel format contains 79 lines (fund categories) and 61 columns reporting Absolute Totals, Net Totals, Overall Transaction Counts, Debit Totals, Debit Counts, Credit Totals, and Credit Counts in each of 7 aging categories. During any given month, as many as 150 field sites will submit an Excel file to Indianapolis. Indianapolis must then roll up all field submitted Excel files into one file for submission to Arlington according to the pre-defined template structure.

In addition, some of the information in the spreadsheets is also needed for local requirements outside of the original design. The new, local requirements include different presentations or summaries of some of the same data.

The current process is very time and labor intensive, sometimes taking as much as 5 days to complete the process. It is also sometimes error-prone due to the need to re-key

information to produce numerous reports from the same source data. The section working this task has requested an improved method to complete this process quicker and allow the data collected to be used for other computations.

This project allows for a more time-efficient and user-friendly process to accomplish the roll-up for higher headquarters while maintaining the data collected in a format that can easily be adapted for other uses eliminating the need for re-keying information into various report formats. All data collected will remain in tact and available for on-the-spot data calls.

This report will discuss the various aspects of the project. It will review the situation prior to the implementation of this project, what the proposed plan of action was, a list of electronic output reports produced, an in-depth description of the application developed to resolve the problem as defined, issues encountered during development and implementation, a walk-through of the new process, a Work Breakdown Schedule and Gantt Chart, table structures, program code, and user interface designs and underlying code.

Statement of Problem

The initial requirement for this project was simply to solve a need for new reporting requirements against data already collected. The figures submitted by the field installations now needed to be summarized according to the installation's Major Command (MACOM), however, the field site submissions did not contain the Major Command Code. This would have to be obtained from a master table on the DFAS system and then assigned to the Fiscal Station Number (FSN) representing the individual installation or activity. In its current form, a technician would have to re-key each of these spreadsheets, along with the associated MACOM into something else allowing the figures to be summarized against the MACOM.

When researching the current process to offer a solution to the request at hand, other issues such as the amount of time this process actually took and the probability of errors due to re-keying were identified as problematic.

The summations currently prepared did not lend themselves easily to other uses. Under the current method, a technician would need to go through each of the spreadsheets manually to find the pieces needed to compute the other report requirements.

Method Prior To Project

This Suspense Account Reconciliation (SAR) process in its current form took one person any where from 3 to 5 days monthly.

As each file was received from the submitting installation, an accounting technician would inspect the file to ensure the data was within the guidelines provided. Guidelines provided include items such as Absolute values are never more than Net, Absolute values are never negative, Count can not be more than zero if Absolute is zero, and specific fund categories can not have aged lines over 60 days. Technicians must also ensure that the field site did not zero out a field by replacing it with a blank character space. A blank character space as a field value would cause an error in the spreadsheet formula expecting a numeric value. When data validation was completed, each file would be named according to the FSN that submitted the file, the Operating Location (OPLOC) that the FSN reported to, and the month/year the file was submitted for. It was then saved in the appropriate version of Excel.

With data validation complete, the technician was now ready to start the compilation process. To accomplish this, the higher headquarters had designed an Excel macro to bring all of these sheets into one book and then total the pages. The technician would open the master template Arlington had provided them. This template included blank versions of the same two sheets each field site had submitted. In addition, this version of the Excel template file contained a "FIRST" tab and a "LAST" tab. The technician would select the "LAST" tab where they would have the option to run the Excel macro compiling all sheets into one file and summarizing the data. It would allow them to select all of the files they wanted summarized into one, essentially opening all 150 files at once. The macro would insert the "Suspense_Template" tab of each file between the "FIRST" and "LAST" tab of the master file. After all of the sheets were inserted into the

master Excel file, formulas would be recalculated populating the cells on the "Suspense_Template" and "Summary" sheets of the master file, thus creating the aggregate submission for higher headquarters.

The current method was a very slow process - imagine running an Excel Macro on 150 files at a time.

The numbers prepared must be reported as a total of all submissions and then broken down into two groups, DFAS or Non-DFAS. Therefore the current macro must be run three times. This portion of the process alone took on average, a half day, during which time the PC used is so overloaded that the technician could not process anything else simultaneously. This process was obviously designed for the smaller centers with much fewer submitting activities and did not run efficiently when combining anywhere from 120 to 150 files.

When the macro combining all of the files was finally finished, and ran three times, the technician would open the file produced from the run of all files submitted. This file, with upwards of 150 tabs, would then be used as the source to copy-and-paste figures cell by cell into a myriad of other reports.

With the new requirement to summarize by MACOM, the technician would have had to print the MACOM table, assign MACOM values to the individual tabs, and come up with some method of totaling the new information by MACOM.

Objectives

Proposed Plan

The proposed application is a form driven executable database application to be loaded on the desktop of the accounting technician previously charged with the completion of this task.

Tables required in this process, aside from the DFAS master table used, will be maintained by the customer using a form-assisted process.

All work would be done using Visual FoxPro 6.0 or Visual FoxPro 8.0. The process will produce all output reports currently submitted. These reports will be produced in the Excel formats as directed by higher headquarters. In other words, it will be a new application that mimics the results produced using the current method. The new results will then be submitted to the higher headquarters, and appear to have been made using the method designed by higher headquarters.

This process will also integrate all reports required, eliminating the need for copy-and-paste process to complete numerous summarizations and views of the same figures.

Adhoc reports will be readily available from the customer's tables. In addition, a one-sheet master Excel file containing all data collected will also be prepared to allow a user less familiar with programming or data query to obtain any summarization required.

In addition the process:

1) Would bring all of the required data in from the Excel Sheets and maintain the information for various computations.

2) Would provide the user with an option to validate data according to the guidelines provided. This would identify activities that submitted invalid data as identified in the guidelines above.

3) Identify new activities that are not on the current tracking system (a paper checklist).

4) Allow the user to be able to do table maintenance as required to add new submitting activities, or re-align Fiscal Station Numbers to new Operating Locations (example – all of the Europe accounts are moving to Rome, NY with the closing of the Europe Accounting Office). This option would also allow the user to print various lists such as which Fiscal Station Numbers currently are assigned to a specific Operating Location.

5) Allow the user to do on-demand reports off of the tables developed listing items such as activities, or groups of activities, that did not submit this month.

6) Allow the user to select an option to run the monthly reports. Output for existing requirements would be produced so that it appeared to have been completed using the macro design provided by higher headquarters. In addition to the three versions of the one higher headquarters requirement that the Excel workbook is currently designed for, an additional 13 reports will be produced. Of this 13, there are 4 unique reports and 1 summary report for each of 9 different fund categories. The unique reports include the following:

a) A report totaling by Operating Agency. Operating Agency is not contained in the original Excel workbook. This information must be assigned based on current accounting system edit tables.

b) A Performance Metric Index report for baseline reporting.

c) A report that ranks and identifies the activities with the top 4 total aged dollar amounts.

d) A report that color codes the summarized results according to how they fit within current guidelines. Balances are identified as red – fail, yellow – watch, or green – satisfactory.

This application was developed according to initial proposal without substantial change prior to implementation. Modifications and improvements were added to the project after implementation.

List of Output Reports Produced

Output File Name followed by **Brief Description Of Output**

(Italics in file name are replaced with actual field value at time file is created)

21ALL*mthyr*.xls Roll up of all files submitted. Output is designed to match original file template and is then submitted to higher headquarters.

21NonDFAS*mthyr*.xls Roll up of all Non-DFAS files submitted. Output is designed to match original file template and is then submitted to higher headquarters.

21DFAS*mthyr*.xls Roll up of all DFAS files submitted. Output is designed to match original file template and is then submitted to higher headquarters.

PMI543.xls Performance Metric Index 543 (PMI543) – tracks absolute totals for various fund categories aged accordingly. For customer internal use only.

RPT702AB.xls Aged Absolute Totals and status according to pre-defined goals prepared to match report requested by higher headquarters. Values are entered into headquarters report template. Report is then submitted to higher headquarters.

FundcategoryMthYr.xls 9 reports created. Summary report by fund category, aged accordingly, and sub-totaled by Operating Location (OPLOC). These reports are for internal customer use only.

SAR*mthyr*.xls Excel file assigns Major Command (MACOM) to Fiscal Station Number (FSN) and creates a summary sheet aged accordingly for each MACOM. Excel file also includes a grand total page for all MACOMs and a key sheet that defines which FSNs were assigned to which MACOM. This report is for internal customer use only.

Ranking.xls Excel file identifies 4 groups with the most change in value from prior month to current month. This report is for internal customer use only.

Scope Of The Project

Overall

The first task in accomplishing this project will be to evaluate the excel file submitted and the published Standard Operating Procedures (SOP) for the current process. This will be critical in the successful completion of all related tasks.

A database structure will be designed to hold the information from the Excel spreadsheets. As columns A through AG (see Excel structure) are either constant or formula based on columns AH through BI, only columns A through AG will be brought into the database. All values in columns AH through BI are represented in one of the preceding formula based columns and would be duplicate information if brought into the database. As lines are brought in from the individual Excel file, the FSN, OPLOC, Month/Year combination will be determined from the name of the file being brought in and then assigned to fields for each line. This information will be used to determine various summary levels and to relate to the MACOM from a DFAS master file later in the process. They will also be assigned a line number value based on the location within the Excel file. This will determine if the line is a header line, detail line, or total line.

This file will be kept in tact for all future uses. Once the file is complete, processes will be designed to run off of the masterfile, preparing all reports currently required. If deletions are required to prepare an individual report, a copy of this masterfile will be created, used for that part of the process, and then erased.

Output will be created using the original Excel report as a template if possible. If no template file is available, output will be designed according to customer specifications.

If done correctly, this will improve the process immensely. It could reduce the time required for the process from 3 to 5 days currently to 10 to 15 minutes overall.

Leaving the masterfile in tact will lend itself to accommodate future uses or immediate data calls against the information collected. This is not readily available with the currently used process. An Excel copy of the masterfile will also be produced to allow users query and summary capability without programming skill.

Excel File (Source) Information and Structure

The Excel files submitted by the field have two tabs, Suspense_Template and Summary. The Summary tab is all formulas based on the values contained on the Suspense_Template tab. The Suspense_Template sheets are all 79 rows long with columns A through BI. The customer can only enter values in columns AH through BI on the Suspense_Template sheet.

All incoming Excel files will have a file name according to one of the following formats:

Char 1 - 2	OPLOC Designator
Char 3 – 7	five digit FSN
Char 8 – 10	three char abbreviation for Month
Char 11 – 12	two digit Year
Char 13 – 16	".xls"

Char 1 - 4	"DFAS"
Char 5 – 10	six digit FSN (leading zero)
Char 11 – 13	three char abbreviation for Month
Char 14 – 15	two digit Year
Char 16 – 19	.xls

Char 1 - 2	"EX"
Char 3 – 6	four character service abbreviation (USAF, NAVY, SDEP, TREA)
Char 7 – 9	three char abbreviation for Month
Char 10 – 11	two digit Year
Char 12 – 15	".xls"
Char 1 - 2	"EX"
Char 3 – 7	five digit FSN
Char 8 – 10	three char abbreviation for Month
Char 11 – 12	two digit Year
Char 13 – 16	".xls"

Each piece of the file name will be used to fill in fields on each line from that Excel file. This information will be used to further assign codes or for further summarization.

A line number will also be assigned as the Excel sheets are brought in. This line number will signify the original row number (1 to 79) within the Excel file. As the source Excel file was primarily designed as a report, the assigned line number will identify the line as header, detail, or total line within the original file. For most processes, only the detail lines will be used but all lines will be kept in tact in a MASTERFILE.DBF.

Excel Source File Structure

Columns	Description	Format
A	Validity of Use	" ", "PROPER", "IMPROPER"
B	Note	" ", "EX"
C	Fund Symbol	various character
D	Limit	various character
E	Use Description/Title	various character

Columns A through E are constant on all spreadsheets submitted. The information in these columns is frozen and cannot be changed.

Columns	Description	Format
F	Use Total – Absolute	formula
G	Use Total – Net	formula
H	Total Number of Trans	formula
I	Total Debit Dollars	formula
J	Total Debit Trans	formula
K	Total Credit Dollars	formula
L	Total Credit Trans	formula

Columns F through L are all formula based on amounts entered in columns AH through BI. Columns F through L are frozen. Values cannot be directly entered into these columns.

Columns	**Description**	**Format**
M,P,S,V, Y,AB,AE	Total Dollar Absolute	formula
N,Q,T,W, Z,AC,AF	Total Dollar Net	formula
O,R,U,X, AA,AD,AG	Total Transactions	formula

Columns M through AG are further designated by age with one three-column grouping for each of the following ages:

0 – 30 days, 31-60 days, 61 – 90 days, 91 – 180 days, 181 days to 1 year, over 1 year but after Oct 1 1997, prior Oct 1 1997

Columns M through AG are all formula based on amounts entered in columns AH through BI. Columns M through AG are frozen. Values cannot be directly entered into these columns.

Columns	Description	Format
AH,AL,AP, AT,AX,BB,BF	Debit Dollars	Numeric – customer entered
AI,AM,AQ, AU,AY,BC,BG	Debit Transactions	Numeric – customer entered
AJ,AN,AR, AV,AZ,BD,BH	Credit Dollars	Numeric – customer entered
AK,AO,AS, AW,BA,BE,BI	Credit Transactions	Numeric – customer entered

Columns AH through BI are further designated by age with one four-column grouping for each of the following ages:

0 – 30 days, 31-60 days, 61 – 90 days, 91 – 180 days, 181 days to 1 year, over 1 year but after Oct 1 1997, prior Oct 1 1997

Columns AH through BI are all numeric amounts entered by the individual customer. Columns AH through BI are the **only** columns the customer can directly enter values into.

Rows 1 – 4 are header lines.

Rows 6 to 23, 25 to 46, 49, 50, 53, 56, 59, 62, 65, 66, 69, and 72 to 79 are detail lines.

Rows 47, 51, 52, 54, 55, 57, 58, 60, 61, 63, 67, 68, and 70 are left blank.

Row 5 is a total line for rows 6 to 24 and 35 to 44.

Row 24 is a total line for rows 25 to 34.

Row 48 is a total line for rows 49 and 50.

Row 64 is a total line for rows 65 and 66.

Row 71 is a total line for rows 72 to 79.

Issues and Implementation

Numerous issues were encountered in the development of this application.

The first issue would have to be actually defining the Excel sheet submitted by each installation and setting up a table to import the Excel sheet correctly. Although each installation submitted the same format sheet prepare from a template, the detail sheet itself was quite complex in design. Each sheet contained 79 rows and 61 columns. In addition to the detail lines, the sheet included report header lines and sub-total lines or total lines rolling up select detail lines. To determine the structure needed to correctly work with the data involved an actual submitted Excel sheet was opened up in Excel. Then the "Suspense Template" sheet, which contained all the detail, was saved as a .dbf file. This file was then used in Visual FoxPro 6.0 and modified as needed to achieve the correct file structure.

Adding onto the issue of defining the structure would be determining how to assign values not included in the actual spreadsheet submitted but required for processing. Fields needed that were not in the actual data submitted included the FSN, the OPLOC, the submission MONTH, the submission YEAR, the MACOM (Major Command), and the row placement from the original Excel sheet. The first four were solved by parsing out the filename and assigning values into the four fields as each file was brought in. After appending one file into the table, the values from the filename would be assigned to the correct fields. A line count field value was then determined by using a count from 1

to 79 based on the starting row of the file just brought in. After the FSN, OPLOC, MONTH, YEAR, and LINECOUNT values were assigned for the current Excel sheet, the process moved on to the next Excel sheet and started the append/assign values routine again. Once all Excel sheets were brought in, the LINECOUNT value was used to determine if the original row was a detail row, total line, or header line. Rows identified as total lines or header lines were deleted from the table leaving just valid detail lines.

The number of Excel sheets being brought in, up to 150, and the extreme number of columns in each sheet turned out to cause the biggest issue of the entire development process. While testing the append process with limited number of files, the process appeared to work fine, however, when running the process with the full set of import files, a fatal error occurred 50% of the time (Figure 4-1). It was as if it was going to fast for itself and couldn't keep up so it just quit.

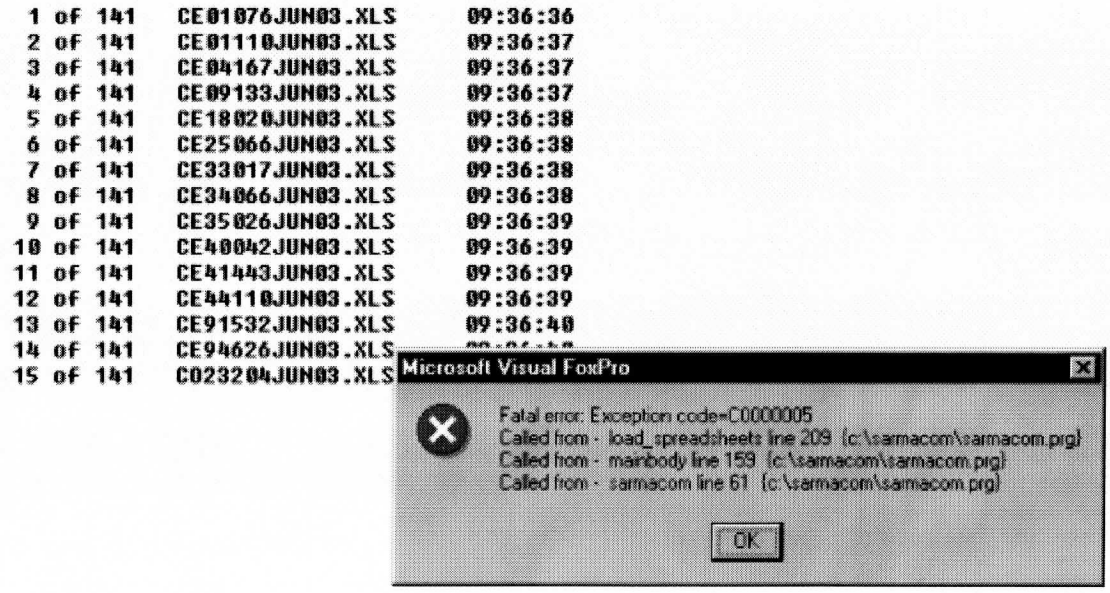


Figure 4-1 Fatal Error

A large amount of unscheduled time was spent researching this and trying to resolve the C0000005 error. This added a whole month onto the 18 day planned implementation schedule.

The first attempt at solving the C0000005 error included putting various timed waits at strategic places within the code giving the program a chance to catch up with itself. I put waits every 10 files, then every 5 files, etc. Each time it appeared as if it was working and would run the program through successfully 5 or 10 times in a row. Then on the next 5 or 10 runs each one would die. I even put a wait after each file and tried that but then the process was so slow it was hardly worth it, and it still didn't guarantee 100% success rate.

I rearranged the code thinking maybe it was just the order presented causing the error. The code was probably rearranged 50 times. I asked more skilled programmers to review the code and offer suggestions and then rearranged the code accordingly. No matter how the code was rearranged, it worked … SOME of the time.

I posted my code and description of the dilemma on Visual FoxPro Forums hoping someone could point me in the right direction. No one had any viable solutions.

I researched the error on Microsoft Visual FoxPro homepage and determined it was a known problem. The problem was supposedly fixed in service pack 2. I was using service pack 5 so that didn't help.

At the time the entire Systems Support team had Visual FoxPro 6.0 or lower. After about a month of trying to get my application to work consistently, we found someone in another work area that had Visual FoxPro 8.0 and decided to give that a try. It couldn't hurt, we tried everything else. Well, I still don't know what caused the error, I just know that shortly thereafter I upgraded to Visual FoxPro 8.0 and never had the problem again.

Even when the application would fail 50% of the time, the customer that requested the application was receiving the final files as requested. If the application did not kill, it worked absolutely correct. Therefore if the process killed, it was just re-run until it didn't. Even with the failed runs and the need to maybe be run repeatedly, it was still faster than the process they were currently using. This was considered our testing phase within the Systems Support section. It was not actually implemented with the customer until it ran 100% of the time correctly.

Other issues included where the final product would be installed and who would run it. It was still undetermined where the application would be installed and if it would be on a specific user's desktop, a public stand-alone machine, or on the network. Until this was determined, the user would come to the Systems Support section and run the process with on-the-spot instruction or assistance available. When this issue was resolved, the application was loaded to an individual user machine. With a change in Network Administration policy, executable applications could be installed on the network given that the proper testing was conducted, documentation was available, and formal written

approval was obtained. The application was modified to run on the network and re-installed.

The last issue involved in this development was changing requirements. Within a month of officially implementing the entire application, headquarters changed the format of the Excel sheet to include an additional aging period. This required complete rework of the original design. All code was reviewed and modified to make future changes more easily accommodated. As part of this upgrade any original code requiring modification was merely commented out and left in place so it could be put back if need be. New code was dated for the same reason. During the re-evaluation to accommodate the additional aging period, multiple file structures needed to be changed to include the new aging. This was changed so that there was one masterfile that remained in tact. If a process needed only selected lines, a workingfile was created as a copy of the masterfile. Then lines were deleted or modified in the working copy. The required report was produced and then the working file was erased leaving the masterfile in original condition.

At the same time, the current process was re-evaluated and additional modifications were implemented to increase run efficiency and make all original data available for future on-the-spot data calls. Initially multiple report processes kept a separate copy of the same data so that certain lines could be deleted or modified for the specific report. Separate files for each report process were no longer maintained and all extra files created during the process were erased at the end of the process.

The project was initially in testing late 2003 with official implementation on the user's desktop in early 2004. As the customer became accustomed to the functionality of the process, they started thinking of other processes or reports that could be added to the system. After about a dozen improvements or change requests, I started tracking and documenting changes made. With the addition of the "History" button on the main user interface screen, information on the changes that had been made became available for review or just curiosity sake.

Process Walk-through and User Interface Screens

Initial Customer Interface screen (Figure 5-1) allows customer to select appropriate month and year of files to be processed. User can then select which process to run.

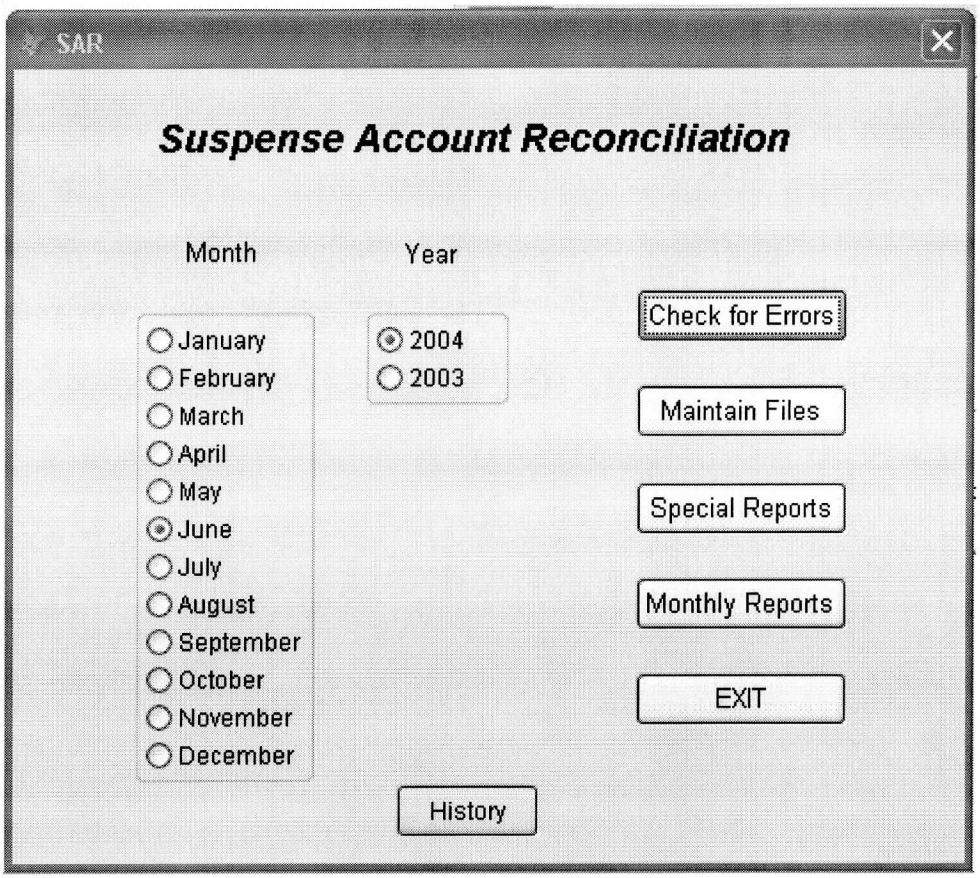


Figure 5-1 Suspense Account Reconciliation Main Interface

Normally the files used to run this process are submitted after month end processing at the various installations. Therefore the month selection on this screen defaults to one month prior to system date. If this is not the correct month to be run, it can be changed as needed.

The year option only consists of two years. The years available for selection are current year and one year prior as determined by analysis of the system date.

The normal monthly sequence would be "Check for Errors" followed by "Monthly Reports". "Maintain Files" and "Special Reports" are as needed processes and are not required to be run on a monthly basis.

If needed, this process could be run repeatedly. Monthly reports are simply re-created with each additional run.

"Check for Errors" would normally be done prior to preparing any monthly reports. It is considered the first step in the normal sequencing of options, however files submitted by the various installations can be brought into the process in either the "Check for Errors" option or the "Monthly Reports" option. As the files are being brought into the process, a "WORKING" screen is displayed (Figure 5-2). This screen lists four items of information for each file as it is brought in. Items displayed are:

1) The number of file being processed

2) The total number of files being brought in

3) The file name

4) The time it started using that file

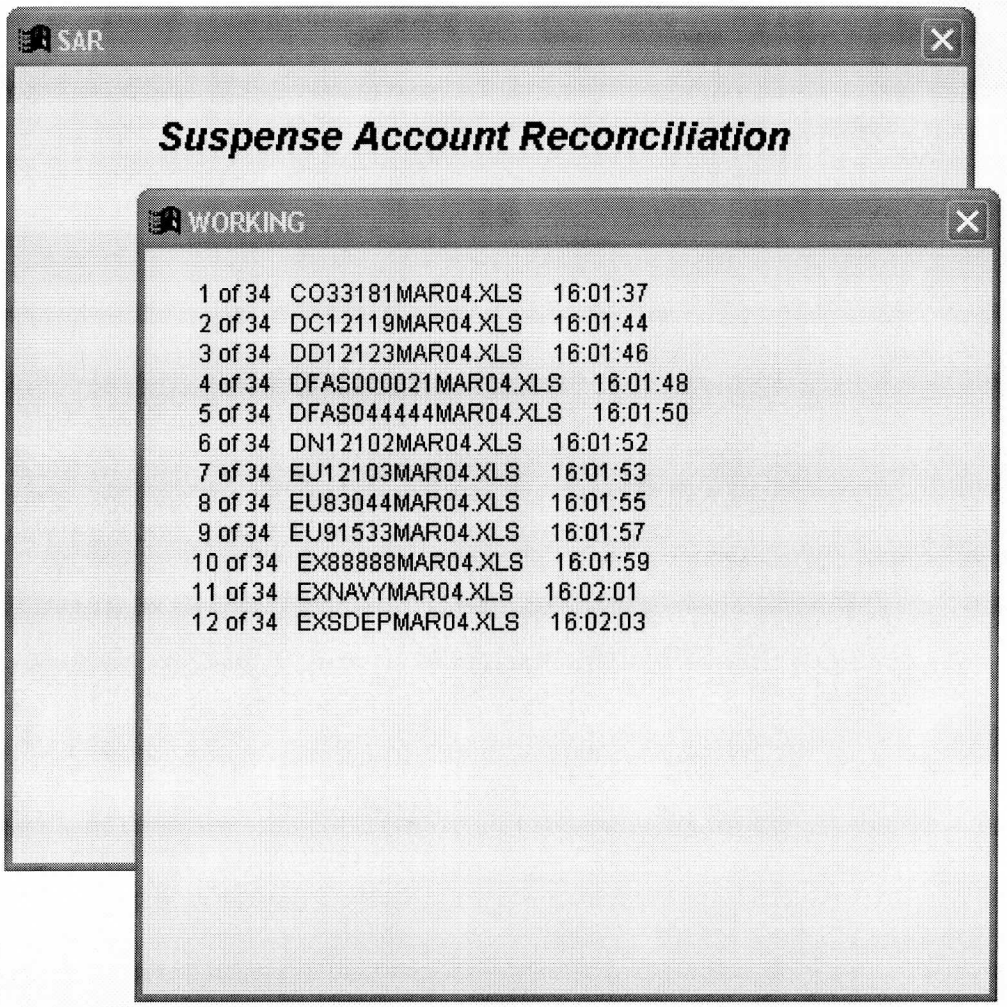


Figure 5-2 WORKING Screen Bringing In Files

The information displayed on this screen was initially built in just for testing and tracking purposes during development but proved to be beneficial for numerous reasons. Therefore it was left in when the final product went into distribution.

To determine which files to bring in, the process looks in a specific directory and identifies all files ending in the 3 character month and 2 digit year immediately followed by the .xls extension. In this example it would bring in all files where the filename ended

in "MAR04.xls". If a file had been misnamed by the user, it would not be brought into the process causing erroneous reports. The user could determine if a file had been misnamed by comparing the total count (34 in this example) to the file count given on the status bar when selecting all intended files in Windows Explorer.

If a file could not be brought in due to some error condition within the file, the screen would stop with the filename of the erroneous file and the technician could research the identified file instead of all 150 files.

The file count and the time displayed also act as a status bar for the possible impatient user.

Normally the file count shown would be between 120 and 150. The screen print shown above was created using a limited number of files for demonstration purposes.

If there were no files found with the specified file naming conventions a message box is displayed identifying the problem to the user (Figure 5-3). This is usually because either the files are in the wrong directory or the user did not select the correct month and year on the initial interface screen.

Figure 5-3 No Files Found

When the user selects "Check for Errors", the lines are brought in from the field submitted Excel files and data is validated according to errors defined by customer. The errors defined are listed below:

1) FSN submitting not matched to OPLOCFSN table

2) FSN submitting file not in Submitters table

3) Filename used has Incorrect OPLOC in filename

4) Absolute amounts cannot be less than net amounts

5) Absolute amounts cannot be less than 0

6) Total Count > 0 but Total Absolute = 0

7) Line 6-13 or 72-74 cannot contain a 61 day amount

8) Line 6-13 or 72-74 cannot contain a 91 day amount

9) Line 6-13 or 72-74 cannot contain a 181 day amount

10) Line 6-13 or 72-74 cannot contain a 1 year amount

11) Line 6-13 or 72-74 cannot contain a prior Oct97 amount

12) The file submitted has an Invalid sheetname

If the error checking routine found errors as defined above, an error report will print out. If no errors were found a message box (Figure 5-4) will identify that no errors were found.

Figure 5-4 No Errors Found

A "Submitters" table is used during the error check routine by this system. This table tracks a 12-month history of the installations that did or did not submit for a particular month. As part of the table, there is a field named according to each month in the year. These fields contain one of usually 5 codes, L, R, blank, Y, or N. The user can maintain the table and insert a value of "L" for late submission or "R" for submitted but file rejected back to installation. The system will mark the field with "Y" if the site submitted a file and the field is currently blank. "N" will be discussed during the "Monthly Reports" option.

At the end of the "Check for Errors" process, a message box is displayed asking if the user wants to empty the current month column (Figure 5-5). This is the only time that values in the corresponding month field are overwritten if they currently contain a value other than blank.

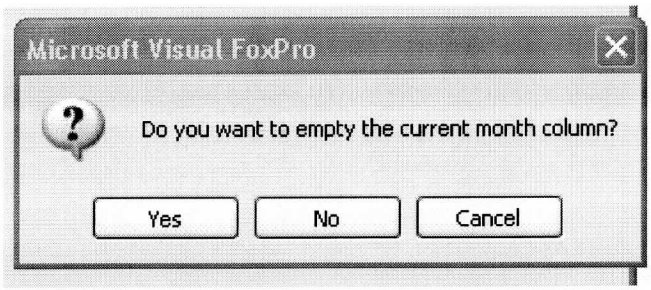


Figure 5-5 Empty Current Month Column

If this is the first run for the month, the current month column contains the values from last year and will need to be emptied out. Answering "Yes" will overwrite all values in the corresponding month field with blank. If it is not the first run this month, this field would already contain values for the correct current month and should not be emptied out.

"Monthly Reports" will run all reports, Headquarters required submissions, and internal customer use only, from the current month set of files submitted.

The "Check for Errors" process brought in all files needed for this process. If you did not have any errors, it is not necessary to bring in all of the files again. The user is given an option to use the existing file or bring in the Excel files again (Figure 5-6).

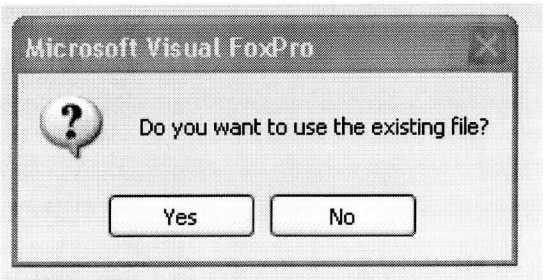


Figure 5-6 Use Existing File

If corrections were made to any of the Excel files used and the error check routine has not been run again, the file should be re-built by selecting the "No" option.

The "WORKING" screen is again utilized and identifies each section of the process as they occur. This is just informational reference point to the user.

During the "Marking Submitters" stage called from the "Monthly Reports" option (as shown on the "WORKING" screen) a message box asks the user if they would like to mark the non-submitters (Figure 5-7). By answering "Yes" the system will mark any remaining blank values in the current month field in the "Submitters" table with "N". This identifies that the site did not submit a file. By answering "No", all blank values remain blank.

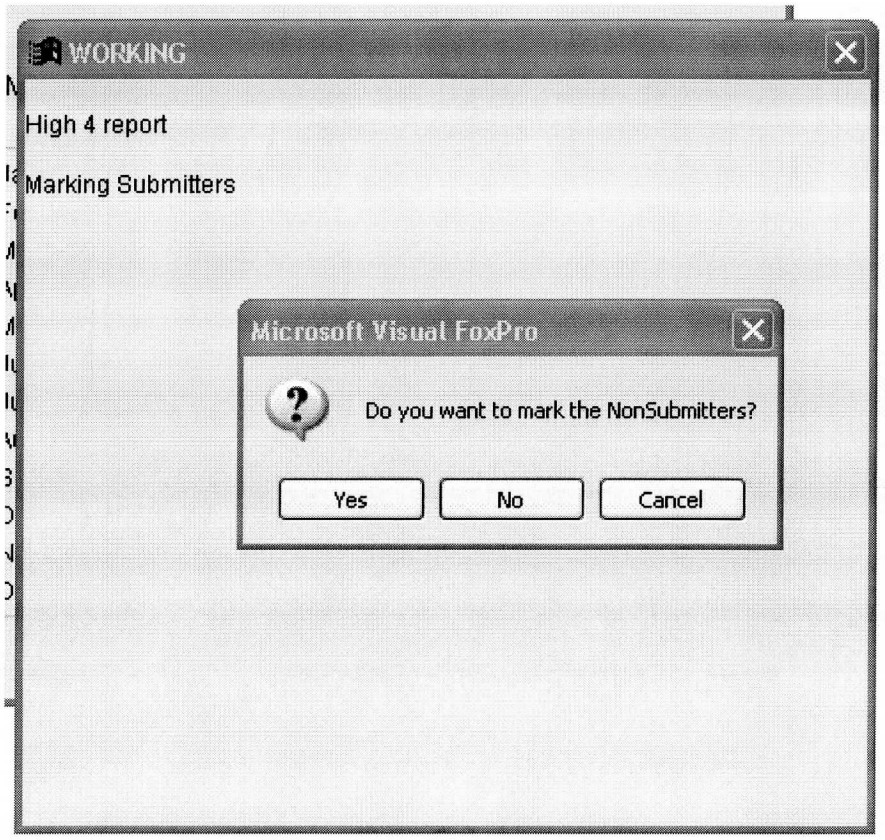


Figure 5-7 Mark Non-Submitters

A message box informs the user when the "Monthly Reports" routine is complete. The user clicks "OK" and is returned to the main "Suspense Account Reconciliation" screen allowing the user to run special reports or maintain files as needed.

If the user selects "Maintain Files" the following screen (Figure 5-8) will be displayed allowing the user to select which file to maintain.

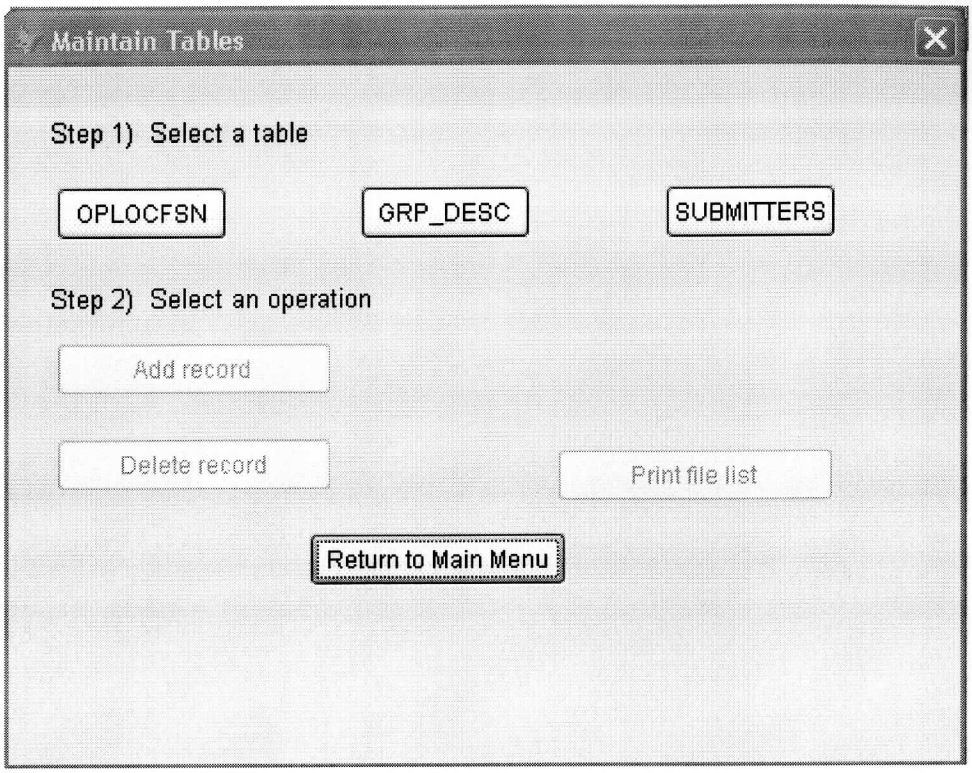


Figure 5-8 Maintain Files Main Interface

Once a file is selected, a table screen will display the file and allow the actual changes (Figures 5-9, 5-10, and 5-11). When this screen is initially displayed the file is in read-only status as a safety precaution.

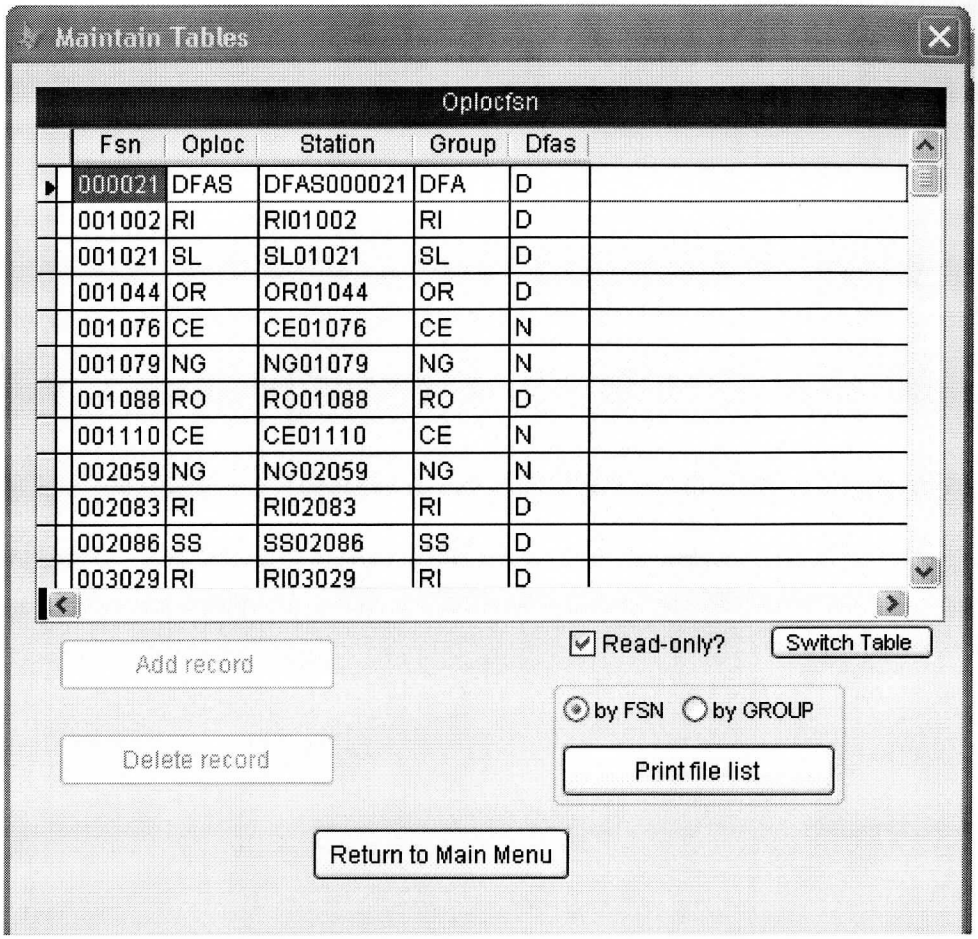


Figure 5-9 OPLOCFSN Table Maintenance

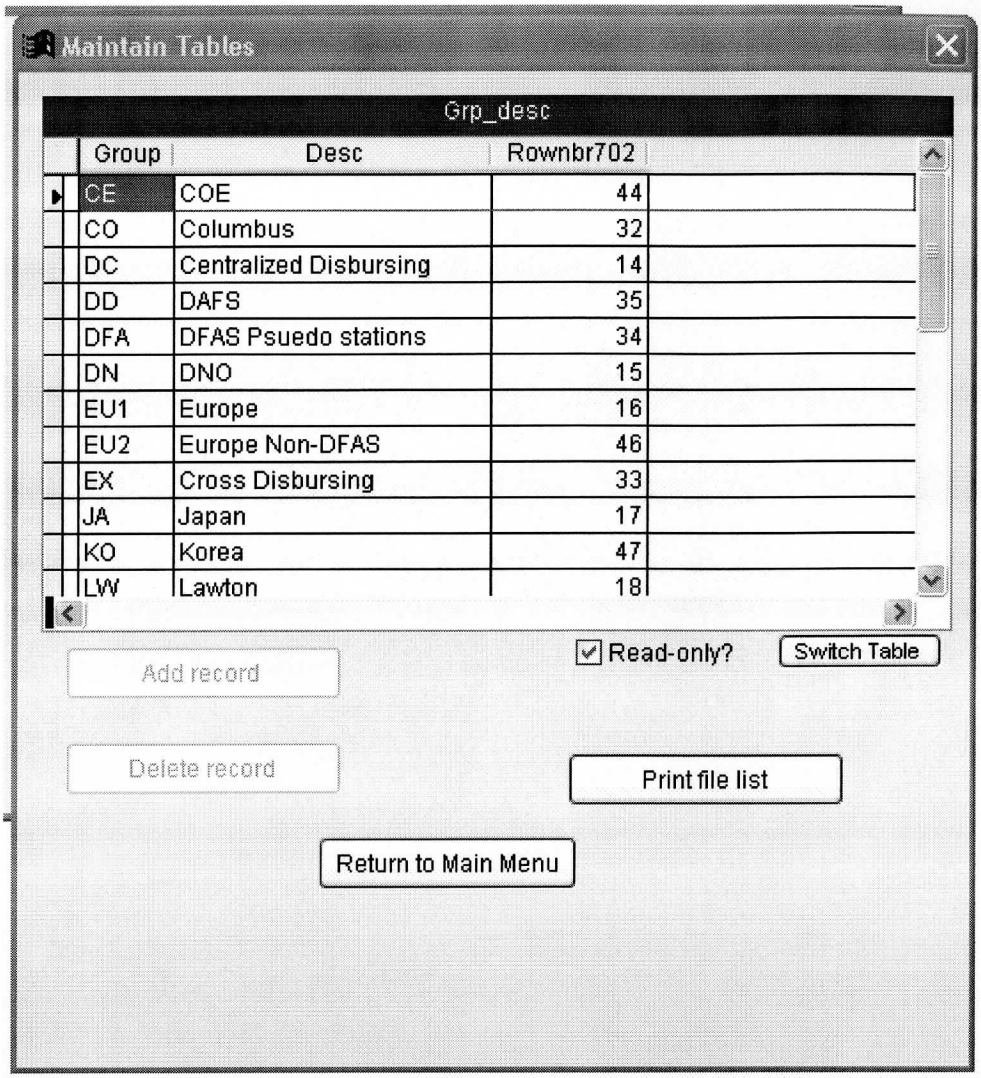


Figure 5-10 GRP_DESC Table Maintenance

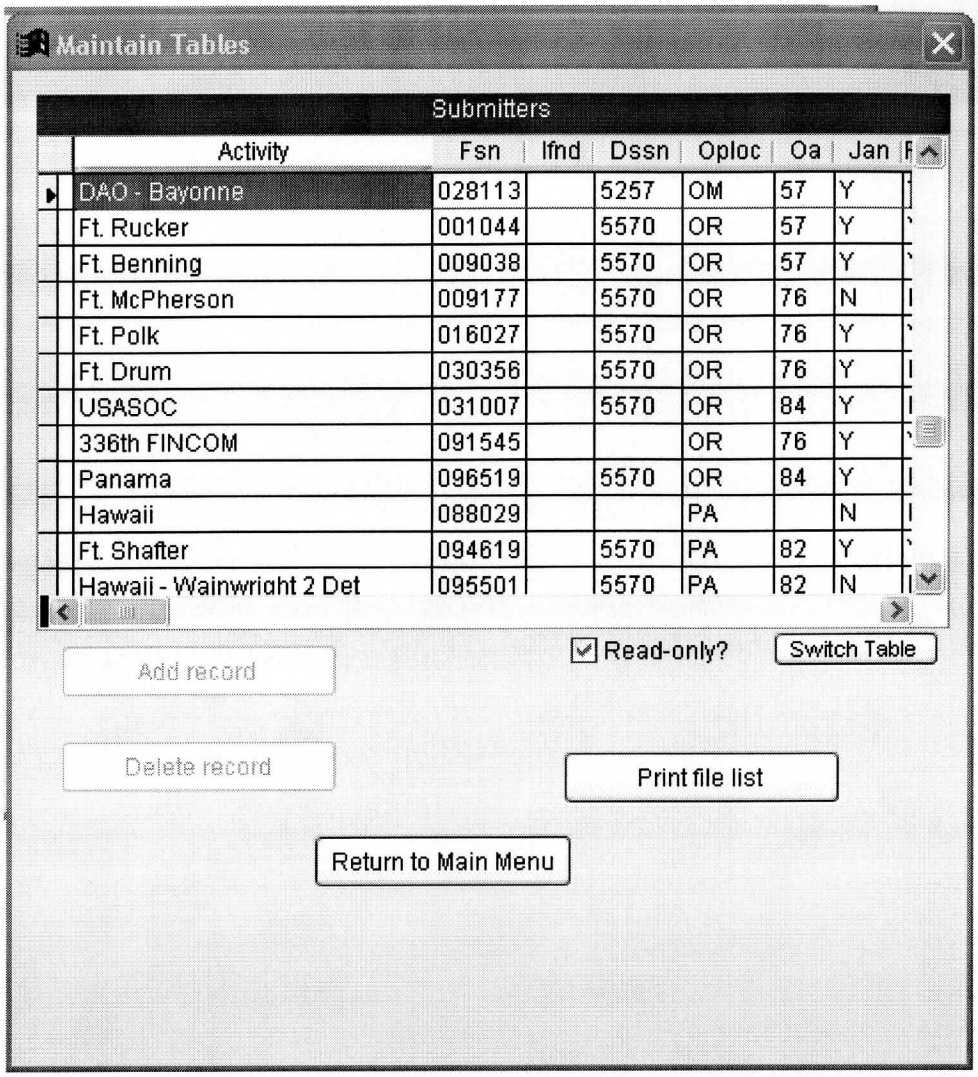


Figure 5-11 Submitters Table Maintenance

The user can change from read only and make changes as necessary. If changed from read only "Add record" and "Delete record" buttons are activated and the user is able to overtype the data in fields.

The file can be printed using the "Print file list" button. The "OPLOCFSN" table has sort options available for the print. "GRP_DESC" prints in alphabetic order and "Submitters" prints in a pre-determined order according to customer specifications.

If the user selects "Special Reports" a screen will be displayed allowing the user to select which reports they would like prepared (Figure 5-12).

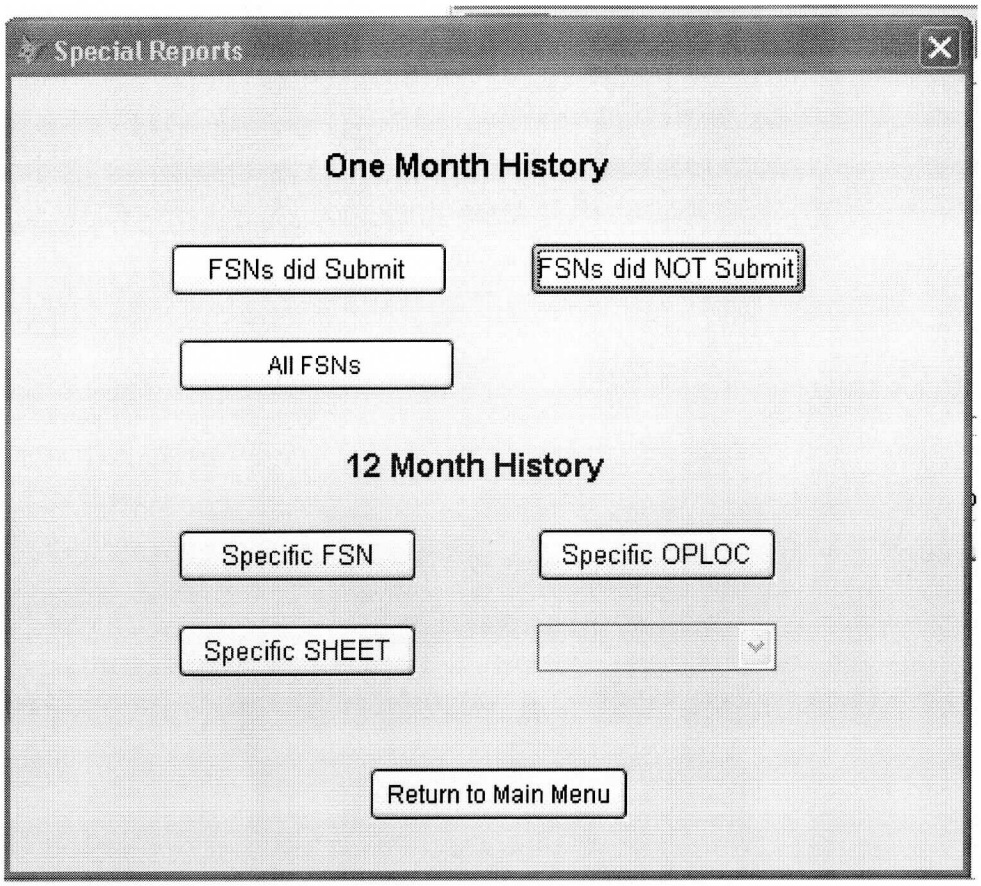


Figure 5-12 Special Reports Interface

The user can print a report detailing FSN's that did submit, FSN's that did not submit, or a report showing the submission status for all FSN's for the currently selected Month and Year (initial screen). They can also print a 12-month submission history for a specific

FSN, a specific OPLOC, or a specific sheet where a sheet is a pre-defined grouping set by the user. If a 12-month option is selected, a drop-down box is populated with all available choices for the particular 12-month option selected.

With the addition of the "History" button, information on the changes that had been made became available for review or just curiosity sake (Figure 5-13). This was designed to be scrollable to show numerous updates.

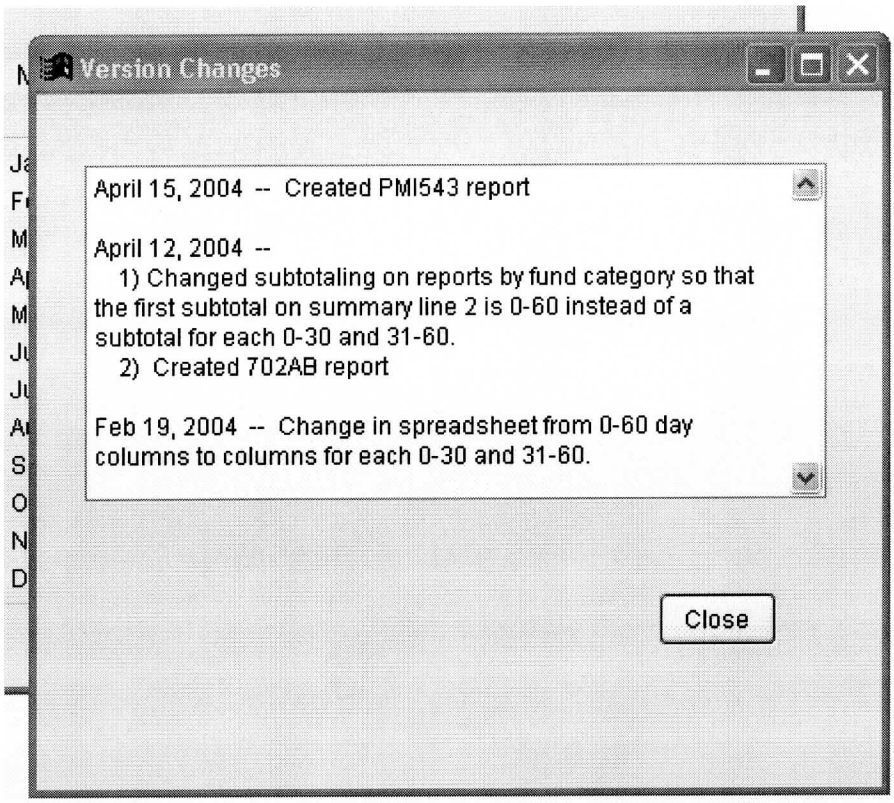


Figure 5-13 Historical Changes

"Exit" closes the application and returns the user to their desktop.

Conclusion

This project, born out of the need for a solution to new requirements of an old process, afforded me the opportunity to develop a somewhat complex application from start to finish including planning, proposal of solutions, development, implementation, user training, and enhancements. I acquired additional programming skills that will be useful in many future applications irregardless of languages used. These skills included interfacing between various languages and software packages, object orientation, development of effective user interface designs, and even additional research avenues available for error resolution. Through a change of required file structure after the initial version of the application was implemented, I also learned the importance of planning in advance for future unknown scenarios, documentation of code, and the need to program in a more easily maintainable and upgradeable style.

The finished product was much more than the customer asked for initially but not more than they needed. At the start, the customer did not believe what they had asked for could be done, let alone the degree to which the current process could be improved upon.

Work Breakdown Schedule

Prepared by: Linda Campbell **Date: July 12, 2004**

1. Initiating
    - 1.1.1. Select Project
    - 1.1.2. Form Project Team
    - 1.1.3. Evaluate Current Process
    - 1.1.4. Recommend Solutions

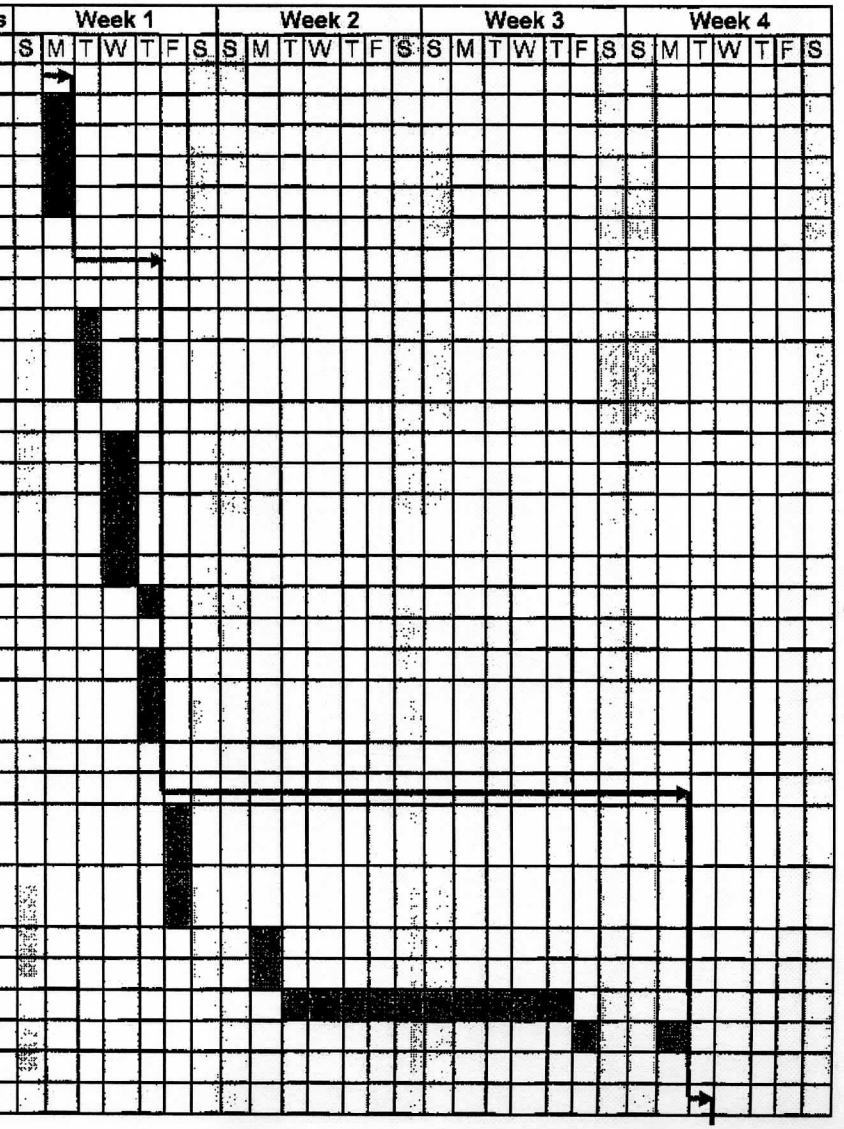
2. Planning
    - 2.1.1. Research
        - 2.1.1.1.1. Research with current user
        - 2.1.1.1.2. Research current files/processes/methods
    - 2.1.2. Define Requirements
        - 2.1.2.1.1. Identify User requirements
        - 2.1.2.1.2. Identify required output
        - 2.1.2.1.3. Identify additional output requested
        - 2.1.2.1.4. Define validation process
    - 2.1.3. Define deliverables
    - 2.1.4. Develop Plan
        - 2.1.4.1.1. Prepare Statement of Work
        - 2.1.4.1.2. Create Work Breakdown Structure

3. Development
    - 3.1.1. Define incoming file structure
    - 3.1.2. Define table structures (*.dbf) to be used
    - 3.1.3. Design interfaces
    - 3.1.4. Design reports
    - 3.1.5. Code application
    - 3.1.6. Test application

4. Implementation
    - 4.1.1. Prepare installable executable
    - 4.1.2. Create implementation package
    - 4.1.3. Install at customer's station

5. Support
    - 5.1.1. Train user
    - 5.1.2. User support and enhancements as required

	Task Name	Duration	Predecessors
1	**Initiating**	1	
1.1	Select Project	1	
1.2	Form Project Team	1	
1.3	Evaluate Current Process	1	
1.4	Recommend Solutions	1	
2	**Planning**	3	1
2.1	Research		
2.1.1	Research with current user	1	
2.1.1	Research current files/processes/methods	1	
2.2	Define Requirements		2.1
2.2.1	Identify User requirements	1	
2.2.2	Identify required output	1	
2.2.3	Identify additional output requested	1	
2.2.4	Define validation process	1	
2.3	Define deliverables	1	2.1
2.4	Develop Plan		2.1
2.4.1	Prepare Statement of Work	1	
2.4.2	Create Work Breakdown Structure	1	
3	**Development**	12	2
3.1	Define incoming file structure	1	
3.2	Define table structures (*.dbf) to be used	1	
3.3	Design interfaces	1	
3.4	Design reports	1	
3.5	Code application	8	3.1, 3.2
3.6	Test application	2	3.5
4	**Implementation**	1	3



Gantt Chart (Figure A-1)

Gantt Chart (Figure A-1 cont.)

	Task Name	Duration	Predecessors	Week 1							Week 2							Week 3							Week 4						
				S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S
4.1	Prepare installable executable	1																													
4.2	Create implementation package	1	4.1																												
4.3	Install at customer's station	1	4.1, 4.2																												
5	**Support**	1	**4**																												
5.1	Train user	1																													
5.2	User support and enhancements as required	1																													

Table Structures (*.Dbf Files)

Structure for table: **MASTERFILE.DBF**
Purpose: to bring all Excel source lines into, stays in tact with all lines as a working file throughout the process.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	VALIDITY	Character	12				No
2	NOTE	Character	7				No
3	FNDSYMBL	Character	18				No
4	LIMIT	Character	28				No
5	DESC	Character	65				No
6	TOTABS	Numeric	16	2			No
7	TOTNET	Numeric	16	2			No
8	TOTCNT	Numeric	20				No
9	TDRDOL	Numeric	15	2			No
10	TDRCNT	Numeric	15				No
11	TCRDOL	Numeric	15	2			No
12	TCRCNT	Numeric	15				No
13	TADOL0	Numeric	15	2			No
14	TNDOL0	Numeric	15	2			No
15	TCNT0	Numeric	15				No
16	TADOL31	Numeric	15	2			No
17	TNDOL31	Numeric	15	2			No
18	TCNT31	Numeric	15				No
19	TADOL61	Numeric	15	2			No
20	TNDOL61	Numeric	15	2			No
21	TCNT61	Numeric	15				No
22	TADOL91	Numeric	15	2			No
23	TNDOL91	Numeric	15	2			No
24	TCNT91	Numeric	15				No
25	TADOL181	Numeric	15	2			No
26	TNDOL181	Numeric	15	2			No
27	TCNT181	Numeric	15				No
28	TADOL1YR	Numeric	15	2			No
29	TNDOL1YR	Numeric	15	2			No
30	TCNT1YR	Numeric	15				No
31	TADOLOCT97	Numeric	15	2			No
32	TNDOLOCT97	Numeric	15	2			No
33	TCNTOCT97	Numeric	15				No
34	DRDOL0	Numeric	15	2			No
35	DRCNT0	Numeric	15				No
36	CRDOL0	Numeric	15	2			No
37	CRCNT0	Numeric	15				No
38	DRDOL31	Numeric	15	2			No

Structure for table: **MASTERFILE.DBF** **continued**

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
39	DRCNT31	Numeric	15				No
40	CRDOL31	Numeric	15	2			No
41	CRCNT31	Numeric	15				No
42	DRDOL61	Numeric	15	2			No
43	DRCNT61	Numeric	15				No
44	CRDOL61	Numeric	15	2			No
45	CRCNT61	Numeric	15				No
46	DRDOL91	Numeric	15	2			No
47	DRCNT91	Numeric	15				No
48	CRDOL91	Numeric	15	2			No
49	CRCNT91	Numeric	15				No
50	DRDOL181	Numeric	15	2			No
51	DRCNT181	Numeric	15				No
52	CRDOL181	Numeric	15	2			No
53	CRCNT181	Numeric	15				No
54	DRDOL1YR	Numeric	15	2			No
55	DRCNT1YR	Numeric	15				No
56	CRDOL1YR	Numeric	15	2			No
57	CRCNT1YR	Numeric	15				No
58	DRDOLOCT97	Numeric	15	2			No
59	DRCNTOCT97	Numeric	15				No
60	CRDOLOCT97	Numeric	15	2			No
61	CRCNTOCT97	Numeric	15				No
62	FSN	Character	6				No
63	OPLOC	Character	5				No
64	MONTH	Character	5				No
65	FUNDCAT	Character	18				No
66	FILENAME	Character	20				No
67	MACOM	Character	2				No
68	LINENUM	Numeric	3				No

Structure for table: **AIMHIGH4.DBF**
Purpose: to keep a 12-month history of each group so the 4 groups with the most change in value from prior month to current month can be identified.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	GROUP	Character	3				No
2	JAN	Numeric	19	2			No
3	FEB	Numeric	19	2			No
4	MAR	Numeric	19	2			No
5	APR	Numeric	19	2			No
6	MAY	Numeric	19	2			No
7	JUN	Numeric	19	2			No
8	JUL	Numeric	19	2			No
9	AUG	Numeric	19	2			No
10	SEP	Numeric	19	2			No
11	OCT	Numeric	19	2			No
12	NOV	Numeric	19	2			No
13	DEC	Numeric	19	2			No
14	CHANGE	Numeric	19	2			No
15	RANKING	Character	1				No

Structure for table: **AIMOUT1.DBF**
Purpose: a working file to use in the AIMOUT processes.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	BSN	Character	4				No
2	B1	Character	1				No
3	STATION	Character	10				No
4	B2	Character	1				No
5	B3	Character	1				No
6	TADOL0	Numeric	15	2			No
7	TNDOL0	Numeric	15	2			No
8	TCNT0	Numeric	15				No
9	TADOL31	Numeric	15	2			No
10	TNDOL31	Numeric	15	2			No
11	TCNT31	Numeric	15				No
12	TADOL61	Numeric	15	2			No
13	TNDOL61	Numeric	15	2			No
14	TCNT61	Numeric	15				No
15	TADOL91	Numeric	15	2			No
16	TNDOL91	Numeric	15	2			No
17	TCNT91	Numeric	15				No
18	TADOL181	Numeric	15	2			No
19	TNDOL181	Numeric	15	2			No
20	TCNT181	Numeric	15				No
21	TADOL1YR	Numeric	15	2			No
22	TNDOL1YR	Numeric	15	2			No
23	TCNT1YR	Numeric	15				No
24	TADOLOCT97	Numeric	15	2			No
25	TNDOLOCT97	Numeric	15	2			No
26	TCNTOCT97	Numeric	15				No
27	TOTABS	Numeric	16	2			No
28	TOTNET	Numeric	16	2			No
29	TOTCNT	Numeric	20				No
30	FSN	Character	6				No
31	OPLOC	Character	5				No
32	GROUP	Character	3				No
33	NOTE	Character	7				No
34	FUNDCAT	Character	10				No

Structure for table: **ERR_MESSAGE.DBF**
Purpose: to hold the Error Message that relates to each error type.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	ERR	Character	1		Asc	Machine	No
2	MESSAGE	Character	50				No

Structure for table: **EXCEL_MAP.DBF**
Purpose: Identifies where to print specific fields on output reports.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	FIELD_NAME	Character	12				No
2	EXCEL_COL	Character	2				No
3	COL_NUM	Numeric	2				No
4	FIELD_TYPE	Character	1				No
5	FIELD_LEN	Numeric	3				No
6	FIELD_DEC	Numeric	3				No
7	FIELD_NULL	Logical	1				No
8	FIELD_NOCP	Logical	1				No
9	TABLE_NAME	Character	20				No
10	FIELD_NEXT	Integer	4				No
11	FIELD_STEP	Integer	4				No

Structure for table: **EXCEL_MAP_543.DBF**
Purpose: Identifies where to print specific fields on the Performance Metric Index 543 output report.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	LINENUM	Numeric	2		Asc	Machine	No
2	EXCEL_ROW	Numeric	2				No

Structure for table: **FILESRECD.DBF**
Purpose: to track installations that submitted files for current processing month.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	FSN	Character	6		Asc	Machine	No
2	IFND	Character	2				No
3	OPLOC	Character	5				No
4	FILENAME	Character	20				No
5	LINENUM	Numeric	3				No
6	ERR	Character	1				No

Structure for table: **GRP_DESC.DBF**
Purpose: associates an English description to the various group codes. Also assigns the Excel row number for output associated to that group in the 702 output report.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	GROUP	Character	3		Asc	Machine	No
2	DESC	Character	22				No
3	ROWNBR702	Numeric	2				No

Structure for table: **OPLOCFSN.DBF**
Purpose: customer maintained file to relate various field values.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	FSN	Character	6		Asc	Machine	No
2	OPLOC	Character	5				No
3	STATION	Character	10		Asc	Machine	No
4	GROUP	Character	3		Asc	Machine	No
5	DFAS	Character	1				No

Structure for table: **PMI543.DBF**
Purpose: working file for the PMI543 process.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	TADOL0	Numeric	16	2			No
2	TADOL31	Numeric	16	2			No
3	TADOL61	Numeric	16	2			No
4	TADOL91	Numeric	16	2			No
5	TADOL181	Numeric	16	2			No
6	TADOL1YR	Numeric	16	2			No
7	TADOLOCT97	Numeric	16	2			No
8	LINENUM	Numeric	3				No

Structure for table: **RPT702AB.DBF**
Purpose: working file to produce output for the 702AB report.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	GROUP	Character	3		Asc	Machine	No
2	M0TO60	Numeric	15	2			No
3	M61TO90	Numeric	15	2			No
4	MOVER90	Numeric	15	2			No

Structure for table: **SARERROR.DBF**
Purpose: identifies errors found during data validation according to customer error definitions.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	FILENAME	Character	20		Asc	Machine	No
2	LINENUM	Numeric	3				No
3	ERR	Character	1				No
4	MESSAGE	Character	50				No

Structure for table: **SUBMITTERS.DBF**
Purpose: to track a 12-month history of installations that did/did not submit files for processing.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	ACTIVITY	Character	25				No
2	FSN	Character	6		Asc	Machine	No
3	IFND	Character	2				No
4	DSSN	Character	4				No
5	OPLOC	Character	4		Asc	Machine	No
6	OA	Character	2				No
7	JAN	Character	1				No
8	FEB	Character	1				No
9	MAR	Character	1				No
10	APR	Character	1				No
11	MAY	Character	1				No
12	JUN	Character	1				No
13	JUL	Character	1				No
14	AUG	Character	1				No
15	SEP	Character	1				No
16	OCT	Character	1				No
17	NOV	Character	1				No
18	DEC	Character	1				No
19	SHEET	Character	5		Asc	Machine	No

Structure for table: **OPFSN.DBF**
Purpose: DFAS driven master table relating OPLOC, FSN, and MACOM fields.

Field	Field Name	Type	Width	Dec	Index	Collate	Nulls
1	CENTER	Character	3				No
2	OPLOC	Character	2		Asc	Machine	No
3	FSN	Character	6		Asc	Machine	No
4	FIELDSITE	Character	25				No
5	FSNINFO	Character	34				No
6	MACOM	Character	2				No
7	SERVICE	Character	2				No

PROGRAM CODE

```
*************************************************************************
**                        main.PRG                                     **
*************************************************************************
**      This program is called by Sar_Aim.exe                          **
**           main program for the executable                           **
*************************************************************************
*************************************************************************

SET SAFETY OFF
SET PROCEDURE TO PROGRAM() additive
CLOSE TABLES all

CLEAR

public gcsort, gcnonsort, mMthYr, mMth, mYr, gcdid_not, mNumMth

STORE '' TO gcsort, gcnonsort, gcdid_not

tempdate = GOMONTH(DATE(),-1)
mMth = UPPER(SUBSTR(cmonth(tempdate),1,3))
mYr = RIGHT(ALLTRIM(STR(YEAR(DATE()))),2)
mNumMth = ALLTRIM(STR(MONTH(tempdate)))
STORE mMth + mYr TO mMthYr

DO FORM scx\menu1

READ events
```

```
*************************************************************************
**          d:\scr44567\aimoutput2.PRG                                 **
*************************************************************************
**      This program is called by Sar_Aim.exe                          **
**                                                                     **
**      takes the SARMACOM data and computes the aim report excel lines **
*************************************************************************
*************************************************************************

************* added 4/07/04 for request to produce Report 702ab *************
************* end of added for request to produce Report 702ab **************

SET SAFETY OFF

****************************************
PUBLIC mfundcat                         && 1/30/04

IF !USED('rpt702ab')
   USE rpt702ab IN 0
endif
   SELECT rpt702ab
   ZAP
   use

** open all files needed during loop
IF !USED('oplocfsn')
   USE oplocfsn IN 0                    && 1/30/04
ENDIF
IF !USED('aimout1')
   USE aimout1    IN 0                  && 1/30/04
ENDIF
IF !USED('grp_desc')
   USE GRP_DESC IN 0                    && 1/30/04
ENDIF
IF !USED('workingfile')
   USE workingfile IN 0                 && 2/17/04
ENDIF

DO prepare_data    && bring data in from SARMACOM.prg and prepare
                   **    for AIM reports

SELECT DISTINCT fundcat FROM workingfile INTO TABLE fundcat_list   && 2/17/04
SELECT fundcat_list                     && 1/30/04

DO WHILE !EOF()                         && 1/30/04
   mfundcat = ALLTRIM(fundcat_list.fundcat)   && 1/30/04

   DO fundcat_data                      && 1/30/04

   DO generate_spreadsheet

   SELECT fundcat_list                  && 1/30/04
   SKIP                                 && 1/30/04
enddo                                   && 1/30/04

*!*   ************* added 4/07/04 for request to produce Report 702ab *************
```

```
*!*   DO rpt702ab
*!*   ************* end of added for request to produce Report 702ab **************

DO close_down


*******************************************************************************
*******************************************************************************
**                                                                           **
** section to bring data in from SARMACOM.prg and prepare for AIM reports    **
**                                                                           **
PROCEDURE prepare_data
**                                                                           **
*******************************************************************************
*******************************************************************************

  sele workingfile          && created in mastercopy.prg

  SELECT oplocfsn
  INDEX ON station TAG t_stat

  SELECT aimout1
    ** modified structure to match required output file has OPLOC, FSN,
    ** STATION, NOTE, and GROUP fields not to be output
  ZAP
  APPEND FROM workingfile
  INDEX ON oploc + fsn TO x

  SCAN
    DO case
      case oploc = 'DFAS'
        m.station = 'DFAS' + fsn
      CASE oploc = 'EX' AND ALLTRIM(fsn) = '088888'
        m.station = 'EX' + '88888'
      CASE oploc = 'EX' AND ALLTRIM(fsn) != '088888'
        m.station = 'EX' + ALLTRIM(fsn)
      otherwise
        m.station = ALLTRIM(oploc) + RIGHT(fsn,5)
    ENDCASE

    IF SEEK(m.station,'oplocfsn','t_stat')
      m.group = oplocfsn.group
    ELSE
      m.group = 'XXX'
    endif

    replace station WITH m.station, group WITH m.group

  ENDSCAN

  **** to display lines with invalid oploc fsn combination
  COUNT FOR group = 'XXX' TO invalid_group
  IF invalid_group > 0
    MESSAGEBOX("There were FSN's not previously assigned to an OPLOC." ;
              + CHR(13) + ;
           "They will be listed at the bottom of the AIM report " ;
```

```
            + CHR(13) + ;
          "until they are entered into the OPLOCFSN table. ";
            + CHR(13) + ;
          "Thank you." ,0+64)
    ** want a list to print out for all of the FSNs that will be printed
    ** at the bottom of this report but only one line per FSN
    SELECT DISTINCT station ;
      FROM aimout1 ;
      WHERE aimout1.group = 'XXX' ;
      INTO CURSOR xxx_cursor
      ** created a cursor file which is automatically removed upon
      ** program completion, also causes the cursor file just created
      ** to be selected as currently active file
    LIST station to print off
    SELECT aimout1          && file that was active before cursor created
  ENDIF

return          && returns to calling part of program
*******************************************************************************
*******************************************************************************
**                                                                           **
**              end of PROCEDURE prepare_data                                **
**                                                                           **
*******************************************************************************
*******************************************************************************


*******************************************************************************
*******************************************************************************
**                                                                           **
** section to bring data in from SARMACOM.prg and prepare for AIM reports    **
**                                                                           **
PROCEDURE fundcat_data
**                                                                           **
*******************************************************************************
*******************************************************************************

? 'reporting for fundcat ', mfundcat

  sele aimout1          && appended from working file in procedure prepare_data
  TOTAL ON oploc + fsn TO aimtotal FOR note != 'EX' AND fundcat = mfundcat

*******************************************************************************
*******************************************************************************
**                                                                           **
**              end of PROCEDURE fundcat_data                                **
**                                                                           **
*******************************************************************************
*******************************************************************************


*******************************************************************************
*******************************************************************************
**                                                                           **
**            section to create formal output to EXCEL                       **
**                                                                           **
```

```
PROCEDURE generate_spreadsheet
**                                                                              **
********************************************************************************
********************************************************************************


************************* output to Excel ******************************
********************************************************************************
** want one workbook with one worksheet listing detail lines by group
** followed by summary lines for the group, then the next group

** uses aimtotal.dbf


  #DEFINE xlLandscape 2                 && defines page orientation
  #DEFINE autoIn2Pts 72

  SELECT GRP_DESC
  INDEX ON group TO y

  use aimtotal IN 0
  SELECT aimtotal
  INDEX ON group + fsn TO z

  local oBook, oRange, oSheet
  mGroup = "   "                        && initialize
  mgrp_desc = "         "               && initialize

  oExcel = createobject("Excel.application")
         ** creates the application, opens an Excel session without
         ** opening a excel file

*******************************
  oExcel.visible = .F.                  && do not show Excel spreadsheet
*!*      oExcel.visible = .T.           && show Excel spreadsheet as it is being created
*******************************
  oExcel.application.displayalerts = .f.  && turns off the excel warning
                                        ** message boxes

  oBook = oExcel.workbooks.add()        && creates an actual workbook

  oSheet = oBook.Activesheet            && opens book with the default
                                        ** number of worksheets

  numsheets = oExcel.activeworkbook.sheets.count
         ** see how many sheets are in a new file

  oSheet1 = oBook.Worksheets[numsheets]
         ** defines the last sheet of the workbook if opened with the
         ** default number of sheets.  used to keep track of the last
         ** sheet worked      and delete original blank sheets later


  ************************************************
  ** get the sheet names for the default number of sheets to delete later
  declare sheetname(numsheets)          && declares array for sheetnames
```

```
  with oExcel
    for i = 1 to numsheets
      .activeworkbook.sheets(i).activate
      sheetname[i] = .activesheet.name
        ** populates array with sheetnames
    endfor
  endwith                                  && oExcel
  ***************************************************

    oSheet2 = oBook.Worksheets.Add(,oSheet1)
        ** Add a sheet for the detail report
        ** oSheet2 identifies the new sheet
        ** (,oSheet1) tells it to add the new sheet AFTER the previous
        **  sheet

    with oSheet2
      .name = "ALL"                        && Name the sheet

**
      oSheet1 = oBook.Activesheet
        ** resets the identifier for the previous sheet
**

      .select()                            && current sheet

      ** assign column headings
      .Range("A1:Z1").Font.Bold = .T.          && bold the column headings
      .range("A1").Value = "BSN"
      .range("C1").Value = "STATION"
********************** change for new structure 2/19/04 **********************
      .range("F1").Value = "0-30"
      .range("I1").Value = "31-60"
      .range("L1").Value = "61-90"
      .range("O1").Value = "91-180"
      .range("R1").Value = "181-1Yr"
      .range("U1").Value = "OVER 1 YEAR to 10/1/97"
      .range("X1").Value = "OVER 10/1/97"
      .range("AA1").Value = "TOTAL"

      ** assign column heading colors
      .range("F1:H1").interior.color = RGB(255,202,155)        && peach
      .range("I1:K1").interior.color = RGB(203,157,253)        && lavender
      .range("L1:N1").interior.color = RGB(255,145,200)        && pink
      .range("O1:Q1").interior.color = RGB(255,255,128)        && yellow
      .range("R1:T1").interior.color = RGB(183,255,183)        && mint
      .range("U1:W1").interior.color = RGB(159,207,255)        && lt. blue
      .range("X1:Z1").interior.color = RGB(247,202,15)         && goldenrod
      .range("AA1:AC1").interior.color = RGB(180,180,180)      && gray

      ** Header Row is wider than other rows
       .ROWS("1:1").ROWHEIGHT = 47.25

            ** detail starts printing on next line after headings
      oRange = .Range("A2:AC2")
            ** sets the range to the first detail line to be filled in
```

```
*!*             .range("F1").Value = "0-60"
*!*             .range("I1").Value = "61-90"
*!*             .range("L1").Value = "91-180"
*!*             .range("O1").Value = "181-1Yr"
*!*             .range("R1").Value = "OVER 1 YEAR to 10/1/97"
*!*             .range("U1").Value = "OVER 10/1/97"
*!*             .range("X1").Value = "TOTAL"
*!*
*!*             ** assign column heading colors
*!*             .range("F1:H1").interior.color = RGB(203,157,253)      && lavender
*!*             .range("I1:K1").interior.color = RGB(255,145,200)      && pink
*!*             .range("L1:N1").interior.color = RGB(255,255,128)      && yellow
*!*             .range("O1:Q1").interior.color = RGB(183,255,183)      && mint
*!*             .range("R1:T1").interior.color = RGB(159,207,255)      && lt. blue
*!*             .range("U1:W1").interior.color = RGB(247,202,15)       && goldenrod
*!*             .range("X1:Z1").interior.color = RGB(180,180,180)      && gray
*!*
*!*             ** Header Row is wider than other rows
*!*              .ROWS("1:1").ROWHEIGHT = 47.25
*!*
*!*                     ** detail starts printing on next line after headings
*!*             oRange = .Range("A2:Z2")
*!*                     ** sets the range to the first detail line to be filled in

****************** end of change for new structure 2/19/04 ******************


        do while not eof()

           mGroup = aimtotal.group
              **  used to keep track of the section being worked
           IF SEEK(mGroup,'GRP_DESC')
              mgrp_desc = GRP_DESC.desc
           ELSE
*!*                   m.group = 'XXX'
              mgrp_desc = "FSN/OPLOC group Unassigned"
           endif

           startRow = oRange.Row
              **  Row property always give the first row of range.
           endRow = oRange.Row
           do while mGroup = aimtotal.group and not eof()
              with oRange
                 ** fill in detail line values
                 .Columns[1].Value = aimtotal.bsn
                 .Columns[3].Value = aimtotal.station
                 .Columns[6].Value = aimtotal.tadol0
                 .Columns[7].Value = aimtotal.tndol0
                 .Columns[8].Value = aimtotal.tcnt0

********************** change for new structure 2/19/04 **********************
                 .columns[9].value = aimtotal.tadol31
                 .columns[10].value = aimtotal.tndol31
                 .columns[11].value = aimtotal.tcnt31
                 .Columns[12].Value = aimtotal.tadol61
```

```
                .Columns[13].Value = aimtotal.tndol61
                .Columns[14].Value = aimtotal.tcnt61
                .Columns[15].Value = aimtotal.tadol91
                .Columns[16].Value = aimtotal.tndol91
                .Columns[17].Value = aimtotal.tcnt91
                .Columns[18].Value = aimtotal.tadol181
                .Columns[19].Value = aimtotal.tndol181
                .Columns[20].Value = aimtotal.tcnt181
                .Columns[21].Value = aimtotal.tadol1yr
                .Columns[22].Value = aimtotal.tndol1yr
                .Columns[23].Value = aimtotal.tcnt1yr
                .Columns[24].Value = aimtotal.tadoloct97
                .Columns[25].Value = aimtotal.tndoloct97
                .Columns[26].Value = aimtotal.tcntoct97
                .Columns[27].Value = aimtotal.totabs
                .Columns[28].Value = aimtotal.totnet
                .Columns[29].Value = aimtotal.totcnt
*!*                     .Columns[9].Value = aimtotal.tadol61
*!*                     .Columns[10].Value = aimtotal.tndol61
*!*                     .Columns[11].Value = aimtotal.tcnt61
*!*                     .Columns[12].Value = aimtotal.tadol91
*!*                     .Columns[13].Value = aimtotal.tndol91
*!*                     .Columns[14].Value = aimtotal.tcnt91
*!*                     .Columns[15].Value = aimtotal.tadol181
*!*                     .Columns[16].Value = aimtotal.tndol181
*!*                     .Columns[17].Value = aimtotal.tcnt181
*!*                     .Columns[18].Value = aimtotal.tadol1yr
*!*                     .Columns[19].Value = aimtotal.tndol1yr
*!*                     .Columns[20].Value = aimtotal.tcnt1yr
*!*                     .Columns[21].Value = aimtotal.tadoloct97
*!*                     .Columns[22].Value = aimtotal.tndoloct97
*!*                     .Columns[23].Value = aimtotal.tcntoct97
*!*                     .Columns[24].Value = aimtotal.totabs
*!*                     .Columns[25].Value = aimtotal.totnet
*!*                     .Columns[26].Value = aimtotal.totcnt
****************** end of change for new structure 2/19/04 ******************
            endwith                 && oRange

            endRow = oRange.Row
               ** track the last row number filled in for each group.
               ** This will be used to create a total line formula.

            ** move range down one row to print next detail in
            oRange = oRange.Offset(1,0)
            skip in aimtotal
          enddo                     && individual group

          ** summary line number 1
          with oRange
            .Font.Bold = .T.        && bold the summary lines

********************** change for new structure 2/19/04 **********************
            FOR i = 6 TO 29
              IF i < 27
                .Columns[i].Formula = "=sum(" + ;
                                CHR(i+64) + ;
```

```
                                ALLTRIM(STR(startRow)) + ;
                                ":" + ;
                                CHR(i+64) + ;
                                ALLTRIM(STR(endRow)) + ")"
            else
               .Columns[i].Formula = "=sum(A" + ;
                                CHR(i+38) + ;
                                ALLTRIM(STR(startRow)) + ;
                                ":A" + ;
                                CHR(i+38) + ;
                                ALLTRIM(STR(endRow)) + ")"
            endif
         next
*!*            FOR i = 6 TO 26
*!*               .Columns[i].Formula = "=sum(" + ;
*!*                                CHR(i+64) + ;
*!*                                ALLTRIM(STR(startRow)) + ;
*!*                                ":" + ;
*!*                                CHR(i+64) + ;
*!*                                ALLTRIM(STR(endRow)) + ")"
*!*            next
****************** end of change for new structure 2/19/04 *****************
      endwith                  && oRange

      endRow = oRange.Row
         ** track the last row number filled in for each group.  This
         ** will be used to create a total line formula.

      ** move range down one row to print next detail in
      oRange = oRange.Offset(1,0)
      ** end of summary line 1


      ** summary line number 2
      **       -- repeats total from summary line 1 but only in
      **       -- absolute dollar amount fields
      with oRange
         .Font.Bold = .T.          && bold the summary lines
         .Columns[3].Value = mgrp_desc

********************** change per request 4/07/04 ******************************
************combine 0-30 and 31-60 into one summary 2 total  ***********************
*!*            .Columns[6].Value = "=sum(F" + ALLTRIM(STR(endRow)) + ")"
*!*            .Columns[9].Value = "=sum(I" + ALLTRIM(STR(endRow)) + ")"
         .Columns[6].Value = "=sum(" +  ;
                        "F" + ALLTRIM(STR(endRow)) + ',' + ;
                        "I" + ALLTRIM(STR(endRow)) + ;
                            ")"
********************** end of change per request 4/07/04 ************************

********************** change for new structure 2/19/04 *********************
         .Columns[12].Value = "=sum(L" + ALLTRIM(STR(endRow)) + ")"
         .Columns[15].Value = "=sum(" +  ;
                        "O" + ALLTRIM(STR(endRow)) + ',' + ;
                        "R" + ALLTRIM(STR(endRow)) + ',' + ;
                        "U" + ALLTRIM(STR(endRow)) + ',' + ;
```

```
                        "X" + ALLTRIM(STR(endRow)) + ;
                            ")"
*!*                 .Columns[12].Value = "=sum(" +  ;
*!*                             "L" + ALLTRIM(STR(endRow)) + ',' + ;
*!*                             "O" + ALLTRIM(STR(endRow)) + ',' + ;
*!*                             "R" + ALLTRIM(STR(endRow)) + ',' + ;
*!*                             "U" + ALLTRIM(STR(endRow)) + ;
*!*                                 ")"

****************** end of change for new structure 2/19/04 ******************

            ** formulas prior to this point use the summary line 1 as
            ** the endRow formulas after this point are based upon
            ** summary line 2 figures
          endRow = oRange.Row
            ** track the last row number filled in.
            ** This will be used to create total line formulas.

********************** change per request 4/07/04 ********************************
************ grand total summary 2 total changed to reflect 0-60 summary 2 total          **
 *********** instead of 0-30 and 31-60 summary 2 totals                                   **
            .Columns[18].Value = "=sum(" + ;
                        "F" + ALLTRIM(STR(endRow)) + ',' + ;
                        "L" + ALLTRIM(STR(endRow)) + ',' + ;
                        "O" + ALLTRIM(STR(endRow)) + ;
                            ")"
*!*   ********************** change for new structure 2/19/04 **********************
*!*                 .Columns[18].Value = "=sum(" + ;
*!*                             "F" + ALLTRIM(STR(endRow)) + ',' + ;
*!*                             "I" + ALLTRIM(STR(endRow)) + ',' + ;
*!*                             "L" + ALLTRIM(STR(endRow)) + ',' + ;
*!*                             "O" + ALLTRIM(STR(endRow)) + ;
*!*                                 ")"
*!*   ********************** original before new structure below 2/19/04 **********************
*!*                 .Columns[15].Value = "=sum(" + ;
*!*                             "F" + ALLTRIM(STR(endRow)) + ',' + ;
*!*                             "I" + ALLTRIM(STR(endRow)) + ',' + ;
*!*                             "L" + ALLTRIM(STR(endRow)) + ;
*!*                                 ")"
****************** end of change for new structure 2/19/04 **********************

************* added 4/07/04 for request to produce Report 702ab **************
** this creates the file used in rpt702ab.prg
          IF mfundcat = 'F3875'
            SCATTER memvar
            m.group = mgroup
            m.m0to60 = .Columns[6].Value
            m.m61to90 = .Columns[12].Value
            m.mOver90 = .Columns[15].Value
            INSERT INTO rpt702ab FROM memvar
          endif
************* end of added for request to produce Report 702ab **************


          endwith                       && oRange
```

```
*!*   don't need this section - - endRow already reassigned during summary line 2
*!*             endRow = oRange.Row
*!*               ** track the last row number filled in for each group.
*!*               ** This will be used to create a total line formula.

          ** move range down one row to print next detail in
          oRange = oRange.Offset(1,0)
          ** end of summary line 1

          ** blank line between groups
          ** move range down one row to print next detail in
          oRange = oRange.Offset(1,0)

        ENDDO                           && end of file


      endwith                           && oSheet2

      nLastRow = oRange.Row             && Row property always give the
                                        ** first row of range.
                                        ** This range has only one row
      cLastRow = alltrim(str(nLastRow))

      with oSheet2

********************** change for new structure 2/19/04 **********************
        .Range("A1:AC1").Font.Bold = .T.          && bold the column headings
*!*             .Range("A1:Z1").Font.Bold = .T.       && bold the column headings
******************** end of change for new structure 2/19/04 ******************

        ** format the output 2 columns currency followed by on column
        ** number with 0 decimal places, if value is blank show $0.00 or 0
        .Range("F2:G" + cLastRow).NumberFormat = ;
          [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
        .Range("H2:H" + cLastRow).NumberFormat = ;
          [###,###,###,###,###;###,###,###,###,###;"0";]

        .Range("I2:J" + cLastRow).NumberFormat = ;
          [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
        .Range("K2:K" + cLastRow).NumberFormat = ;
          [###,###,###,###,###;###,###,###,###,###;"0";]

        .Range("L2:M" + cLastRow).NumberFormat = ;
          [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
        .Range("N2:N" + cLastRow).NumberFormat = ;
          [###,###,###,###,###;###,###,###,###,###;"0";]

        .Range("O2:P" + cLastRow).NumberFormat = ;
          [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
        .Range("Q2:Q" + cLastRow).NumberFormat = ;
          [###,###,###,###,###;###,###,###,###,###;"0";]

        .Range("R2:S" + cLastRow).NumberFormat = ;
          [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
        .Range("T2:T" + cLastRow).NumberFormat = ;
          [###,###,###,###,###;###,###,###,###,###;"0";]
```

```
        .Range("U2:V" + cLastRow).NumberFormat = ;
          [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
        .Range("W2:W" + cLastRow).NumberFormat = ;
          [###,###,###,###,###;###,###,###,###,###;"0";]

********************** change for new structure 2/19/04 **********************
        .Range("X2:Y" + cLastRow).NumberFormat = ;
          [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
        .Range("Z2:Z" + cLastRow).NumberFormat = ;
          [###,###,###,###,###;###,###,###,###,###;"0";]
        .Range("AA2:AB" + cLastRow).NumberFormat = ;
          [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
        .Range("AC2:AC" + cLastRow).NumberFormat = ;
          [###,###,###,###,###;###,###,###,###,###;"0";]

        ** autofit the column width
        .Range("A2:AC" + cLastRow).Columns.AutoFit()

        ** hide columns - - every third column remains showing
        FOR i = 1 TO 29
          IF MOD(i,3) != 0
            .Columns[i].entirecolumn.hidden = .T.
          endif
        next

*!*             .Range("X2:Y" + cLastRow).NumberFormat = ;
*!*               [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
*!*             .Range("Z2:Z" + cLastRow).NumberFormat = ;
*!*               [###,###,###,###,###;###,###,###,###,###;"0";]

*!*             ** autofit the column width
*!*             .Range("A2:Z" + cLastRow).Columns.AutoFit()

*!*             ** hide columns - - every third column remains showing
*!*             FOR i = 1 TO 26
*!*               IF MOD(i,3) != 0
*!*                 .Columns[i].entirecolumn.hidden = .T.
*!*               endif
*!*             next

******************** end of change for new structure 2/19/04 ******************

        ** set up the page
        with .PageSetup
          .Orientation = xlLandscape
          .FitToPagesTall = 1
          .FitToPagesWide = 1
          .CenterHorizontally = .T.
          .CenterVertically = .F.
          .TopMargin = 1.0 * autoIn2Pts
          .BottomMargin = 1.0 * autoIn2Pts
          .LeftMargin = 1.0 * autoIn2Pts
          .RightMargin = 1.0 * autoIn2Pts
          .leftfooter = "&F  &A"       && filename       sheetname
          .rightfooter = "&D"          && date
```

```
        endwith                 && Pagesetup
    endwith                     && oSheet2

** section to create yellow-red page in workbook

    oSheet3 = oBook.Worksheets.Add(,oSheet2)
        ** Add a sheet for the yellow-red page
        ** oSheet3 identifies the new sheet
        ** (,oSheet2) tells it to add the new sheet AFTER the previous
        **  sheet

    with oSheet3
      .name = "yellow-red"              && Name the sheet

**
        oSheet1 = oBook.Activesheet
          **    resets the identifier for the previous sheet
**

        .select()                       && current sheet

********************** change for new structure 2/19/04 **********************
        .range("K1").Value = "suspense: XXXXXX (noon)"
        ** assign column headings
        .Range("A1:M2").Font.Bold = .T.     && bold the column headings
        .range("D2").Value = "0-30"
        .range("E2").Value = "31-60"
        .range("F2").Value = "61-90"
        .range("G2").Value = "91-180"
        .range("H2").Value = "181-1Yr"
        .range("I2").Value = "OVER 1 YEAR to 10/1/97"
        .range("J2").Value = "OVER 10/1/97"
        .range("K2").Value = "COMMENTS"
        .range("L2").Value = "POC Contacted"
        .range("M2").Value = "date"

        ** assign column heading colors
        .range("D2").interior.color = RGB(255,202,155)      && peach
        .range("E2").interior.color = RGB(203,157,253)      && lavender
        .range("F2").interior.color = RGB(255,145,200)      && pink
        .range("G2").interior.color = RGB(255,255,128)      && yellow
        .range("H2").interior.color = RGB(183,255,183)      && mint
        .range("I2").interior.color = RGB(159,207,255)      && lt. blue
        .range("J2").interior.color = RGB(247,202,15)       && goldenrod

        ** Header Row is wider than other rows
        .ROWS("2:2").ROWHEIGHT = 47.25

        ** word wrap column headings Column D:J
        .range("D2:J2").wraptext = .T.

        ** autofit or specify the column width
        .Range("A1:C2").Columns.ColumnWidth = 8.00
        .range("D1:J2").Columns.ColumnWidth = 13.57
        .range("K1:M2").Columns.AutoFit()
```

```
*!*             .range("J1").Value = "suspense: XXXXXX (noon)"
*!*             ** assign column headings
*!*             .Range("A1:Z2").Font.Bold = .T.                 && bold the column headings
*!*             .range("D2").Value = "0-60"
*!*             .range("E2").Value = "61-90"
*!*             .range("F2").Value = "91-180"
*!*             .range("G2").Value = "181-1Yr"
*!*             .range("H2").Value = "OVER 1 YEAR to 10/1/97"
*!*             .range("I2").Value = "OVER 10/1/97"
*!*             .range("J2").Value = "COMMENTS"
*!*             .range("K2").Value = "POC Contacted"
*!*             .range("L2").Value = "date"
*!*
*!*             ** assign column heading colors
*!*             .range("D2").interior.color = RGB(203,157,253)    && lavender
*!*             .range("E2").interior.color = RGB(255,145,200)    && pink
*!*             .range("F2").interior.color = RGB(255,255,128)    && yellow
*!*             .range("G2").interior.color = RGB(183,255,183)    && mint
*!*             .range("H2").interior.color = RGB(159,207,255)    && lt. blue
*!*             .range("I2").interior.color = RGB(247,202,15)     && goldenrod
*!*
*!*             ** Header Row is wider than other rows
*!*              .ROWS("2:2").ROWHEIGHT = 47.25

*!*             ** word wrap column headings Column D:I
*!*             .range("D2:I2").wraptext = .T.
*!*
*!*             ** autofit or specify the column width
*!*             .Range("A1:C2").Columns.ColumnWidth = 8.00
*!*             .range("D1:I2").Columns.ColumnWidth = 13.57
*!*             .range("J1:L2").Columns.AutoFit()

****************** end of change for new structure 2/19/04 ******************

    endwith                                     && with oSheet3


  ******************************************************
  ******************************************************
  ** delete the original sheets ex: sheet1, sheet2, sheet3
  ** show sheet "ALL" when opening the Excel file
  with oExcel
    for i = 1 to numsheets
          ** set at original default number of sheets
      .activeworkbook.sheets(sheetname[i]).activate
      .activesheet.delete
    ENDFOR
    .activeworkbook.sheets("ALL").activate
  endwith                                       && oExcel
  ******************************************************
  ******************************************************


  ** Save the file
  ** set up the file name
```

```
   XLFile = fullpath(curdir()) + "output\" + "&mfundcat" + "&mMthYr" + ".XLS"

   XLFileJustName = juststem(XLFile)

   ** determine whether to use SaveAs or Save
   if oBook.Name <> XLFileJustName

      ** if the file already exists, delete it
      if file(XLFile)
         erase(XLFile)
      endif

      ** save it without user dialog box
      oBook.SaveAs(XLFile)

   else

      ** save it, since it's already been saved with SaveAs
      oBook.Save()
   endif


   oExcel.Quit()
   release oExcel

   SELECT aimtotal
   USE          && closes the aimtotal file so that it can be overwritten


return          && returns to calling part of program
*******************************************************************************
*******************************************************************************
**                                                                           **
**          end of PROCEDURE prepare_spreadsheet                             **
**                                                                           **
*******************************************************************************
*******************************************************************************


*******************************************************************************
*******************************************************************************
**                                                                           **
**                    clean up section                                       **
**                                                                           **
PROCEDURE close_down
**                                                                           **
*******************************************************************************
*******************************************************************************
IF USED('fundcat_list')
   SELECT fundcat_list
   USE
endif
IF USED('aimtotal')
   SELECT aimtotal
   SET ORDER TO
   USE
```

```
endif
IF USED('aimout1')
   SELECT aimout1
   SET ORDER TO
   use
ENDIF
IF USED('grp_desc')
   SELECT grp_desc
   SET ORDER TO
   use
endif

ERASE x.idx
ERASE y.idx
ERASE z.idx
ERASE fundcat_list.dbf

ERASE aimtotal.dbf

SET SAFETY on

CLEAR

*!*   MESSAGEBOX('Process is completed. Thank you.',0+64)

*!*   cancel

*************************************************************************
*************************************************************************
**                                                                     **
**              end of PROCEDURE close_down                            **
**                                                                     **
*************************************************************************
*************************************************************************
```

```
*************************************************************************
**                d:\scr44567\checkdata.PRG                            **
*************************************************************************
**      This program is called by Sar_Aim.exe                          **
**                                                                     **
** checks for errors as defined in SOP.                                **
**   1) Absolute cannot be less than net                               **
**  2) Absolute cannot be less than 0                                  **
**  3) Linenumber 6 - 13 and 72 - 74 can only have amounts in the less **
**       than 60 day columns                                           **
**  4) GetFiles.prg checks to make sure each Excel file has a sheet named **
**       "Suspense Template"                                           **
**  5) FSN not already established in OPLOCFSN table - done in checkfsn.prg **
*************************************************************************
*************************************************************************

SET SAFETY OFF
CLEAR

IF !USED('oplocfsn')
   USE oplocfsn IN 0
endif
SELECT oplocfsn
SET ORDER TO FSN   && FSN

IF !USED('submitters')
   USE submitters IN 0
endif
SELECT submitters
SET ORDER TO fsn    &&fsn

IF !USED('masterfile')
   USE masterfile IN 0
ENDIF
SELECT masterfile

SCAN
   IF TOTABS < TOTNET
      **    1) Absolute cannot be less than net
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"D")
   ENDIF

   IF TOTABS < 0
      **  2) Absolute cannot be less than 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"E")
   ENDIF

   IF totcnt > 0 AND totabs = 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"F")
   endif


   IF BETWEEN(linenum,6,13) OR BETWEEN(linenum,72,74)
```

```
** 3) Linenumber 6 - 13 and 72 - 74 can only have amounts in the less
**       than 60 day columns

    DO case
    *************
      case drdol61 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"G")

      case drcnt61 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"G")

      case crdol61 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"G")

      case crcnt61 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"G")

    *************
    *************
      case drdol91 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"H")

      case drcnt91 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"H")

      case crdol91 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"H")

      case crcnt91 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"H")

    *************
    *************
      case drdol181 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"I")

      case drcnt181 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"I")

      case crdol181 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"I")

      case crcnt181 != 0
    INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"I")
```

```
      ************
      ************
        case drdol1yr != 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"J")

        case drcnt1yr != 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"J")

        case crdol1yr != 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"J")

        case crcnt1yr != 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"J")

      ************
      ************
        case drdoloct97 != 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"K")

        case drcntoct97 != 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"K")

        case crdoloct97 != 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"K")

        case crcntoct97 != 0
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"K")
      ************

      ENDCASE
    endif

    IF masterfile.linenum = 1
      IF SEEK(masterfile.fsn,'oplocfsn','fsn')
        ** no action taken for FSN       fsn is valid to oplocfsn
        IF masterfile.oploc != oplocfsn.oploc
          ** incorrect oploc in filename
          INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename,
masterfile.linenum, "C")
        endif
      ELSE
        INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"A")
          ** message text will be FSN not in OPLOCFSN
      ENDIF

      IF SEEK(masterfile.fsn,'submitters','fsn')
```

```
      ** no action taken for FSN     fsn is valid to submitters
    else
      INSERT INTO sarerror (filename, linenum, err) VALUES (masterfile.filename, masterfile.linenum,
"B")
          ** message text will be FSN not in SUBMITTERS
    endif
  ENDIF

endscan

IF !USED('err_message')
  USE err_message IN 0
ENDIF
SELECT err_message
SET ORDER TO err

SELECT sarerror
if reccount() > 0
  SET ORDER TO filename
  SET RELATION TO err INTO err_message
  REPORT FORM frx\rptsarerror TO PRINTER NOCONSOLE
ELSE
  MESSAGEBOX('No errors were found',0+64)
endif


*****************************************************************************
** section to mark who submitted in the submitters table

IF !USED('submitters')
  USE submitters IN 0
endif
SELECT submitters
SET ORDER to fsnif                          && submitters

nAnswer = MESSAGEBOX('Do you want to empty the current month column? ',3+32)
DO case
  CASE nAnswer = 6                          && yes
    REPLACE ALL &mMth WITH "                && in submitters
  CASE nAnswer = 7                          && no
*!*       SET STEP ON
    ** no action taken
  CASE nAnswer = 2                          && cancel
    ** no action taken
endcase

IF !USED('filesrecd')
  USE filesrecd IN 0
endif
SELECT filesrecd
SET ORDER to fsnif                          && filesrecd
REPLACE ifnd WITH "IF" FOR filename = "IF"
SET RELATION TO fsn + ifnd INTO submitters
SCAN FOR !INLIST(submitters.&mMth,"L","R")
  IF NOT EOF()
    REPLACE submitters.&mMth WITH 'Y'
```

```
   ENDIF
endscan

SET SAFETY On

return
```

```
*********************************************************************************
**                    dfasall.PRG                                                **
*********************************************************************************
**      This program is called by Sar_Aim.exe                                    **
**                                                                               **
**       Makes three Excel workbooks using ONE predetermined template and        **
**      inputting total figures accordingly on the Suspense_Template             **
**      worksheet. - 1 ea. DFAS, NonDFAS, and ALL.                               **
*********************************************************************************
*********************************************************************************

set safety off
set resource off
set talk on


*!*   set procedure to program() additive
#DEFINE xlExcelVersion 33                    && Excel 4.0 (does not save multiple sheets)

IF !USED('workingfile')
   USE workingfile IN 0
ENDIF
SELECT workingfile
DELETE FOR EMPTY(totabs)
   ** if TOTal ABSolute is 0 or empty the entire line should be 0
   ** deletes header lines and report lines with no numbers
PACK

IF !USED('excel_map')
   USE excel_map IN 0 exclusive
ENDIF

IF !USED('oplocfsn')
   USE oplocfsn IN 0 order fsn
ELSE
   SELECT oplocfsn
   SET ORDER TO fsn
ENDIF

SELECT workingfile
SET RELATION TO fsn INTO oplocfsn

** templates\21all will be filled in with all lines
**      21all also gets each Suspense_Template page copied from the original
**      file into 21all after page "First" and before page "Last"
** templates\21dfas will be filled in with FSN's coded as D in OPLOCFSN field
** DFAS templates\21nondfas will be filled in with FSN's coded as N in OPLOCFSN
** field DFAS

? ''
? 'Preparing 21DFAS.XLS'
do generate_spreadsheet1 WITH "oplocfsn.dfas = 'D'", "21dfas"+mMthYr
   ** Generate the actual output spreadsheet
? ''
? 'Preparing 21NonDFAS.XLS'
do generate_spreadsheet1 WITH "oplocfsn.dfas = 'N'", "21nondfas"+mMthYr
```

```
   ** Generate the actual output spreadsheet
? ' '
? 'Preparing 21ALL.XLS - this one takes a little longer'
do generate_spreadsheet1 WITH ".t.", "21all"+mMthYr
   ** Generate the actual output spreadsheet

RETURN


*****
*
* section to create formal output to Excel
*
PROCEDURE generate_spreadsheet1
PARAMETERS pForCond, pOutName
   ************************* output to Excel ******************************
   ************************************************************************
   ** adds the amounts on each individual excel sheet to the appropriate
   ** summary file


   #DEFINE xlLandscape 2                    && defines page orientation
   #DEFINE autoIn2Pts 72


   local k, oBook, oRange, oSheet, lCell

   oExcel = createobject("Excel.application")
            ** creates the application, opens an Excel session without
            ** opening a excel file

   oExcel.visible = .f.                     && do not show Excel spreadsheet

   oExcel.application.displayalerts = .f.   && turns off the excel warning
                                             ** message boxes
   oExcel.Workbooks.Open(FULLPATH(CURDIR()+"templates\21dfas.xls"))
   oExcel.Sheets("Suspense_Template").Select
   SCAN FOR &pForCond                       && scan through all records matching the
                                            ** for condition (dfas, nondfas, all)

      SELECT excel_map
      SCAN  && scan through fieldname to column conversion
         k = "workingfile." + field_name
         IF &k != 0
            oExcel.activesheet.cells(workingfile.linenum,col_num).Value = ;
               oExcel.activesheet.cells(workingfile.linenum,col_num).Value + &k
         endif
      endscan
   endscan



*!*      oExcel.visible = .t.             && do not show Excel spreadsheet

   ** if the file already exists, delete it
   pOutName = FULLPATH(CURDIR()+'output\'+pOutName)
```

```
if file(pOutName)
   erase(pOutName)
endif

** save it without user dialog box
oExcel.activeworkbook.SaveAs(pOutName)
oExcel.Quit()
release oExcel
```

```
*********************************************************************************
**              d:\scr44567\getfiles.prg.PRG                                   **
*********************************************************************************
**      This program is called by Sar_Aim.exe                                  **
**        determine how many excel files there are and append the              **
**        data from each                                                       **
*********************************************************************************
*********************************************************************************


set safety off
set resource off
set talk on
set procedure to program() additive
#DEFINE xlExcelVersion 33                    && Excel 4.0 (does not save multiple sheets)

IF !USED('masterfile')
  use masterfile IN 0
ENDIF

SELECT masterfile
zap

do how_many_files                            && Determine how many files are being used this
                                               ** month, this will be used as our loop counter

IF numfile != 0
  do mainbody                                && Saves the Excel files to the proper version and
                                              ** appends the data into the table
                                              ** replaces filename and linenum fields

  DO assign_fundcat                          && procedure to replace fundcat field

  REPLACE fundcat WITH " FOR linenum < 5     && in masterfile
    ** these are header lines in the Excel file. The fundcat filed in is
    ** from the previous fsn report.  These lines are not used further.
    ** If they are marked with a fundcat, other processes will try and
    ** use them.

  DO assign_fsn_month_oploc

  USE filesrecd IN 0
  SELECT filesrecd
  ZAP
  APPEND FROM masterfile FOR linenum = 1
  use
ENDIF


*********************************************************************************
*********************************************************************************
**                                                                             **
** Count how many excel files are in the working directory with the current    **
** month as part of the file name.                                             **
** These are the files that will be used for the rest of the process           **
**                                                                             **
```

```
PROCEDURE how_many_files
**    uses masterfile.dbf                                                    **
*****************************************************************************
*****************************************************************************

   PUBLIC mfileloc, numfile, mDirLoc, temp, listfile
   DECLARE listfile[1]

   mDirLoc = sys(5) + sys(2003) + '\datafile\'
   mFileLoc = mDirLoc + '*' + upper(mMthYr) + '.xls'

      ** in this instance the asterick is a multi-char wildcard
   numfile = adir(listfile, '&mfileloc')
      ** this creates an array of current cycle filenames and returns a
      ** count of the files found with MMMYY in the file name
   if numfile = 0
      * No files were found, so there's really nothing to do.
      messagebox('There were no files found for ' + upper(mMthYr) + ', returning to main menu' ,0+16)
      return
   endif

   temp = sys(2023) + '\' + sys(2015) + ".xls"
      ** Create a single temp file that will be re-used over and over again
      ** sys(2023) returns system defined path for tempfiles
      ** sys(2015) returns a unique 10 char procedure (or file) name
      ** starting with an underscore
return
*****************************************************************************
*****************************************************************************
**                                                                          **
**              end of PROCEDURE how_many_files                             **
**                                                                          **
*****************************************************************************
*****************************************************************************


*****************************************************************************
*****************************************************************************
**                                                                          **
** Open an excel session and call a procedure to save and append the excel  **
** files. This part was nested as another Procedure trying to avoid C000005 **
** error by allowing FoxPro to "catch up" before returning back to the the  **
** calling Procedure                                                        **
**                                                                          **
PROCEDURE mainbody
**    uses masterfile.dbf                                                    **
*****************************************************************************
*****************************************************************************
   loExcel = CreateObject("Excel.application")
      ** Instantiate the excel application in the background (it will carry
      ** out the processing for us). This opens Excel itself but does not
      ** open any files.
   clear                 && Clear the screen because this algorithm
                        ** reports its progress as it goes
   start = time()             && Begin at whatever time (it's known through
                         ** debugging that this takes several minutes)
```

```
   for i = 1 to numfile
      do load_spreadsheets    && Load in each of the spreadsheets in the
                        **specified directory
   endfor
   end = time()              && Grab the ending time
   ? 'Ending append section  - start ', start, '    end - ', end
      ** Display the start and ending times (just as a reference)

   release loExcel       && Release the excel instance
   erase "&temp"         && this file is no longer needed
return
*************************************************************************
*************************************************************************
**                                                                     **
**              end of PROCEDURE mainbody                              **
**                                                                     **
*************************************************************************
*************************************************************************


*************************************************************************
*************************************************************************
**                                                                     **
** loop to save each Excel source file as the correct version          **
** (using Excel 4.0 because it only saves one sheet), then append the  **
** Excel file into the table                                           **
**                                                                     **
PROCEDURE load_spreadsheets
** this is being done in a loop called from mainbody procedure, one time **
** for each file being used                                            **
**    uses masterfile.dbf                                              **
*************************************************************************
*************************************************************************
   mfilename = alltrim(listfile(i,1))
   ?str(i)+' of '+alltrim(str(numfile))+'   '+mfilename+'     '+time()
      ** statement displays on the screen where we are at in the loop and
      ** the time we got to each file as a reference point

   if file("&temp")
      erase "&temp"       && The single temp file already exists.  Delete it
                     ** and reuse it again
   endif

   WITH loExcel
      .visible = .F.
      .application.displayalerts = .F.
                              ** to stop excel message box
      .application.workbooks.open("&mDirLoc&mfilename")
                              ** opens individual workbook


   numsheets = .activeworkbook.sheets.count
           ** see how many sheets are in a new file

   ************************************************
   ** get the sheet names for the default number of sheets to delete later
```

```
declare sheetname(numsheets)          && declares array for sheetnames

   k = .f.
   for j = 1 to numsheets
      .activeworkbook.sheets(j).activate
      sheetname[j] = .activesheet.name
         ** populates array with sheetnames
      IF sheetname[j] = "Suspense_Template"
         k = .t.
      endif
   endfor
***************************************************


   IF k = .f.
      filename = mfilename
      ** Sheet Name Invalid
      INSERT INTO sarerror (filename, linenum, err) VALUES (mfilename, 0, "L")
       .workbooks.close()                          && Done twice to try to get
       .workbooks.close()                          && rid of the C0000005 errors
   ENDIF

   IF k = .t.
      .activeworkbook.sheets("Suspense_Template").activate
                  ** finds sheet by name
       .activesheet.saveas("&temp", xlExcelVersion)
       .workbooks.close()                          && Done twice to try to get
       .workbooks.close()                          && rid of the C0000005 errors
                                                    ** save as version 4.0
   ENDIF

   ENDWITH

   IF k = .t.
      select masterfile                            && main working table
      startnum = RECCOUNT()
      append from "&temp" xls
*!*      replace filename with mfilename, linenum WITH RECNO() - startnum for empty(filename)
      replace filename with mfilename, linenum WITH RECNO() - startnum for empty(filename)
            ** Add the filename to each of the recently added records.
            ** This statement must stay here - it is the only time the
            ** information is available
   ENDIF

return
*************************************************************************
*************************************************************************
**                                                                     **
**          end of PROCEDURE load_spreadsheets                         **
**                                                                     **
*************************************************************************
*************************************************************************


*************************************************************************
*************************************************************************
```

```
**                                                                              **
** assign a fund category to each line                                          **
**                                                                              **
PROCEDURE assign_fundcat
**    uses masterfile.dbf                                                       **
********************************************************************************
********************************************************************************
  **********
  ** Right now, the files have all been loaded.  Probably more than 100 of
  ** them were processed.  Now that we have our data loaded, we can go
  ** through our files and process them.
  **
  ** Check each line and fill in the FUNDCAT according to the fndsymbl for
  ** that section on the Excel report.  The section header line in excel
  ** contains the fund symbol followed by "TOTAL" in the FNDSYMBL field
  ** but the associated lines do not contain a fund symbol.  Therefore look
  ** for the word "TOTAL" and assign the FUNDCAT based on the last line that
  ** contained the word "TOTAL"

  USE masterfile
  mFundcat = "header"                 && needs to be initialized to something and
                                      ** lines before the first FNDSYMBL will not be
                                      ** needed
  scan                                && ensures start with first record
    if "TOTAL" $ fndsymbl             && looks for the word TOTAL within field
                                      ** fndsymbl and resets mFundcat if found
      mFundcat = strtran(fndsymbl, " TOTAL", "")
        ** replaces the word TOTAL with no spaces and saves the new
        ** value as mFundcat   example: the first one is "F3875 TOTAL"
        ** so "F3875" is saved as mFundcat
    endif
    replace fundcat with mFundcat
  endscan
return
********************************************************************************
********************************************************************************
**                                                                              **
**              end of PROCEDURE assign_fundcat                                 **
**                                                                              **
********************************************************************************
********************************************************************************


********************************************************************************
********************************************************************************
**                                                                              **
** assign a the month, fsn, and oploc to each line                              **
**                                                                              **
PROCEDURE assign_fsn_month_oploc
**    uses masterfile.dbf                                                       **
********************************************************************************
********************************************************************************
  ********** incorporate the into scan section outside of loop ************
  ** need 6 character fsn to compare with MACOM table.  DFAS filenames were
  ** assigned 6 character fsn in previous section.  All other filenames are
  ** still 5 character fsn.
```

```
********** incorporate the into scan section outside of loop ***********


scan      && Scan through the remaining detail lines and assign the FSN,
          ** month and OPLOC
  scatter memvar
          ** next three lines broke apart for ease in reading
  mFiletrim = ALLTRIM(filename)
          ** eliminates leading and trailing spaces from the filename
  mFile9 = RIGHT(mFiletrim, 9)
          ** grab the last 9 characters of the filename are always
          ** MMMYY.XLS
  mMthNum = atc(mFile9, filename) - 1
          ** The "-1" is because the atc() will find the character
          ** *AFTER* the string we're looking for.  mMthNum is the
          ** starting location for the month from the leftmost character
          ** of mfilename

  ** month section
  m.month = UPPER(mMthYr)
          ** the input they specified (above) goes to the table value

  ** fsn section
  m.fsn_prefix = space(0)

do case
 case left(filename, 2) = "EX"
     ** filenames starting with EX have variable length FSN
  ** ex: EXNAVYMAR03.XLS or EXSTATEMAR03.XLS
  ** for this purpose NAVY or STATE is the fsn
  ** know fsn starts in 3rd character and ends 1 short of month
  m.fsn = upper(substr(FILENAME, 3, (mMthNum-2)))
  IF m.fsn = '88888'
    m.fsn_prefix = '0'
  ENDIF

 case left(filename, 4) = "DFAS"
  ** oploc is 4 characters and fsn is all characters between oploc
  ** and month - usually 6 characters
  m.fsn = SUBSTR(FILENAME, 5, (mMthNum-4))

 case left(filename, 2) = "IF"
  ** INTERFUND files are named "IF" followed by the 2-character oploc
  ** letter portion of filename is 4 characters
  ** fsn is all characters between oploc
  ** and month - usually 6 characters
  m.fsn = SUBSTR(FILENAME, 5, (mMthNum-4))
  m.fsn_prefix = "0"

 otherwise
  ** regular lines - 2 char oploc, 5 char fsn
  m.fsn = substr(filename, 3, 5)
  m.fsn_prefix = "0"
endcase

* Populate the oploc with the appropriate portion
```

```
   IF LEFT(filename, 2) = "IF"
      m.oploc = substr(filename, 3, atc(alltrim(m.fsn), filename) - 3)
      m.fsn = m.fsn_prefix + m.fsn
   else
      m.oploc = substr(filename, 1, atc(alltrim(m.fsn), filename) - 1)
      m.fsn = m.fsn_prefix + m.fsn
   ENDIF
      * All done, re-populate the line
      gather memvar
   endscan
return
*******************************************************************************
*******************************************************************************
**                                                                           **
**              end of PROCEDURE assign_fsn_month_oploc                      **
**                                                                           **
*******************************************************************************
*******************************************************************************
```

```
*************************************************************************
**                    high4.PRG                                        **
*************************************************************************
**      This program is called by Sar_Aim.exe                          **
**                                                                     **
**       keep track of the total lines from the 3875 report for fundcat **
**      F3875 for each month so that month to month comparisons can be done. **
**       need the 4th total figure from summary line 2 for each group  **
**      (should be sum of Total Absolute for all age group columns)    **
**       need to compute increase/decrease since last month as a dollar **
**      figure and a percentage                                        **
*************************************************************************
*************************************************************************

SET TALK OFF

SET SAFETY Off

? 'High 4 report'

** determine prior month
** mNumMth is set during main.prg, on init and on click in optiongroup1
** mNumMth is the number for month used


mMthPrior = SUBSTR(CMONTH(GOMONTH(ctod(ALLTRIM(mNumMth)+'/01/'+mYr),-1)),1,3)
** first 3 characters of the spelled out month that is one day less than the first day of the month and year set


IF !USED('aimhigh4')
   USE aimhigh4 IN 0
endif
SELECT aimhigh4
INDEX ON group TO x


if !USED('aimout1')
   USE aimout1    IN 0
ENDIF
SELECT aimout1                          && filled in aimoutput2.prg
INDEX ON group TO y
TOTAL ON group TO temptotal FOR note != 'EX' AND fundcat = 'F3875'

USE temptotal IN 0
SELECT temptotal
DELETE FOR group = 'DFA'                && DFA is DFAS Psuedo FSNs, these are not included in the
high 4 report
PACK
INDEX ON group TO z


SELECT aimhigh4
SCAN

   lcgroup = group                      && lcgroup is local character group
```

```
********************** change for new structure 2/19/04 **********************
  IF SEEK(lcgroup, 'temptotal')
    replace aimhigh4.&mMth WITH temptotal.tadol0 + temptotal.tadol31 ;
                        + temptotal.tadol61 + temptotal.tadol91 ;
                        + temptotal.tadol181 + temptotal.tadol1yr ;
                        + temptotal.tadoloct97
  endif
*!*     IF SEEK(lcgroup, 'temptotal')
*!*       replace aimhigh4.&mMth WITH temptotal.tadol0 + temptotal.tadol61 + temptotal.tadol91 + ;
*!*                           temptotal.tadol181 + temptotal.tadol1yr + temptotal.tadoloct97
*!*     endif
******************* end of change for new structure 2/19/04 ******************

endscan


** determine high 4 values
REPLACE ALL ranking WITH ''
REPLACE ALL change WITH &mMthPrior - &mMth

calculate MAX(change) TO tempvalue
REPLACE ranking WITH '1' FOR change = tempvalue
calculate MAX(change) TO tempvalue FOR EMPTY(ranking)
REPLACE ranking WITH '2' FOR change = tempvalue
calculate MAX(change) TO tempvalue FOR EMPTY(ranking)
REPLACE ranking WITH '3' FOR change = tempvalue
calculate MAX(change) TO tempvalue FOR EMPTY(ranking)
REPLACE ranking WITH '4' FOR change = tempvalue
REPLACE ranking WITH '5' FOR EMPTY(ranking)

*!* can't get this to work
*!*   FOR i = 1 TO 4
*!*     calculate MAX(change) TO tempvalue FOR EMPTY(ranking)
*!*     REPLACE ranking WITH STR(i) FOR change = tempvalue
*!*   endfor


INDEX ON ranking TO w
TOTAL ON ranking TO ranktotal

************************** works to here *********************************

************************** output to Excel ******************************
************************************************************************
** want one workbook with one worksheet listing detail lines by group
** followed by summary lines for the group, then the next group

** uses aimtotal.dbf


  #DEFINE xlLandscape 2                 && defines page orientation
  #DEFINE autoIn2Pts 72

  IF !USED('grp_desc')
    use GRP_DESC IN 0
```

```
ENDIF
SELECT grp_desc
INDEX ON group TO u

use ranktotal IN 0
SELECT ranktotal
REPLACE group WITH 'Oth' FOR ranking = '5'
INDEX ON ranking TO v
*!*     SET RELATION TO group INTO grp_desc


local oBook, oRange, oSheet
mGroup = "  "                          && initialize
mgrp_desc = "       "                  && initialize

oExcel = createobject("Excel.application")
        ** creates the application, opens an Excel session without
        ** opening a excel file

*****************************
oExcel.visible = .f.                   && do not show Excel spreadsheet
*!*     oExcel.visible = .T.             && show Excel spreadsheet as it is being created
*****************************
oExcel.application.displayalerts = .f.  && turns off the excel warning
                                        ** message boxes

oBook = oExcel.workbooks.add()          && creates an actual workbook

oSheet = oBook.Activesheet              && opens book with the default
                                        ** number of worksheets

numsheets = oExcel.activeworkbook.sheets.count
        ** see how many sheets are in a new file

oSheet1 = oBook.Worksheets[numsheets]
        ** defines the last sheet of the workbook if opened with the
        ** default number of sheets.  used to keep track of the last
        ** sheet worked     and delete original blank sheets later


************************************************
** get the sheet names for the default number of sheets to delete later
declare sheetname(numsheets)          && declares array for sheetnames

with oExcel
  for i = 1 to numsheets
    .activeworkbook.sheets(i).activate
    sheetname[i] = .activesheet.name
      ** populates array with sheetnames
  endfor
endwith                               && oExcel
************************************************

  oSheet2 = oBook.Worksheets.Add(,oSheet1)
        ** Add a sheet for the detail report
        ** oSheet2 identifies the new sheet
```

```
        ** (,oSheet1) tells it to add the new sheet AFTER the previous
        **  sheet

    with oSheet2
      .name = "ALL"                    && Name the sheet

**
      oSheet1 = oBook.Activesheet
        ** resets the identifier for the previous sheet
**

      .select()                        && current sheet

      ** assign column headings
      .Range("A1:F2").Font.Bold = .T.          && bold the column headings
      .range("B1").Value = "PREVIOUS PERIOD RESULTS"
      .range("C1").Value = "CURRENT PERIOD RESULTS"
      .range("D1").Value = "AMOUNT INCREASE / DECREASE"
      .range("E1").Value = "PERCENTAGE OF TOTAL CHANGE"
      .range("F1").Value = "RANKING"
      .range("A2").Value = "GROUP"
      .range("B2").Value = "&mMthPrior"
      .range("C2").Value = "&mMth"


      ** Header Row is wider than other rows
       .ROWS("1:1").ROWHEIGHT = 47.25
       .range("B1:e1").wraptext = .t.

            ** detail starts printing on next line after headings
      oRange = .Range("A3:F3")
            ** sets the range to the first detail line to be filled in


      do while not eof()

*!*             mGroup = ranktotal.group
*!*               **  used to keep track of the section being worked
*!*             IF SEEK(mGroup,'GRP_DESC')
*!*               mgrp_desc = GRP_DESC.desc
*!*             ELSE
*!*  *!*                m.group = '5'
*!*               mgrp_desc = "All Others"
*!*             endif

        startRow = oRange.Row
          **  Row property always give the first row of range.
        endRow = oRange.Row

        DO WHILE NOT EOF()
          with oRange
            ** fill in detail lines

            .Columns[1].Value = ranktotal.group
            .Columns[2].Value = ranktotal.&mMthPrior
            .Columns[3].Value = ranktotal.&mMth
```

```
            .Columns[4].Value = ranktotal.change
            .Columns[5].Value = ranktotal.&mMthPrior/ranktotal.&mMth
            .Columns[6].Value = ranktotal.ranking
         endwith                    && oRange

         endRow = oRange.Row
            ** track the last row number filled in for each group.
            ** This will be used to create a total line formula.

         ** move range down one row to print next detail in
         oRange = oRange.Offset(1,0)
         skip in ranktotal
      ENDDO


      endRow = oRange.Row
         ** track the last row number filled in for each group.  This
         ** will be used to create a total line formula.

   ENDDO                              && end of file


endwith                               && oSheet2

nLastRow = oRange.Row                 && Row property always give the
                                       ** first row of range.
                                       ** This range has only one row
cLastRow = alltrim(str(nLastRow))

with oSheet2

   .Range("A1:Z1").Font.Bold = .T.          && bold the column headings

   ** format the output 2 columns currency followed by on column
   ** number with 0 decimal places, if value is blank show $0.00 or 0
   .Range("b2:d" + cLastRow).NumberFormat = ;
      [$##,###,###,###,###.##;$###,###,###,###,###.##;"$0.00";]
      ** dollar amount fields

   .Range("e2:e" + cLastRow).NumberFormat = ;
      [###.####]
      ** 4 digit decimal - percentage fields

   .Range("f2:f" + cLastRow).NumberFormat = ;
      [###,###,###,###,###;###,###,###,###,###;"0";]
      ** integer field

   ** autofit the column width
   .range("e1:e1").columns.columnwidth = 13.25
   .range("f1:f1").columns.autofit()
   .Range("A2:d" + cLastRow).Columns.AutoFit()


   ** set up the page
   with .PageSetup
      .Orientation = xlLandscape
```

```
        .FitToPagesTall = 1
        .FitToPagesWide = 1
        .CenterHorizontally = .T.
        .CenterVertically = .F.
        .TopMargin = 1.0 * autoIn2Pts
        .BottomMargin = 1.0 * autoIn2Pts
        .LeftMargin = 1.0 * autoIn2Pts
        .RightMargin = 1.0 * autoIn2Pts
        .leftfooter = "&F  &A"                  && filename      sheetname
        .rightfooter = "&D"                     && date
      endwith                                   && Pagesetup
    endwith                                     && oSheet2


*********************************************************
*********************************************************
** delete the original sheets ex: sheet1, sheet2, sheet3
** show sheet "ALL" when opening the Excel file
with oExcel
   for i = 1 to numsheets
         ** set at original default number of sheets
      .activeworkbook.sheets(sheetname[i]).activate
      .activesheet.delete
   ENDFOR
   .activeworkbook.sheets("ALL").activate
endwith                                         && oExcel
*********************************************************
*********************************************************




** Save the file
** set up the file name

XLFile = fullpath(curdir()) + "output\ranking" + "&mMthYr" + ".XLS"

XLFileJustName = juststem(XLFile)

** determine whether to use SaveAs or Save
if oBook.Name <> XLFileJustName

   ** if the file already exists, delete it
   if file(XLFile)
      erase(XLFile)
   endif

   ** save it without user dialog box
   oBook.SaveAs(XLFile)

else

   ** save it, since it's already been saved with SaveAs
   oBook.Save()
endif
```

```
   oExcel.Quit()
   release oExcel


*************************************************************************
*************************************************************************
**                                                                     **
**            end of PROCEDURE prepare_spreadsheet                     **
**                                                                     **
*************************************************************************
*************************************************************************

******* close tables
IF USED('ranktotal')
   SELECT ranktotal
   SET ORDER TO
   USE
ENDIF
IF USED('temptotal')
   SELECT temptotal
   SET ORDER TO
   USE
ENDIF
IF USED('aimhigh4')
   SELECT aimhigh4
   SET ORDER TO
   USE
ENDIF
IF USED('aimout1')
   SELECT aimout1
   SET ORDER TO
   USE
ENDIF
IF USED('grp_desc')
   SELECT grp_desc
   SET ORDER TO
   USE
endif
CLOSE INDEXES

******* erase tables

ERASE ranktotal.dbf
ERASE temptotal.dbf
ERASE u.idx
ERASE v.idx
erase w.idx
ERASE x.idx
erase y.idx
ERASE z.idx

SET TALK ON

SET SAFETY ON
```

```
*!*   return                    && returns to calling part of program
```

```
*************************************************************************
**                    MARK_SUBMIT.PRG                                  **
*************************************************************************
**      This program is called by Sar_Aim.exe                          **
**                                                                     **
**      this program marks the submitters file a Y-received or         **
**      N-not Submitted    for the month                               **
*************************************************************************
*************************************************************************

IF !USED('filesrecd')
  USE filesrecd IN 0
ENDIF
SELECT filesrecd
REPLACE ifnd WITH "IF" FOR filename = "IF"

***********************************************
IF !USED('submitters')
  USE submitters IN 0
endif
SELECT submitters
SET ORDER TO fsn                              &&fsn
nAnswer = MESSAGEBOX('Do you want to mark the NonSubmitters? ',3+32)
DO case
  CASE nAnswer = 6                            && yes
    **   mark the yes lines and then mark all remaining empty lines with no

    **    mark who submitted in the submitters table
    IF !USED('filesrecd')
      USE filesrecd IN 0
    endif
    SELECT filesrecd
    SET ORDER to fsnif                        && filesrecd

    SELECT submitters
    SET ORDER to fsnif                        && submitters

    SELECT filesrecd
    SET RELATION TO fsn + ifnd INTO submitters
    SELECT filesrecd
    SCAN FOR !INLIST(submitters.&mMth,"L","R")
      IF NOT EOF()
        REPLACE submitters.&mMth WITH 'Y'
      ENDIF
    ENDSCAN

    SELECT submitters
    replace &mMth WITH 'N' FOR EMPTY(&mMth)


  CASE nAnswer = 7                            && no  -  mark the yes lines only

    ** mark who submitted in the submitters table
    IF !USED('filesrecd')
      USE filesrecd IN 0
    endif
```

```
      SELECT filesrecd
      SET ORDER to fsnif                    && filesrecd


      SELECT submitters
      SET ORDER to fsnif                    && submitters

      SELECT filesrecd
      SET RELATION TO fsn + ifnd INTO submitters
      SELECT filesrecd
      SCAN FOR !INLIST(submitters.&mMth,"L","R")
        IF NOT EOF()
          REPLACE submitters.&mMth WITH 'Y'
        ENDIF
      ENDSCAN


    CASE nAnswer = 2                        && cancel
      ** no action taken
endcase

return
```

```
*************************************************************************
**                    MasterCopy.PRG                                   **
*************************************************************************
**      This program is called by Sar_Aim.exe                          **
**                                                                     **
**      It is used to create a working copy of the master file         **
**      This way there is always a copy intact with all original       **
**    information and multiple file structures are not needed          **
*************************************************************************
*************************************************************************
SET SAFETY OFF

IF USED('workingfile')                  && close the old version of workingfile
   SELECT workingfile
   USE
ENDIF

IF !USED('masterfile')
   USE masterfile IN 0
ENDIF
SELECT masterfile


COPY TO workingfile                     && will overwrite any existing copies of the workingfile

use                                     && closes the masterfile

USE workingfile IN 0                    && uses new version of workingfile
SELECT workingfile

SET SAFETY on
```

```
*********************************************************************************
**                    movefiles.PRG                                              **
*********************************************************************************
**       This program is called by Sar_Aim.exe                                   **
**                                                                               **
**       Moves submitted Excel files to their respective directories             **
**       (DFAS or NonDFAS)                                                       **
*********************************************************************************
*********************************************************************************

set safety off
set resource off
set talk on


*******************************

? ' '
? 'Copying Excel files to DFAS or NONDFAS directories'
SELECT workingfile
INDEX ON filename UNIQUE TO m
SCAN
   IF oplocfsn.dfas = 'D'
      COPY file 'datafile\' + UPPER(ALLTRIM(filename)) to ;
         'datafile\dfas\' + ALLTRIM(filename)
   ENDIF

   IF oplocfsn.dfas = 'N'
      COPY file 'datafile\' + UPPER(alltrim(filename)) to ;
         'datafile\nondfas\' + ALLTRIM(filename)
   endif
ENDSCAN

********** clean up
IF USED('workingfile')
   SELECT workingfile
   USE
endif
ERASE workingfile.dbf

CLOSE INDEXES
ERASE m.idx
*******************************

RETURN
```

```
**pmi543.prg

SET SAFETY OFF

SELECT sum(tadol0) as tadol0, ;
      sum(tadol31) as tadol31,;
      sum(tadol61) as tadol61,;
      sum(tadol91) as tadol91,;
      sum(tadol181) as tadol181,;
      sum(tadol1yr) as tadol1yr,;
      sum(tadoloct97) as tadoloct97,;
      linenum ;
   FROM masterfile ;
   WHERE INLIST(linenum,5,48,56,59,62,64,69,71,53);
   GROUP BY linenum;
   INTO dbf pmi543

IF !USED('excel_map_543')
   USE excel_map_543 IN 0 order linenum
ELSE
   SELECT excel_map_543
   SET ORDER TO linenum
ENDIF


SELECT pmi543
INDEX ON linenum TO idx543

SET RELATION TO linenum INTO excel_map_543

#DEFINE xlLandscape 2                          && defines page orientation
#DEFINE autoIn2Pts 72


oExcel = createobject("Excel.application")
        ** creates the application, opens an Excel session without
        ** opening a excel file

*!*    oExcel.visible = .t.                    && do not show Excel spreadsheet
   oExcel.visible = .f.                        && do not show Excel spreadsheet

oExcel.application.displayalerts = .f.         && turns off the excel warning
                                               ** message boxes
oExcel.Workbooks.Open(FULLPATH(CURDIR()+"templates\pmi543.xls"))
oExcel.Sheets("Sheet1").Select
oExcel.activesheet.cells(25,4).Value = mMthYr

GO top
DO WHILE !EOF()
   oExcel.activesheet.cells(excel_map_543.excel_row,2).Value = pmi543.tadol0
   oExcel.activesheet.cells(excel_map_543.excel_row,4).Value = pmi543.tadol31
   oExcel.activesheet.cells(excel_map_543.excel_row,6).Value = pmi543.tadol61
   oExcel.activesheet.cells(excel_map_543.excel_row,8).Value = pmi543.tadol91
   oExcel.activesheet.cells(excel_map_543.excel_row,10).Value = pmi543.tadol181
   oExcel.activesheet.cells(excel_map_543.excel_row,12).Value = pmi543.tadol1yr
   oExcel.activesheet.cells(excel_map_543.excel_row,14).Value = pmi543.tadoloct97
```

```
   SKIP

enddo

** if the file already exists, delete it
pOutName = FULLPATH(CURDIR()+'output\pmi543')
if file(pOutName)
   erase(pOutName)
endif

** save it without user dialog box
oExcel.activeworkbook.SaveAs(pOutName)
oExcel.Quit()
release oExcel


IF USED('pmi543')
   USE                                          && to close pmi543
ENDIF

*!*   ERASE pmi543.dbf
*!*   ERASE idx543.idx
```

```
** rpt702ab.prg

SET SAFETY OFF

IF !USED('rpt702ab')
   USE rpt702ab IN 0
ENDIF
SELECT rpt702ab

IF !USED('grp_desc')
   USE grp_desc IN 0 order group
ELSE
   SELECT grp_desc
   SET ORDER TO group
ENDIF

SELECT rpt702ab
SET RELATION TO group INTO grp_desc

#DEFINE xlLandscape 2                    && defines page orientation
#DEFINE autoIn2Pts 72

oExcel = createobject("Excel.application")
        ** creates the application, opens an Excel session without
        ** opening a excel file

*!*    oExcel.visible = .t.              && do not show Excel spreadsheet
   oExcel.visible = .f.                  && do not show Excel spreadsheet

oExcel.application.displayalerts = .f.   && turns off the excel warning
                                          ** message boxes
oExcel.Workbooks.Open(FULLPATH(CURDIR()+"templates\Report702ab.xls"))
oExcel.Sheets("702ab").Select
oExcel.activesheet.cells(4,2).Value = mMthYr

GO top

DO WHILE !EOF()
   oExcel.activesheet.cells(grp_desc.rownbr702,2).Value = rpt702ab.m0to60
   oExcel.activesheet.cells(grp_desc.rownbr702,3).Value = rpt702ab.m61to90
   oExcel.activesheet.cells(grp_desc.rownbr702,4).Value = rpt702ab.mOver90
   SKIP
enddo

** if the file already exists, delete it
pOutName = FULLPATH(CURDIR()+'output\rpt702ab')
if file(pOutName)
   erase(pOutName)
endif

** save it without user dialog box
oExcel.activeworkbook.SaveAs(pOutName)
oExcel.Quit()
release oExcel

*!*    IF !USED('rpt702ab')
```

```
*!*     USE rpt702ab IN 0
*!*   ENDIF
*!*     SELECT rpt702ab
*!*     zap
```

```
*********************************************************************************
**                      SarMacom.PRG                                           **
*********************************************************************************
**      This program is called by Sar_Aim.exe                                  **
**                                                                             **
**       This program appends numerous excel sheets submitted by the field     **
**      Then assigns a macom according to v:\shared\sf\er\tables\opfsn         **
**      and summarizes by macom.  Excel output is then created                 **
*********************************************************************************
*********************************************************************************

set safety off
set resource off
set talk on
set procedure to program() additive
#DEFINE xlExcelVersion 33    && Excel 4.0 (does not save multiple sheets)


? ' '
? 'Working the SAR by Macom report'

do clean_house
     ** Remove the unwanted/unnecessary lines for sarmacom process


** Rearranged the order of next two commands to allow for file name changes
** to MasterFile and WorkingFile instead of SarLines, DFASall, and AIM_INfile
DO check_map

DO assign_macom


do total_for_output                    && Total everything and produce an interim output
                                       ** file
do generate_spreadsheet                && Generate the actual output spreadsheet
do close_down                          && close everything and erase extra files


*****
*
* To delete un-needed lines
*
PROCEDURE clean_house                  && uses workingfile
  IF !USED('workingfile')
     USE workingfile IN 0
  ENDIF
  SELECT workingfile

  delete for !inlist(validity, "PROPER", "IMPROPER")
     ** Lines that do not contain the words "PROPER" or "IMPROPER" in the
     ** validity field are header lines or prior total lines.  We do not
     ** need these lines.  This delete must be after the fundcat assignment
     ** section because the line used to determine fundcat is a total line
     ** and will be deleted.
  delete for upper(DESC) = "TOTAL TRUE SUSPENSE"
     ** Lines with desc "Total True Suspense" is a subtotal line.  The
```

```
      ** individual lines are included in other totals.  Do not want to
      ** include the lines AND the subtotal line.
   pack
return


*****
*
* To check and see if a computer is mapped to V:\SF\ or V:\SHARED\SF and copy
* the LAN OPFSN.dbf for use in assigning macom
*
PROCEDURE check_map
*!*   Original code
*!*      if file('V:\SHARED\SF\ER\tables\opfsn.dbf')
*!*         copy file V:\SHARED\SF\ER\tables\opfsn.dbf to c:\temp\macomfile.dbf
*!*      else
*!*         if file('V:\SF\ER\tables\opfsn.dbf')
*!*            copy file V:\SF\ER\tables\opfsn.dbf to c:\temp\macomfile.dbf
*!*         endif
*!*      ENDIF
*!*   Original code

   copy file FULLPATH(CURDIR())+'\V_SF_ER_tables\opfsn.dbf' to c:\temp\macomfile.dbf

return


*****
*
* to assign the macom based on matching fsn to OPFSN.dbf
*
PROCEDURE assign_macom

   use c:\temp\macomfile in 0
   select macomfile
   index on fsn to m4

   IF !USED('workingfile')
      use workingfile in 0
   endif
   select workingfile
   index on fsn to m5

   scan
      lcfsn = fsn       && lcfsn is local character fsn
      if seek(lcfsn, 'macomfile')
         replace macom with macomfile.macom
      ELSE
         replace macom with 'XX'    && invalid fsn
      endif
   endscan


return
```

```
*****
*
* create a file containing total lines for each macom+note+fundcat.  The final
* output spreadsheets will only contain total lines.
*
PROCEDURE total_for_output          && uses workingfile
   ****************** sum files by macom fundcat note **********************

   * Create the actual output file (mtotfile)
   IF !USED('workingfile')
      use workingfile IN 0
   ENDIF
   SELECT workingfile
   index on macom + note + fundcat to m1
   total on macom + note + fundcat to mtotfile

   * Add a grand total section to the actual output file (mtotfile), will
   * create a new page for macom "GT"
   select workingfile
   index on note + fundcat to m2
   total on note + fundcat to mGTfile
   use mGTfile
   replace all macom with "GT"                  && ficticious macom for Grand Total page
   IF !USED('mtotfile')
      use mtotfile IN 0                         && actual total file for further use
   ENDIF
   SELECT mtotfile
   appe from mGTfile


*****
*
* section to create formal output to Excel
*
PROCEDURE generate_spreadsheet
   ************************** output to Excel ******************************
   ************************************************************************
   ** want one workbook with one worksheet for each MACOM
   ** plus one worksheet to identify which FSNs were assigned to which MACOM,
   ** "XX" is used for unidentified MACOM

   ** uses mtotfile.dbf

   #DEFINE xlLandscape 2                        && defines page orientation
   #DEFINE autoIn2Pts 72

   sele mtotfile
   GO top                                       && need to be at the first record to go through a
                                                ** loop correctly


   local oBook, oRange, oSheet
   mMacom = "  "                                && initialize

   oExcel = createobject("Excel.application")
         ** creates the application, opens an Excel session without
```

```
        ** opening a excel file

oExcel.visible = .f.                          && do not show Excel spreadsheet

oExcel.application.displayalerts = .f.        && turns off the excel warning
                                               ** message boxes

oBook = oExcel.workbooks.add()                && creates an actual workbook

oSheet = oBook.Activesheet                    && opens book with the default
                                               ** number of worksheets

numsheets = oExcel.activeworkbook.sheets.count
        ** see how many sheets are in a new file

oSheet1 = oBook.Worksheets[numsheets]
        ** defines the last sheet of the workbook if opened with the
        ** default number of sheets.  used to keep track of the last
        ** sheet worked


*************************************************
** get the sheet names for the default number of sheets to delete later
declare sheetname(numsheets)                  && declares array for sheetnames

with oExcel
   for i = 1 to numsheets
      .activeworkbook.sheets(i).activate
      sheetname[i] = .activesheet.name        && populates array with sheetnames
   endfor
endwith                                       && oExcel
*************************************************

*************************************************************************
** worksheets will be arranged GT, FSN to Macom, then individual macoms
*************************************************************************
do while not eof()
   mMacom = mtotfile.macom                    && used for the sheet name

   oSheet2 = oBook.Worksheets.Add(,oSheet1)
        ** Add a sheet for the new macom
        ** oSheet2 identifies the new sheet
        ** (,oSheet1) tells it to add the new sheet AFTER the previous
        **  sheet

   with oSheet2
      .name = mMacom                          && Name the sheet with the Macom


      if .name = 'XX'                         && next sheet will be GT - want it in front
         ***************
         oSheet1 = oBook.Worksheets[numsheets]
         ** resets the identifier for the previous sheet
         ***************
      else
         ***************
```

```
      oSheet1 = oBook.Activesheet
        ** resets the identifier for the previous sheet
      ***************
    endif

    .select()                                   && current sheet

    ** do the headings and put macom in heading
    .range("C1").Value = "SUSPENSE ACCOUNTS"
    .range("C2").Value = "MACOM " + mMacom
    .range("F2").Value = mMthYr
    .range("C3").Value = "(Absolute Values)"

    ** assign column headings
    .range("A5").Value = "Appn"
    .range("B5").Value = "Ex/Non-Ex"
    .range("C5").Value = "TOTAL"

********************** change for new structure 2/19/04 **********************
    .range("D5").value = "0-30"
    .range("E5").Value = "30-60 days"
    .range("F5").Value = "61-90 days"
    .range("G5").Value = "91-180 days"
    .range("H5").Value = "181-1yr"
    .range("I5").Value = "1yr - 10/1/97"
    .range("J5").Value = "Over 10/1/97"
        ** leave one blank line between header and detail
    oRange = .Range("A7:J7")
        ** sets the range to the first detail line to be filled in

    do while mMacom = mtotfile.macom and not eof()
      with oRange
        .Columns[1].Value = mtotfile.fundcat
        .Columns[2].Value = mtotfile.note
        .Columns[3].Value = mtotfile.totabs
        .Columns[4].Value = mtotfile.tadol0
        .columns[5].value = mtotfile.tadol31
        .Columns[6].Value = mtotfile.tadol61
        .Columns[7].Value = mtotfile.tadol91
        .Columns[8].Value = mtotfile.tadol181
        .Columns[9].Value = mtotfile.tadol1yr
        .Columns[10].Value = mtotfile.tadoloct97

*!*         .range("D5").Value = "0-60 days"
*!*         .range("E5").Value = "61-90 days"
*!*         .range("F5").Value = "91-180 days"
*!*         .range("G5").Value = "181-1yr"
*!*         .range("H5").Value = "1yr - 10/1/97"
*!*         .range("I5").Value = "Over 10/1/97"
*!*             ** leave one blank line between header and detail
*!*         oRange = .Range("A7:I7")
*!*             ** sets the range to the first detail line to be filled in

*!*         do while mMacom = mtotfile.macom and not eof()
*!*           with oRange
*!*             .Columns[1].Value = mtotfile.fundcat
```

```
*!*                 .Columns[2].Value = mtotfile.note
*!*                 .Columns[3].Value = mtotfile.totabs
*!*                 .Columns[4].Value = mtotfile.tadol0
*!*                 .Columns[5].Value = mtotfile.tadol61
*!*                 .Columns[6].Value = mtotfile.tadol91
*!*                 .Columns[7].Value = mtotfile.tadol181
*!*                 .Columns[8].Value = mtotfile.tadol1yr
*!*                 .Columns[9].Value = mtotfile.tadoloct97

****************** end of change for new structure 2/19/04 ******************

          endwith                                   && oRange

          ** move range down one row
          oRange = oRange.Offset(1,0)
          skip in mtotfile

        enddo                                       && individual macom

      endwith                                       && oSheet2

      nLastRow = oRange.Row                         && Row property always give the
                                                     ** first row of range.
                                                     ** This range has only one row

      cLastRow = alltrim(str(nLastRow))

      with oSheet2
        ** autofit the column width
        .Range("F2").NumberFormat = "Mmm-YY"

********************** change for new structure 2/19/04 **********************
        .Range("C7:J" + cLastRow).NumberFormat = ;
          [$###,###,###,###,###,###.00;$###,###,###,###,###,###.00;"-"]
        .Range("A5:J" + cLastRow).Columns.AutoFit()
*!*             .Range("C7:I" + cLastRow).NumberFormat = ;
*!*               [$###,###,###,###,###,###.00;$###,###,###,###,###,###.00;"-"]
*!*             .Range("A5:I" + cLastRow).Columns.AutoFit()
****************** end of change for new structure 2/19/04 ******************

        ** set up the page
        with .PageSetup
          .Orientation = xlLandscape
          .FitToPagesTall = 1
          .FitToPagesWide = 1
          .CenterHorizontally = .T.
          .CenterVertically = .F.
          .TopMargin = 1.0 * autoIn2Pts
          .BottomMargin = 1.0 * autoIn2Pts
          .LeftMargin = 1.0 * autoIn2Pts
          .RightMargin = 1.0 * autoIn2Pts
          .leftfooter = "&F  &A"                    && filename      sheetname
          .rightfooter = "&D"                       && date
        endwith                                     && Pagesetup
      endwith                                       && oSheet2

    enddo                                           && end of file
```

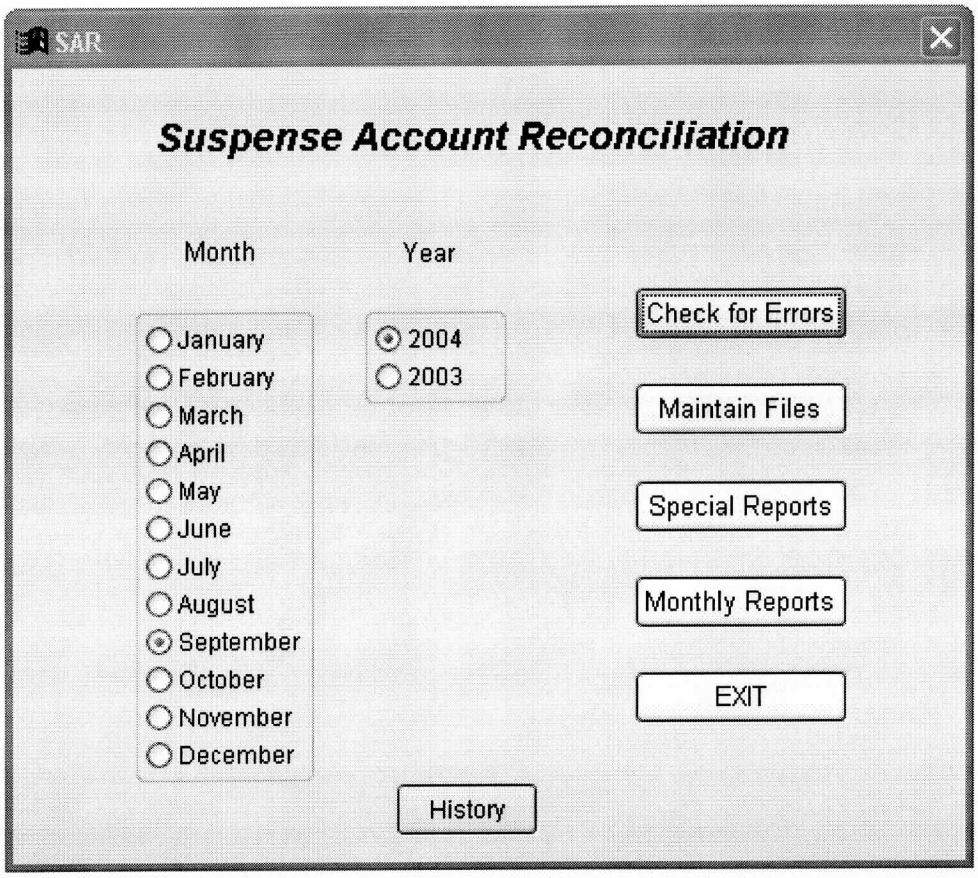
```
********** start of output FSN to MACOM list to Excel **********
** to identify each fsn assigned to the macom

use workingfile
index on macom + fsn to m3
use workingfile index m3

mMacom = "  "                                   && initialize
mFSN = "      "                                 && initialize
mColumn = 0


oSheet2 = oBook.Worksheets.Add(,oSheet1)
** Add a sheet for the fsn to macom list AFTER the last macom sheet(GT)

with oSheet2

   .name = "FSN to MACOM"                       && Name the sheet with the Macom
   .select()

   scan
      if mMacom != macom and macom != "GT"
         ** do not want a column for GT (Grand Total) macom
         row = 3
         mColumn = mColumn +1
         mMacom = macom
         .cells(1,mColumn).NumberFormat = "@"
         .cells(1,mColumn).Value = macom
      endif
      if mFSN != fsn and macom != "GT"
         .cells(row,mColumn).NumberFormat = "@"
         .cells(row,mColumn).Value = fsn
         mFSN = fsn
         row = row + 1
      endif
   endscan
endwith                                         && oSheet2
********** end of output FSN to MACOM list to Excel **********


*****************************************************
*****************************************************
** delete the original sheets ex: sheet1, sheet2, sheet3
with oExcel
   for i = 1 to numsheets
      ** set at original default number of sheets
      .activeworkbook.sheets(sheetname[i]).activate
      .activesheet.delete
   endfor
endwith                                         && oExcel
*****************************************************
*****************************************************


** Save the file
```

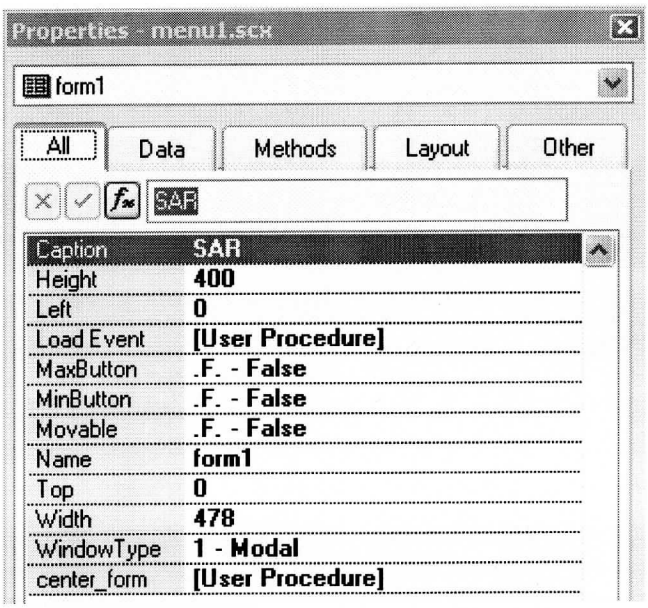
```
   ** set up the file name
   XLFile = fullpath(curdir()) + "output\SAR" + "&mMthYr" + ".XLS"
   XLFileJustName = juststem(XLFile)

   ** determine whether to use SaveAs or Save
   if oBook.Name <> XLFileJustName

      ** if the file already exists, delete it
      if file(XLFile)
         erase(XLFile)
      endif

      ** save it without user dialog box
      oBook.SaveAs(XLFile)

   else

      ** save it, since it's already been saved with SaveAs
      oBook.Save()
   endif


   oExcel.Quit()
   release oExcel


*****
*
* section to close down files and erase tables as needed
*
PROCEDURE close_down

   ********************* finishing steps - clean up ************************
   *** close the files
   IF USED('mtotfile')
      SELECT mtotfile
      USE
   ENDIF
   IF USED('mGTfile')
      SELECT mGtfile
      USE
   ENDIF
   IF USED('macomfile')
      SELECT macomfile
      USE
   ENDIF
   IF USED('workingfile')
      SELECT workingfile
      SET ORDER TO
   endif
   CLOSE INDEXES
   *** erase files
   erase mtotfile.dbf
   erase c:\temp\macomfile.dbf
   erase m1.idx
   erase m2.idx
```

```
erase m3.idx
erase m4.idx
erase m5.idx
erase mGTfile.dbf

SET SAFETY on
```

USER INTERFACE DESIGN & CODE

Properties shown in this section include only the Non-default values.



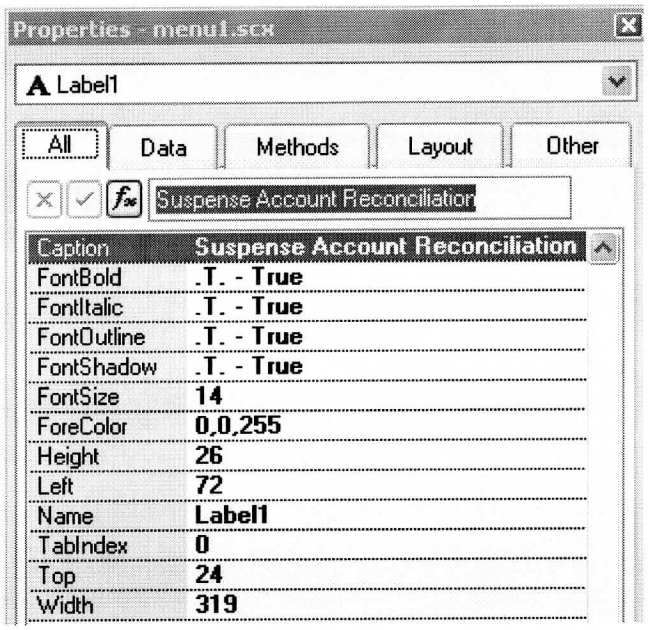


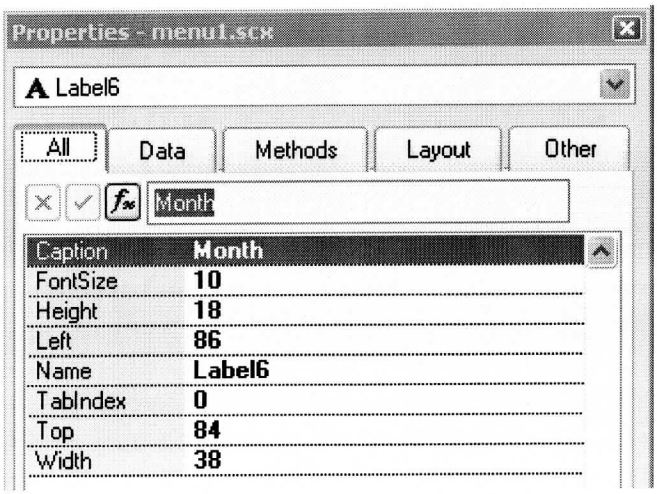
** menu1 form1 center_form

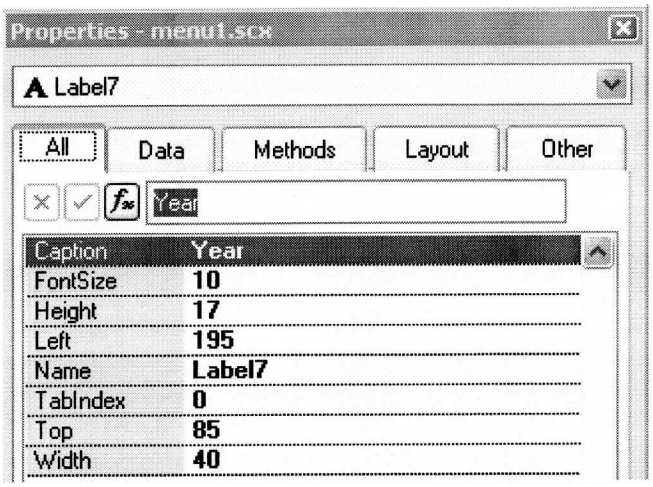
```
thisForm.Left = (_screen.width - thisForm.Width) / 2
thisform.top = (_screen.height - thisForm.height) / 2
```

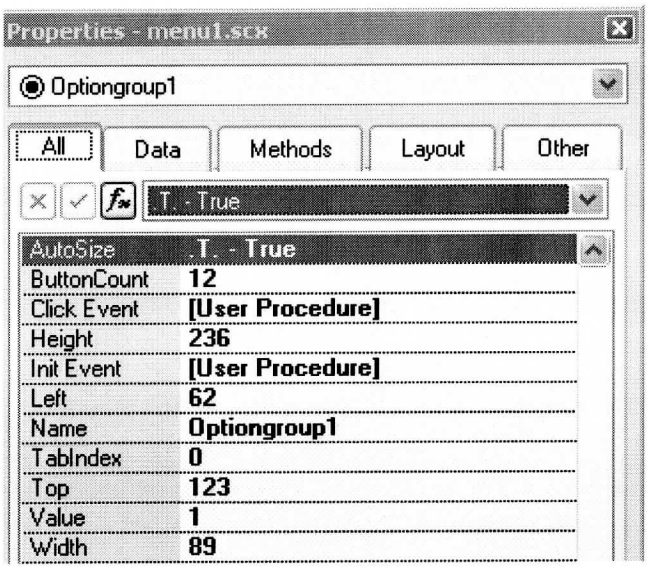
** menu1 form1 Load

```
thisForm.Left = (_screen.width - thisForm.Width) / 2
thisform.top = (_screen.height - thisForm.height) / 2
```







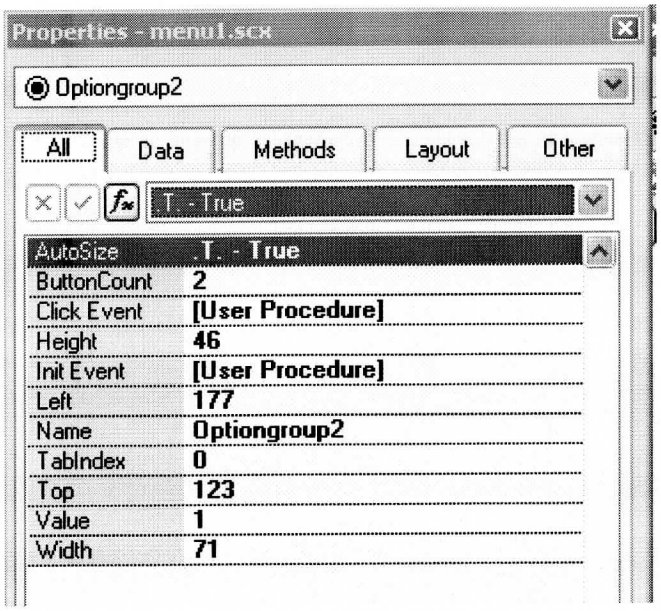


** menu1 optiongroup1 init

```
this.value = MONTH(GOMONTH(DATE(),-1))

mNumMth = ALLTRIM(Str(MONTH(GOMONTH(DATE(),-1))))
```

** menu1 optiongroup1 click

```
option_num = INT(this.Value)
k = 'this.option'+ALLTRIM(STR(option_num))+'.caption'
lmonth = &k
mMth = UPPER(SUBSTR(lmonth,1,3))
mMthYr = mMth + mYr
mNumMth = ALLTRIM(STR(option_num))
```

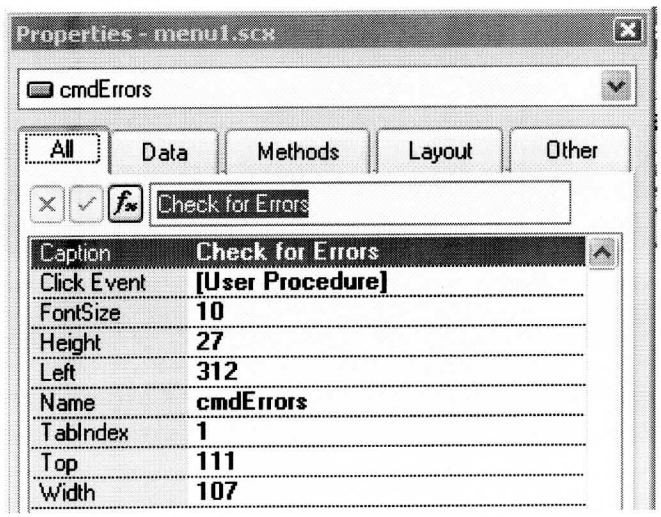


** menu1 optiongroup2 init

```
this.option1.Caption = ALLTRIM(STR(YEAR(DATE())))
this.option2.Caption = ALLTRIM(STR(YEAR(DATE())-1))
```

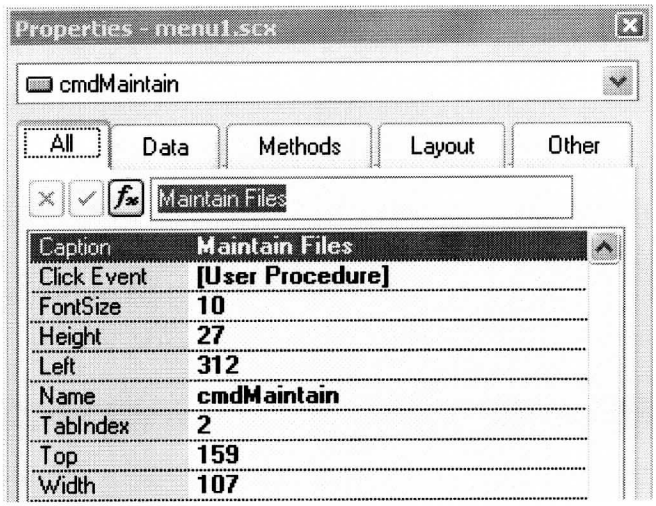
** menu1 optiongroup2 click

```
option_yr = INT(thisform.optiongroup2.Value)
k = 'thisform.optiongroup2.option'+ALLTRIM(STR(option_yr))+'.caption'
lyear = &k
mYr = SUBSTR(lyear,3,2)
mMthYr = mMth + mYr
```

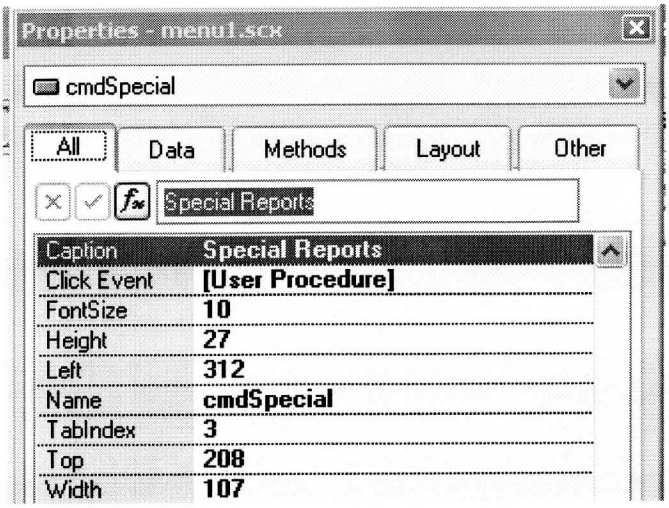


** menu1 cmdErrors Click

```
DO FORM scx\filelist

SET SAFETY OFF
CLOSE TABLES all

IF !USED('sarerror')
   USE sarerror                  && want to zap now instead of in checkdata procedure
   ZAP                           && if excel file does not have properly named sheets,
endif                            && an error message will be put into sarerror during
                                 ** getfiles procedure


DO prg\getfiles                  && count the number of files
                                 ** bring in the lines
                                 ** assign line numbers corresponding to Excel line number

IF numfile != 0
   DO prg\checkdata              && Validates fields
endif

RELEASE WINDOWS filelist
```

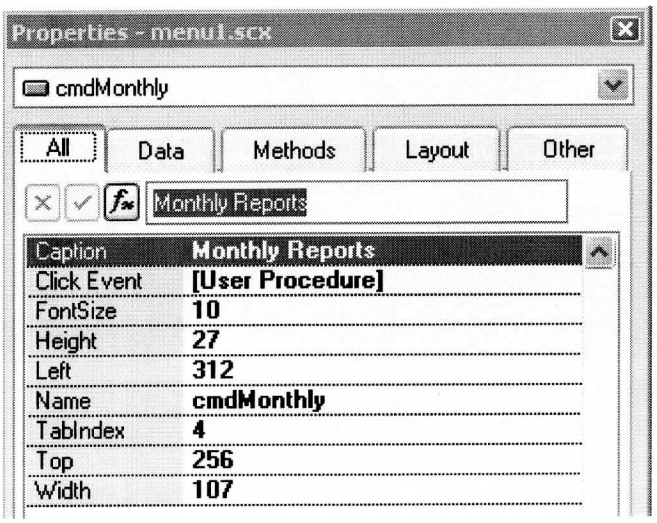



```
** menu1                cmdMaintain      Click

DO FORM scx\menu_maintain      && open the menu for Maintain Files option
```



** menu1 cmdSpecial Click

```
IF USED("submitters")
   SELECT submitters
   SET ORDER TO fsn
ELSE
   USE submitters ORDER fsn
ENDIF

PUBLIC gcCriteria, gc12, gcanswer as Character
STORE '' TO gcCriteria, gc12, gcanswer

DO FORM scx\menu_special
```

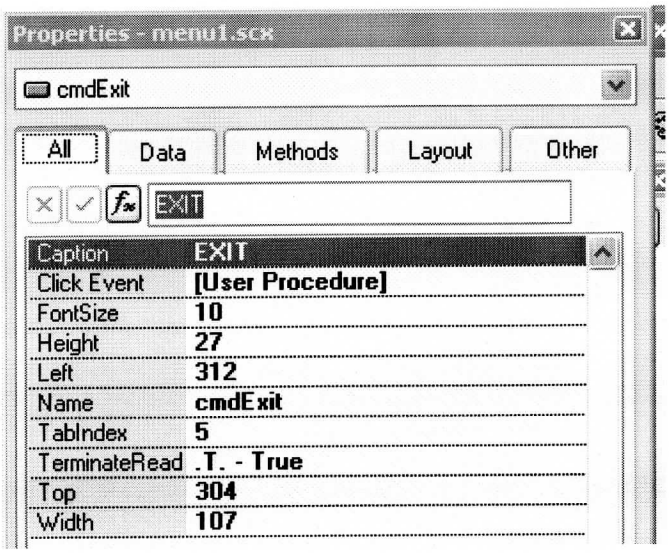


** menu1 cmdMonthly Click

```
CLOSE TABLES all
```

```
DO FORM scx\filelist

*!* added 9/5/04 to allow files brought in earlier to be used instead of creating the file again
nAnswer2 = MESSAGEBOX('Do you want to use the existing file? ',4+32)
DO case
   CASE nAnswer2 = 6                && yes
      set safety off
      set resource off
      set talk on

      IF !USED('masterfile')
         use masterfile IN 0
      ENDIF

      SELECT masterfile

      ****************************
      ** if masterfile was not created in current FoxPro session, numfile will not exist
      ** recreate
      PUBLIC mfileloc, numfile, mDirLoc, temp, listfile
      DECLARE listfile[1]

      mDirLoc = sys(5) + sys(2003) + '\datafile\'
      mFileLoc = mDirLoc + '*' + upper(mMthYr) + '.xls'

         ** in this instance the asterick is a multi-char wildcard
      numfile = adir(listfile, '&mfileloc')
         ** this creates an array of current cycle filenames and returns a
         ** count of the files found with MMMYY in the file name
      ****************************************

   CASE nAnswer2 = 7                && no
      ? 'Getting the Files'
      DO prg\getfiles               && count the number of files
            ** bring in the lines
            ** assign line numbers corresponding to Excel line number
endcase

*!* as was 9/5/04
*!*    ? 'Getting the Files'
*!*    DO prg\getfiles             && count the number of files
*!*          ** bring in the lines
*!*          ** assign line numbers corresponding to Excel line number


IF numfile != 0

   IF !USED('filesrecd')
      USE filesrecd IN 0
   endif
   SELECT filesrecd
   ZAP
   APPEND FROM masterfile FOR linenum = 1

   DO prg\mastercopy              && copies the masterfile to the first working copy, following
         ** programs will remove the header lines and "TOTAL TRUE SUSPENSE"
```

```
        ** lines for further processing.
        ** Want to keep the masterfile intact for other processes.

  DO prg\sarmacom              && prepares spreadsheet of SAR submitters by MACOM

  ?''
  ? 'Preparing the 3875 report'

  ** 01/30/2004
  DO prg\aimoutput2            && prepares AIM 3875 report
  *!*  DO prg\aimoutput        && prepares AIM 3875 report

********************** added 4/12/04 to create 702ab report ***************************
  DO prg\rpt702ab              && added 4/12/04 to produce the 702ab report
        ** uses a file created in aimoutput2
        ** did not run correctly within aimoutput2 as a procedure but
        ** works fine if run as its own program
********************** end of section to create 702ab report ***************************

  DO prg\high4


  ?''
  ? 'Marking Submitters'
  DO prg\mark_submit

  DO prg\mastercopy            && copies the masterfile to the second working copy, following
        ** programs will remove all zero amount lines.
        ** Want to keep the masterfile intact for other processes.

  DO prg\dfasall
  ?''

********************** added 4/15/04 to create pmi543 report ***************************
DO prg\pmi543
********************** end of section to create pmi543 report ***************************

  DO prg\movefiles
ENDIF

RELEASE WINDOWS filelist


SET SAFETY On

IF numfile != 0
  MESSAGEBOX('Process is completed. Thank you.',0+64)
endif

**CANCEL
```

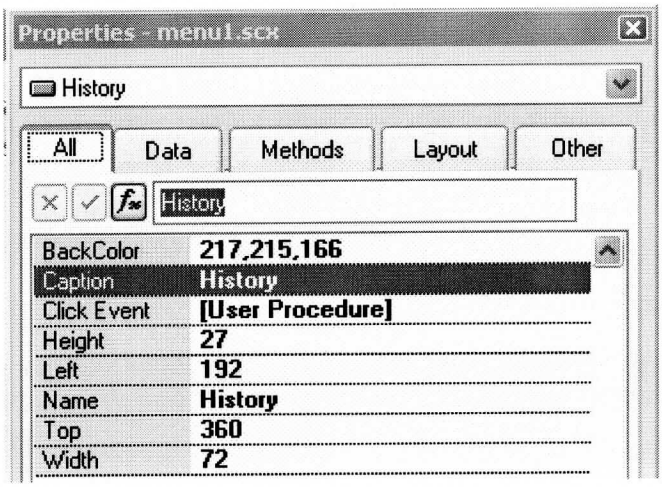


** menu1 cmdExit Click

CLEAR EVENTS

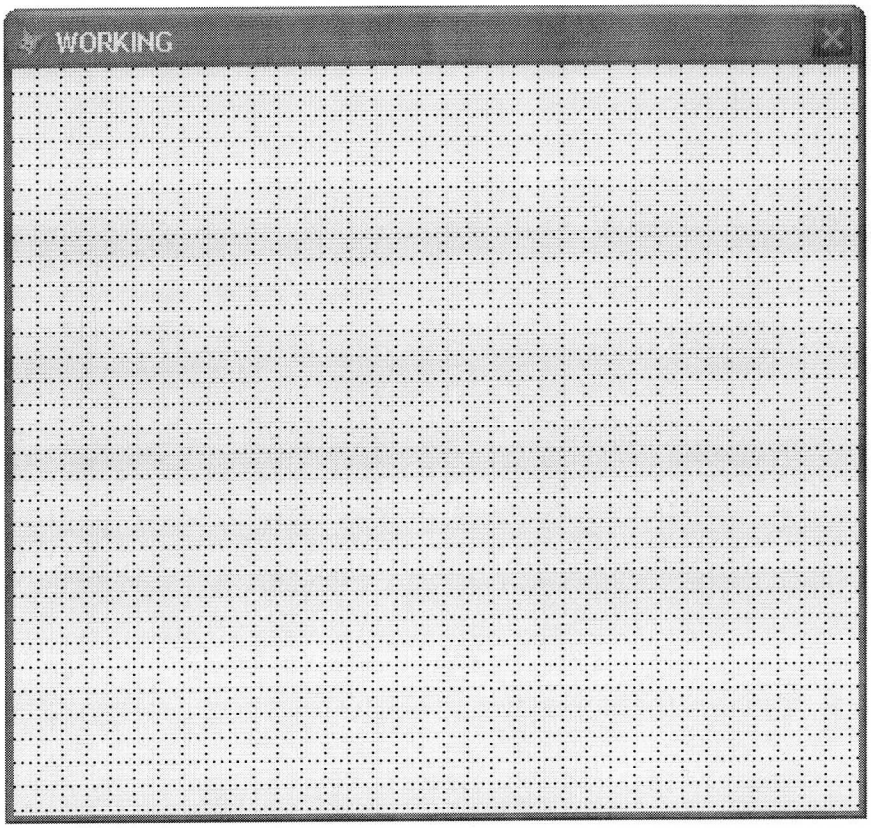
CLOSE TABLES all

thisform.Release
cancel



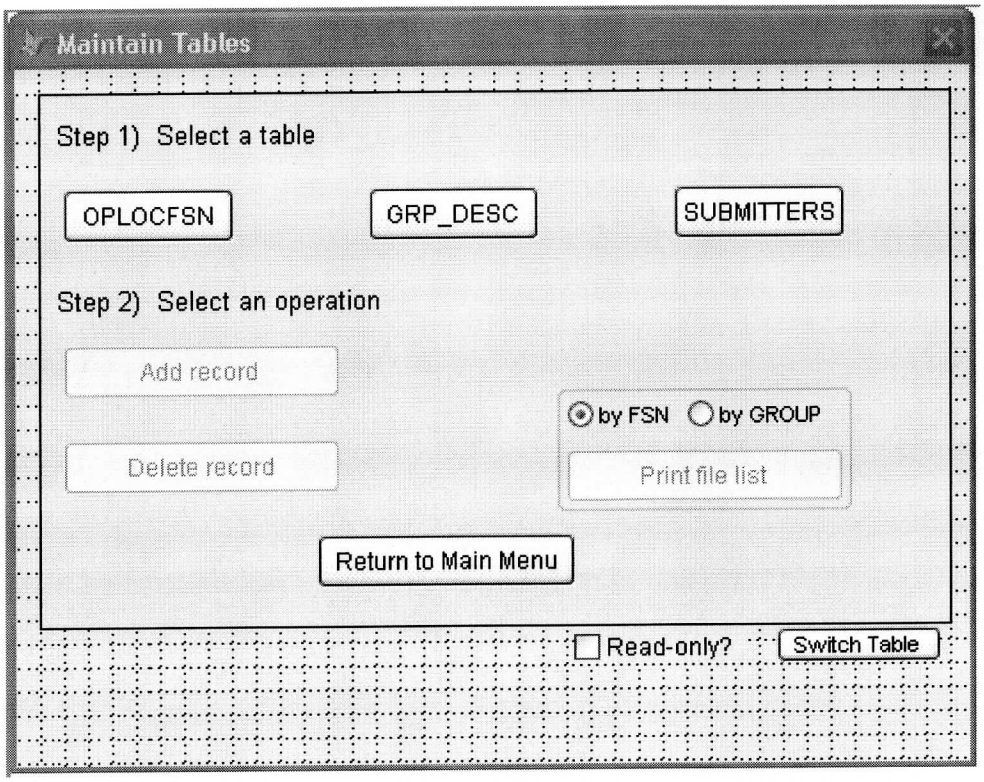
** History Click

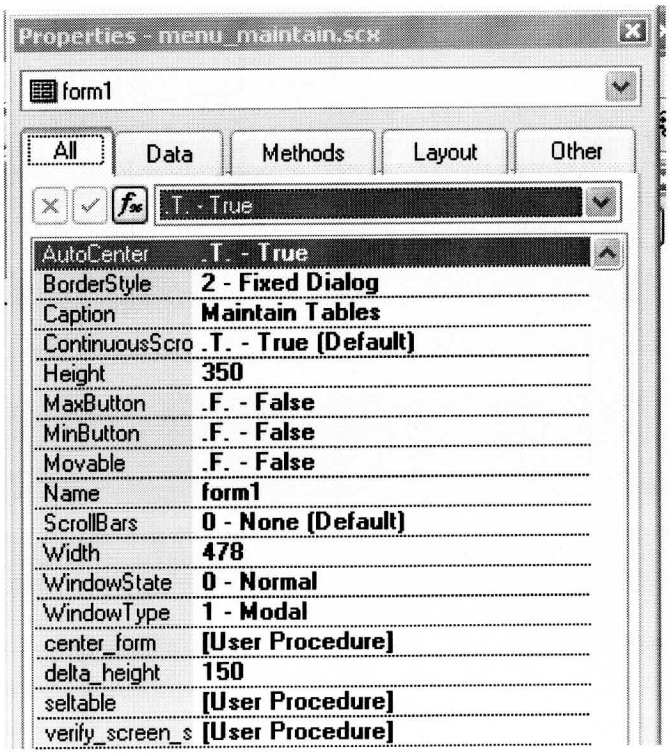
DO FORM scx\version



**filelist Load

thisForm.Left = (_screen.width - thisForm.Width) / 2
thisform.top = (_screen.height - thisForm.height) / 2



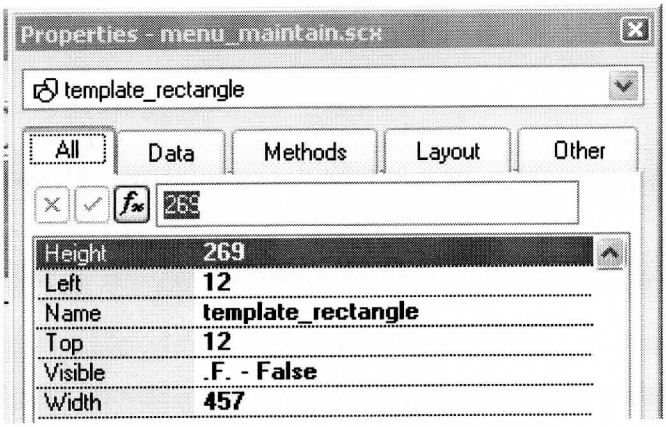


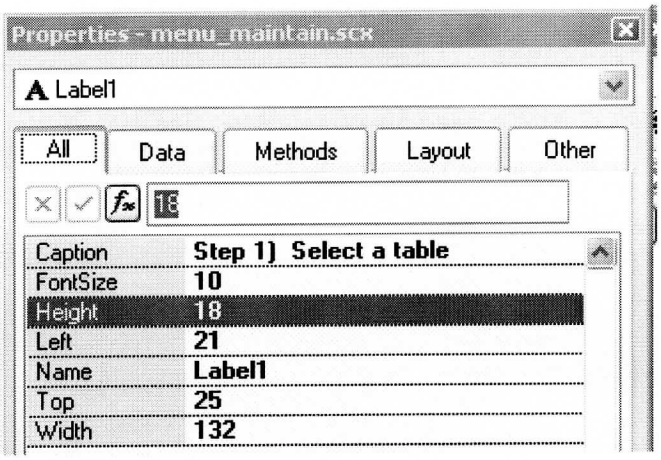
** menu_maintain form1 seltable

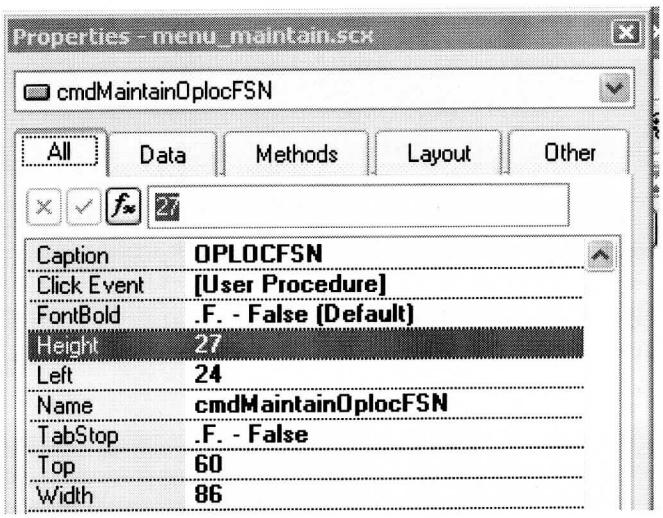
```
*PARAMETERS rcCaption
PARAMETERS rcCaption, rcSortOrder

CLOSE TABLES all

USE "&rcCaption" ALIAS grid_table
SET ORDER TO "&rcSortOrder"

* Add the descriptive label
thisForm.AddObject("grid_label", "Label")
thisForm.grid_label.left = thisForm.template_rectangle.Left
thisForm.grid_label.top = thisForm.template_rectangle.Top
thisForm.grid_label.height = 20
thisForm.grid_label.width = thisForm.template_rectangle.Width
thisForm.grid_label.backColor = RGB(0,128,0)
thisForm.grid_label.foreColor = RGB(255,255,255)
thisForm.grid_label.alignment = 2       && Centered
thisForm.grid_label.caption = PROPER(rcCaption)
thisForm.grid_label.visible = .t.

* Add the grid
thisForm.AddObject("grid1", "grid")
thisForm.grid1.left = thisForm.template_rectangle.left
thisForm.grid1.top = thisForm.template_rectangle.top + thisForm.grid_label.height
thisForm.grid1.width = thisForm.template_rectangle.width
thisForm.grid1.height = thisForm.template_rectangle.height - thisForm.grid_label.height
thisForm.grid1.recordSource = "grid_table"
thisForm.grid1.readonly = .t.
thisform.grid1.Visible = .t.
thisform.grid1.setfocus

thisform.bottom.cmdMaintainAdd.Enabled = .f.
thisform.bottom.cmdMaintainDelete.Enabled = .f.

* Make the "change table" button visible
thisForm.switch_table.Visible = .t.
thisForm.read_only.Visible = .t.

* Change the size of the form and position of the bottom controls
thisForm.Height = thisform.Height + thisForm.delta_height
thisForm.bottom.Top = thisform.bottom.Top + thisForm.delta_height
thisForm.center_form

* Update the read-only connection so it relates to this one
thisForm.read_only.ControlSource = "thisForm.grid1.readonly"

thisform.bottom.cmdMaintainPrint.Enabled = .t.


** menu_maintain          form1                    center_form

thisForm.Left = (_screen.width - thisForm.Width) / 2
thisform.top = (_screen.height - thisForm.height) / 2
```

** menu_maintain form1 verify_screen_size

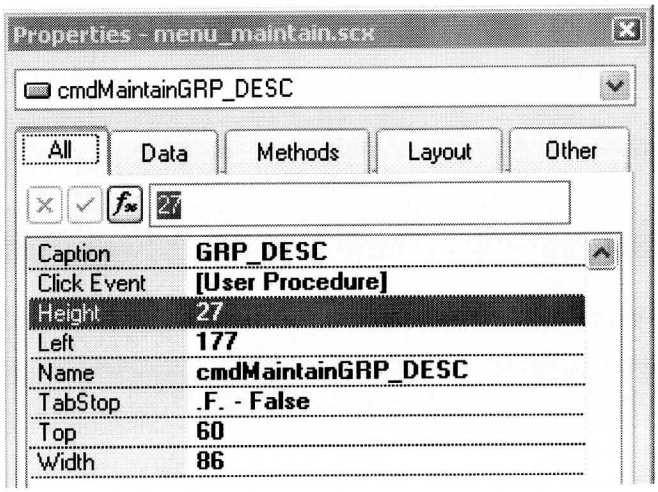
```
IF thisForm.Top < 0
   thisForm.Top = 0
endif
```





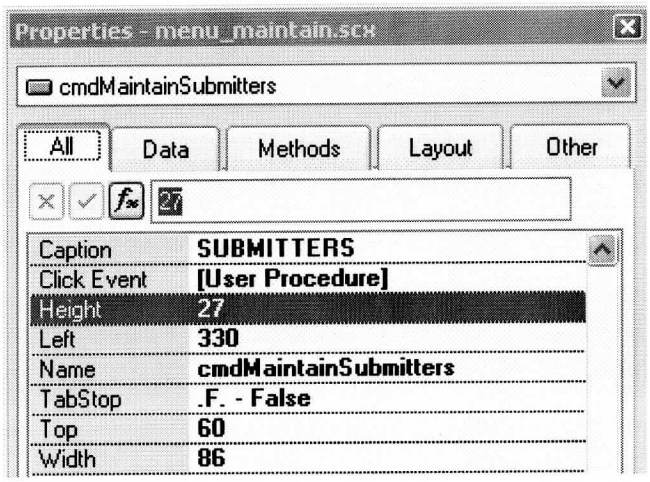


** menu_maintain cmdMaintainOplocFSN Click
thisform.seltable(this.Caption,"fsn")
thisform.bottom.optgrpMaintain1.Visible = .t.
thisForm.verify_screen_size



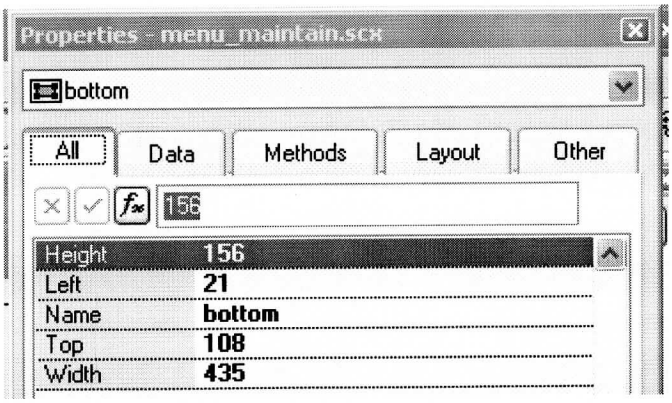
** menu_maintain cmdMaintainGRP_DESC Click

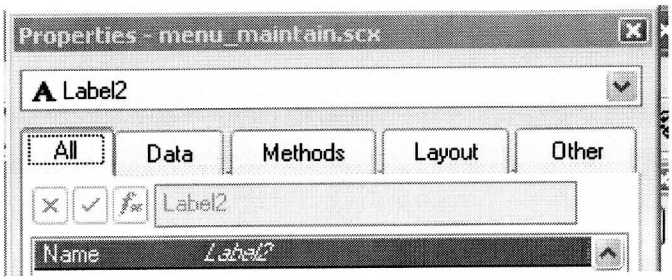
thisform.seltable(this.Caption,"group")
thisForm.verify_screen_size

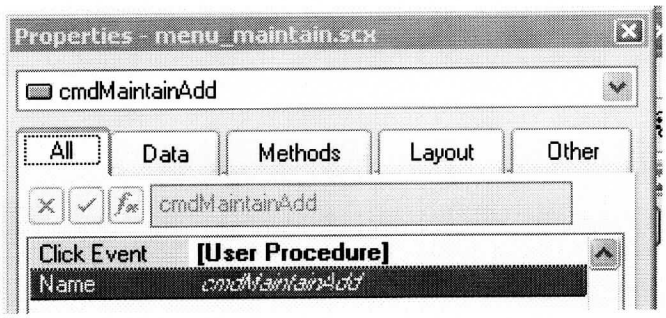


** menu_maintain cmdMaintainSubmitters Click

thisform.seltable(this.Caption,'triple')
thisForm.verify_screen_size

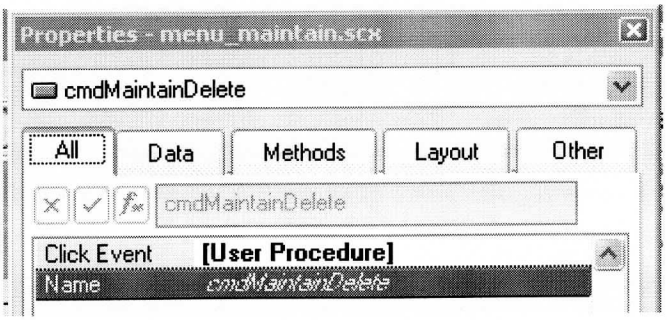






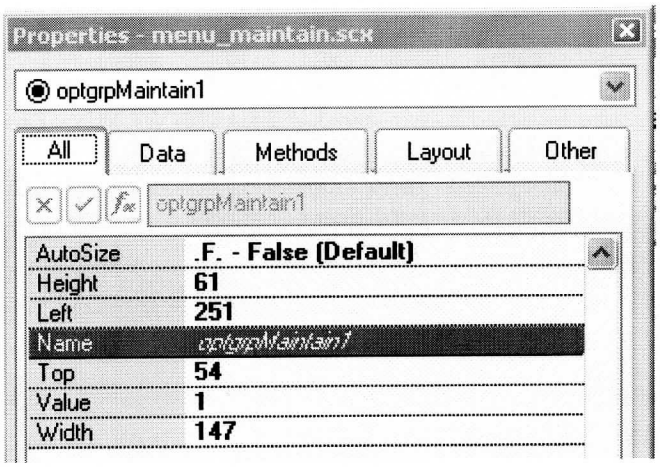
** menu_maintain cmdMaintainAdd Click

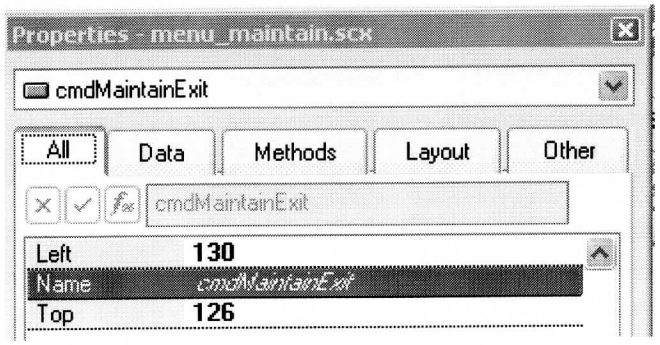
APPEND BLANK
GO top
thisform.grid1.setfocus

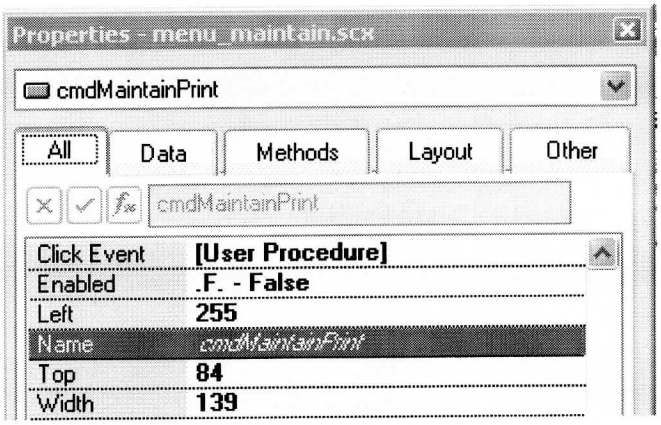


** menu_maintain cmdMaintainDelete Click

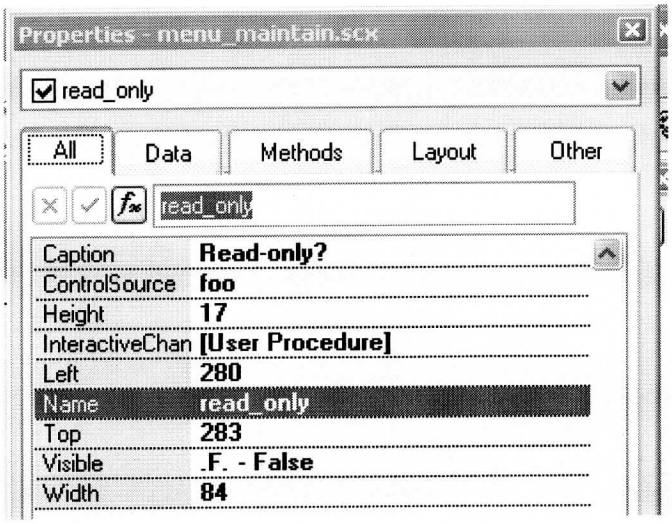
```
SET SAFETY ON
PACK
SET SAFETY OFF

k = JUSTSTEM(DBF())

thisform.switch_table.Click

DO case

   CASE UPPER(k) = thisform.cmdMaintainOPLOCFSN.Caption
      thisform.cmdMaintainOPLOCFSN.Click

   CASE UPPER(k) = thisform.cmdMaintainGRP_DESC.Caption
      thisform.cmdMaintainGRP_DESC.Click

   CASE UPPER(k) = thisform.cmdMaintainSubmitters.Caption
      thisform.cmdMaintainSubmitters.Click

endcase
```





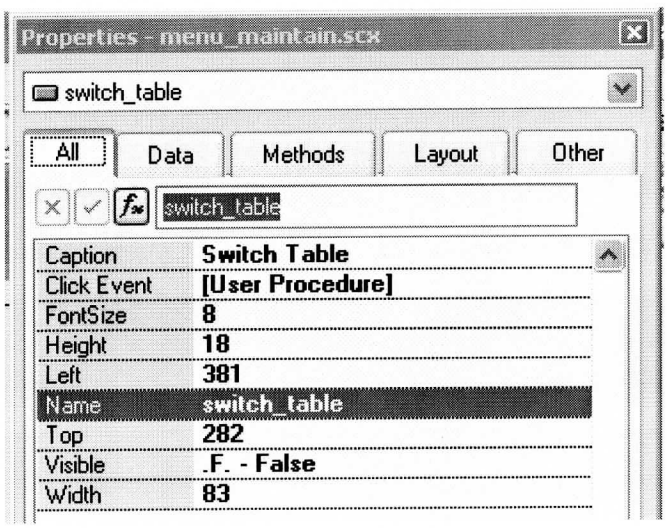



```
** menu_maintain          cmdMaintainPrint          click

printfile = JUSTSTEM(DBF())

DO case
   CASE UPPER(printfile) = "OPLOCFSN"
      thisform.bottom.optgrpMaintain1.Visible = .t.

      DO case
         CASE thisform.bottom.optgrpMaintain1.value = 1      &&option1 = by fsn
            printorder = 'FSN'          && order is fsn
            gcSort = 'FSN'              && tells a universal form which field is the sort field
            gcNonSort = 'GROUP'         && and which field is just a field on the detail line
         CASE thisform.bottom.optgrpMaintain1.value = 2      &&option2 = by group
            printorder = 'DBL'          && order is combination of group and fsn
            gcSort = 'GROUP'            && tells a universal form which field is the sort field
            gcNonSort = 'FSN'           && and which field is just a field on the detail line

      endcase

      SET ORDER to &printorder
*!*         USE &printfile INDEX m
*!*         USE &printfile INDEX gcSort
      REPORT FORM frx\rptoplocfsn TO PRINTER noconsole

   CASE UPPER(printfile) = "GRP_DESC"
      REPORT FORM frx\rptgrp_desc TO PRINTER noconsole

   CASE UPPER(printfile) = "SUBMITTERS"
      SET ORDER TO triple
      REPORT FORM frx\rptsubmitters TO PRINTER noconsole

endcase

**LIST to print off noconsole
```



** menu_maintain read_only checkbox interactive change

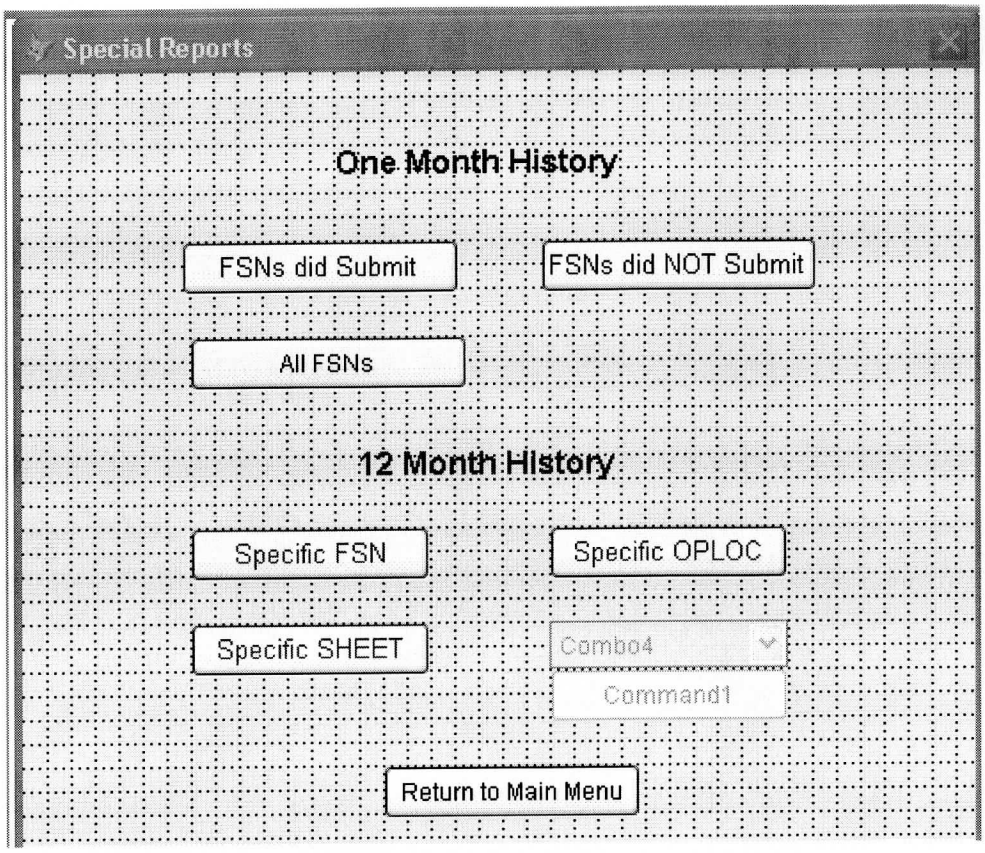
```
IF this.Value
   thisForm.bottom.cmdMaintainAdd.Enabled = .f.
   thisForm.bottom.cmdMaintainDelete.Enabled = .f.

ELSE
   thisForm.bottom.cmdMaintainAdd.Enabled = .t.
   thisForm.bottom.cmdMaintainDelete.Enabled = .t.

endif
```

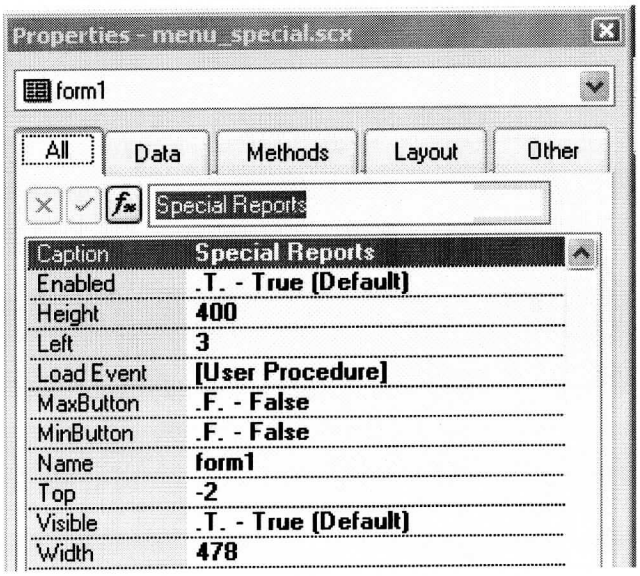


** menu_maintain switch_table Click

```
IF TYPE("thisForm.grid1.name") != "U"
   * The grid is defined
   thisform.read_only.ControlSource = "foo"
```

```
    thisForm.RemoveObject(thisForm.grid_label.name)
    thisForm.RemoveObject(thisForm.grid1.name)
    thisForm.Height = thisForm.Height - thisForm.delta_height
    thisForm.bottom.Top = thisForm.bottom.Top - thisForm.delta_height
    thisForm.center_form
    this.Visible = .f.
    thisForm.read_only.visible = .f.
    USE IN grid_table

    thisform.bottom.optgrpMaintain1.Visible = .f.
    thisform.bottom.cmdMaintainPrint.Enabled = .f.
ENDIF
```

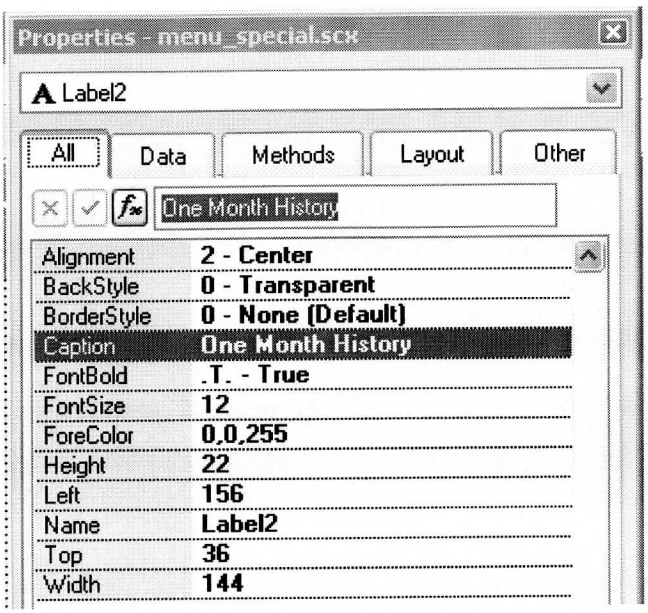


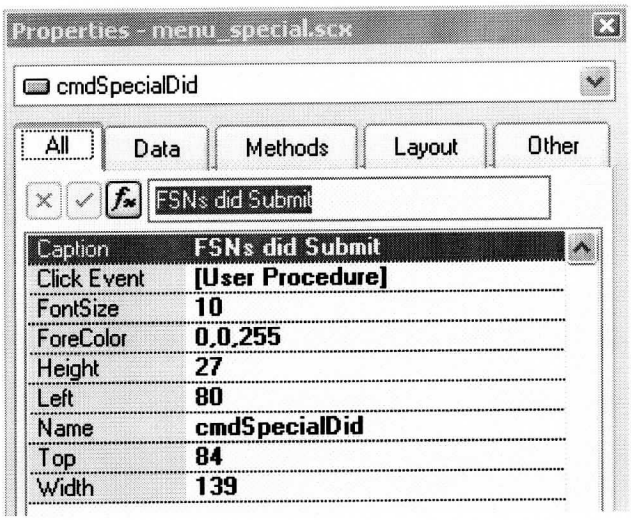


** menu_special form1 load

thisForm.Left = (_screen.width - thisForm.Width) / 2
thisform.top = (_screen.height - thisForm.height) / 2

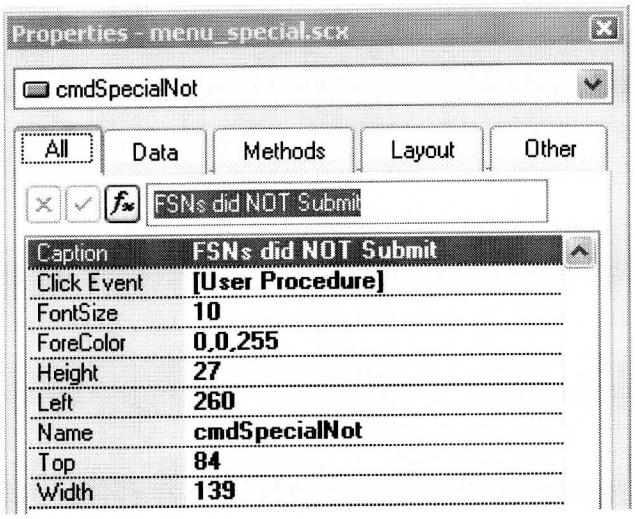
gcanswer = ''





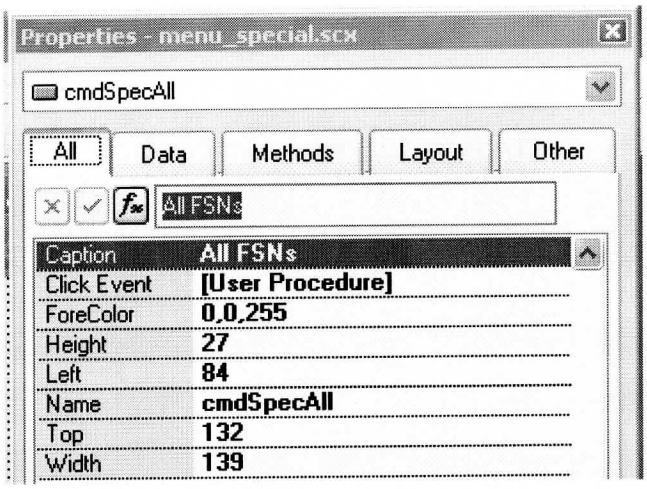
** menu_special cmdSpecialDid click

SET ORDER to triple
gcdid_not = 'DID'
REPORT FORM frx\rptSpec1mth TO PRINTER NOCONSOLE FOR &mMth = 'Y'



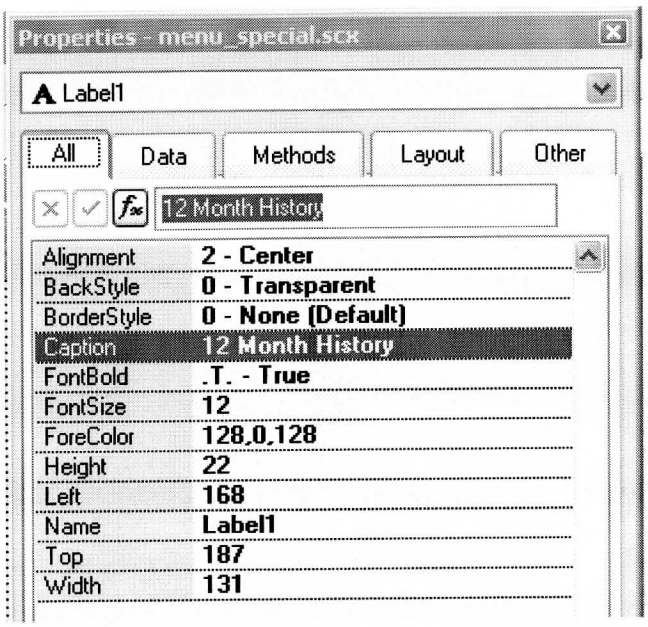
** menu_special cmdSpecialNot click

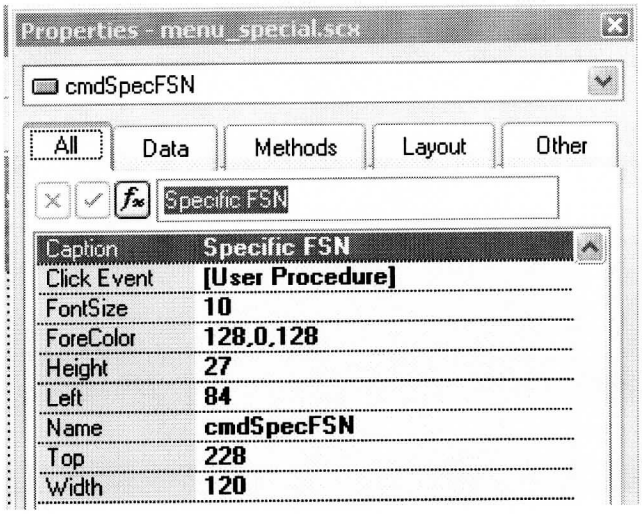
SET ORDER to triple
gcdid_not = 'DID NOT'
REPORT FORM frx\rptSpec1mth TO PRINTER NOCONSOLE FOR INLIST(&mMth, 'N', ' ')



** menu_special cmdSpecialAll click

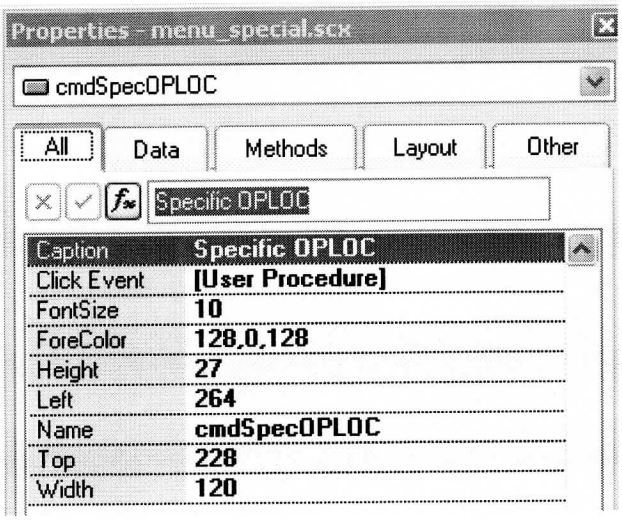
SET ORDER to triple
REPORT FORM frx\rptSpec1mthAll TO PRINTER NOCONSOLE





** menu_special cmdSpecFSN click

```
gcanswer = ''

select distinct fsn from submitters into cursor uniqfsn
thisform.combo4.RowSource = 'uniqfsn'
thisform.combo4.Enabled = .t.
thisform.combo4.Visible = .t.

gc12 = 'F'

thisform.cmdPrint12.Visible = .f.
thisform.cmdPrint12.Enabled = .f.
thisform.cmdPrint12.Caption = ' '
```



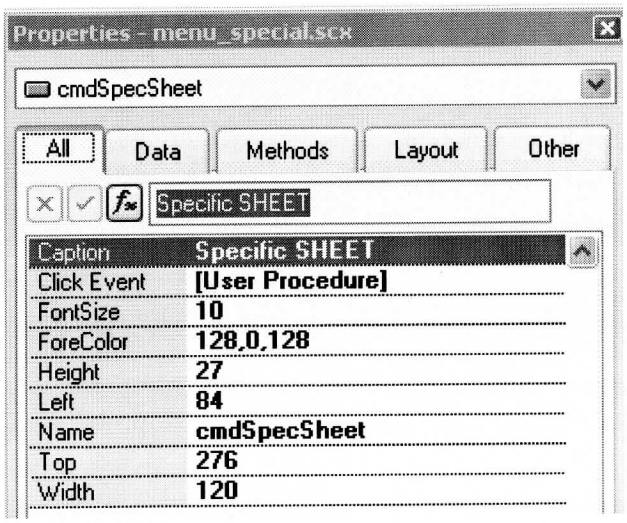
** menu_special cmdSpecOPLOC click

gcanswer = ''

select distinct oploc from submitters into cursor uniqoploc
thisform.combo4.RowSource = 'uniqoploc'
thisform.combo4.Enabled = .t.
thisform.combo4.Visible = .t.

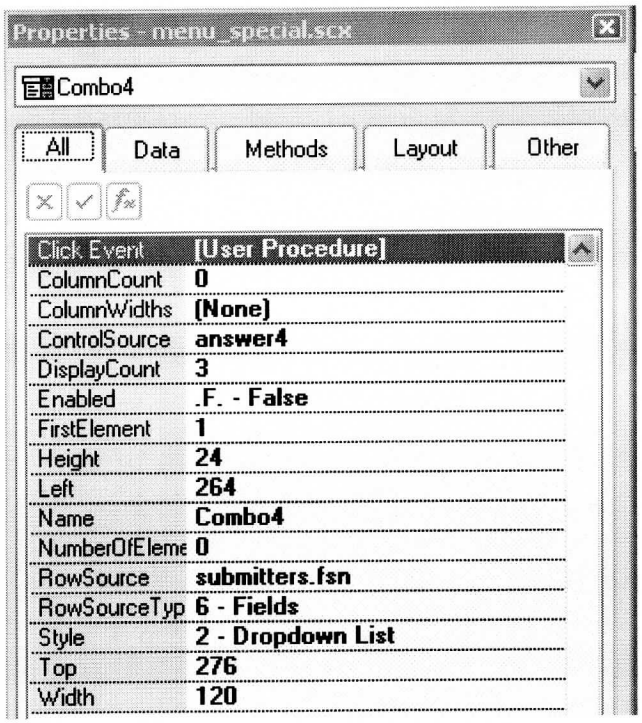
gc12 = 'O'

thisform.cmdPrint12.Visible = .f.
thisform.cmdPrint12.Enabled = .f.
thisform.cmdPrint12.Caption = ' '



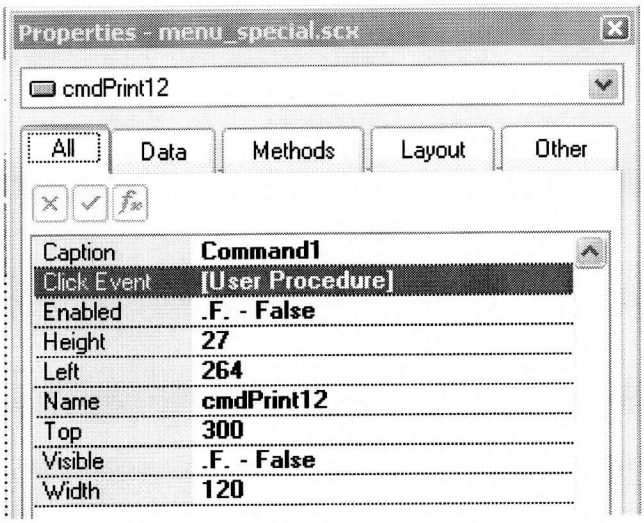
** menu_special cmdSpecSheet click

```
gcanswer = ''

select distinct sheet from submitters into cursor uniqsheet
thisform.combo4.RowSource = 'uniqsheet'
thisform.combo4.Enabled = .t.
thisform.combo4.Visible = .t.

gc12 = 'S'

thisform.cmdPrint12.Visible = .f.
thisform.cmdPrint12.Enabled = .f.
thisform.cmdPrint12.Caption = ' '
```



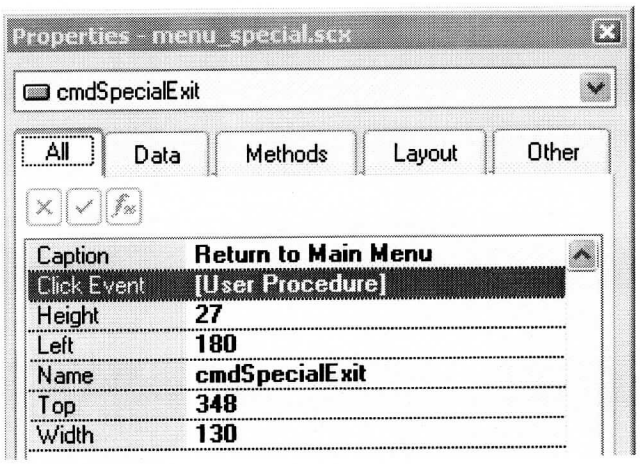
** menu_special Combo4 Click

```
gcanswer = thisform.combo4.Value

DO case
   CASE gc12 = 'F'
      thisform.cmdPrint12.Caption = 'Print FSN '+gcanswer
   CASE gc12 = 'O'
      thisform.cmdPrint12.Caption = 'Print OPLOC '+gcanswer
   CASE gc12 = 'S'
      thisform.cmdPrint12.Caption = 'Print SHEET '+gcanswer
ENDCASE

thisform.cmdPrint12.Visible = .t.
thisform.cmdPrint12.Enabled = .t.
```



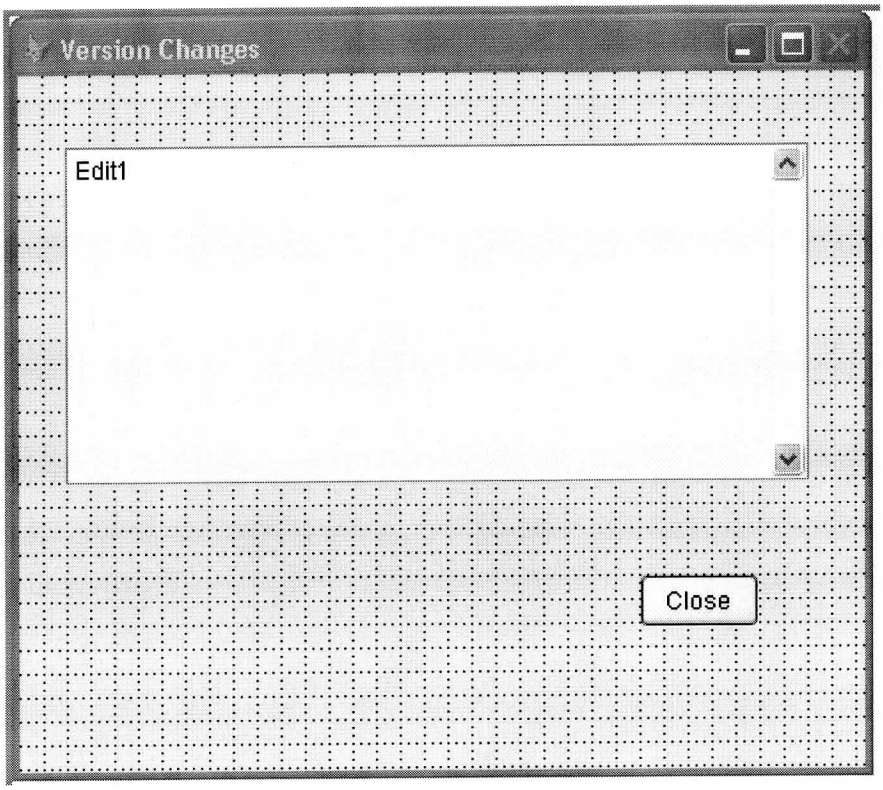
** menu_special cmdPrint12 Click

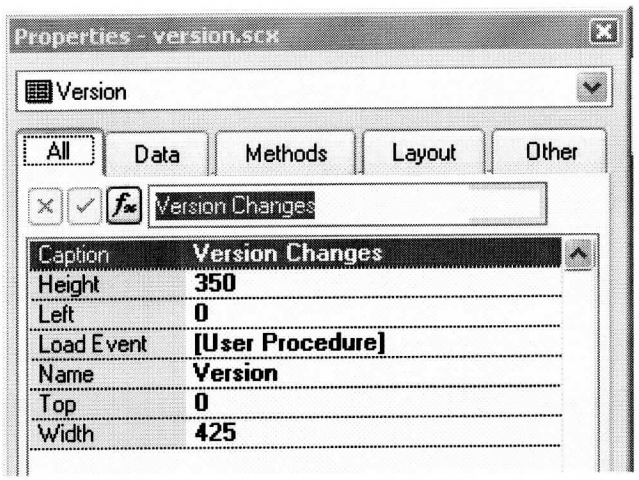
```
SELECT submitters
SET ORDER TO triple
**PUBLIC gcCriteria as Character      done in main menu

DO case
   CASE gc12 = 'F'
      gcCriteria = 'Specific FSN  -  '+ gcanswer
      REPORT FORM frx\rptSpec12mth TO PRINTER NOCONSOLE FOR FSN = gcanswer
   CASE gc12 = 'O'
      gcCriteria = 'Specific OPLOC  -  '+ gcanswer
      REPORT FORM frx\rptSpec12mth TO PRINTER NOCONSOLE FOR OPLOC = gcanswer
   CASE gc12 = 'S'
      gcCriteria = 'Specific SHEET  -  '+ gcanswer
      REPORT FORM frx\rptSpec12mth TO PRINTER NOCONSOLE FOR SHEET = gcanswer
Endcase
```



** menu_special cmdSpecialExit Click

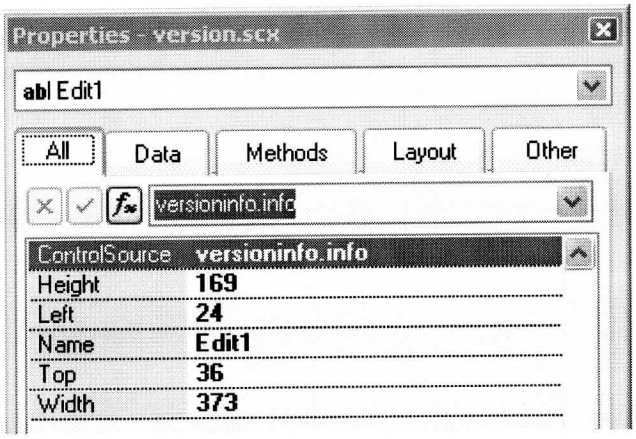
thisform.release

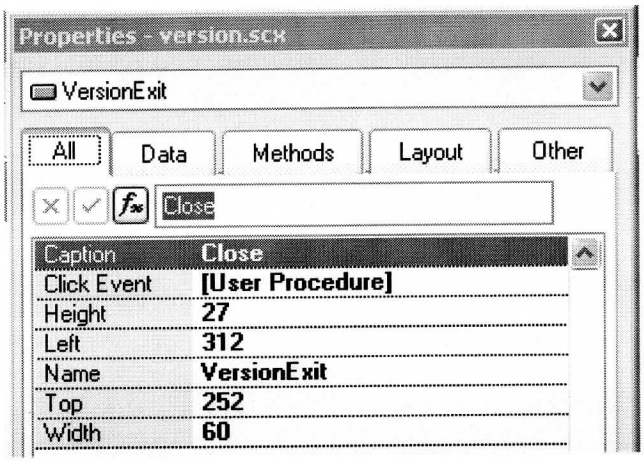




** version Load

```
thisForm.Left = (_screen.width - thisForm.Width) / 2
thisform.top = (_screen.height - thisForm.height) / 2

IF !USED('versioninfo')
   USE scx\versioninfo IN 0
endif
SELECT versioninfo
```





** version cmdVersionExit Click

thisform.release

Acronyms

DFAS	Defense Finance & Accounting Service
FSN	Fiscal Station Number
MACOM	Major Command
OPLOC	Operating Location
SAR	Suspense Account Reconciliation
SOP	Standard Operating Procedure