Standardizing and Automating Blaise Output Test Data Delivery (TransSAS)

Latha Srinivasamohan (U.S. Census Bureau)

1. Background

This paper will discuss an automated Blaise output delivery tool known as TransSAS. This tool runs on a multi-user instrument testing environment developed and supported by the Technologies Management Office (TMO) of the U.S. Census Bureau.

2. Overview

As Blaise instruments at the Census Bureau became complex, the need for a tool that facilitated testing of the instrument data as well as the instrument itself became apparent. There are several tools to help in the software development process in Blaise, but we realized that a utility tool for testing the data that is output from Blaise would be useful. For the CASES instruments we provided data output from the testing environment using an application developed by the Demographic Surveys Division (DSD) data processors, called TransCASES. The TransSAS utility has been developed recently to provide similar output from our Blaise instruments. It is a tool, which allows testers and data processors a way to retrieve and review SAS output from a Blaise instrument from the TMO testing environment.

This paper will briefly describe the testing environment (known as TMOusers Menu), discuss the importance of data verification by sponsors and data processors during the early stages of instrument development, and point out some of the issues we have encountered while running TransSAS in a multi-user environment.

3. Overview of the Multi-User Testing Environment at the Census Bureau

The TMOusers Menu is a WinBatch application, which has a survey-specific Manipula interface to access the Blaise instruments. All current surveys are listed here, and access is given to appropriate sponsors/testers for testing of their surveys. Developers have read/write access to the shared drive on which the TMOusers Menu runs to allow for submitting and testing of the instruments. The figures below show the three screens that come up before arriving at the Maniplus interface.

Figure 3.1

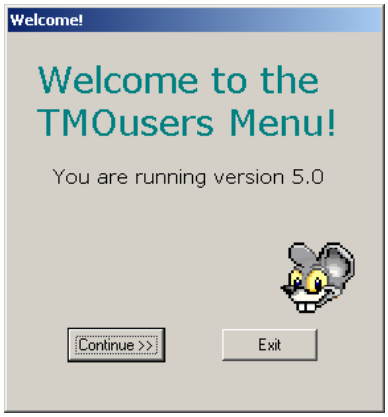


Figure 3.2

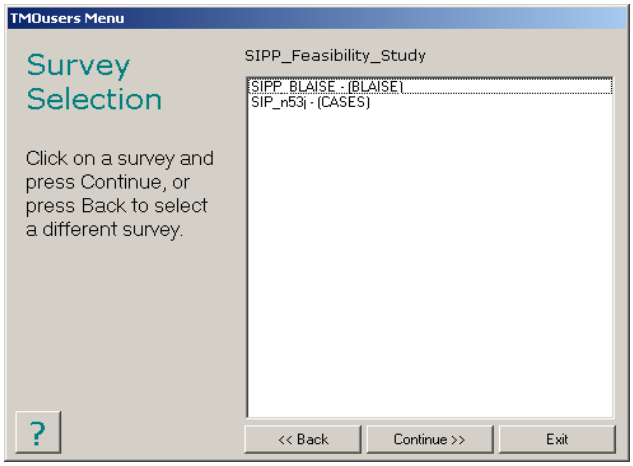
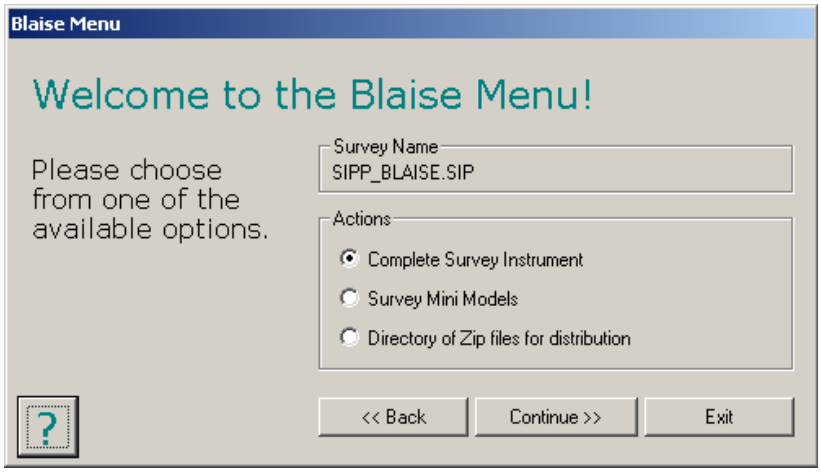


Figure 3.3



4. Why TransSAS?

The need for a centralized automated testing tool became apparent as more of our surveys were being converted to Blaise from CASES. The data processors already had an output processing system based on the data received from the CASES instruments. The output processing teams needed to modify their system for Blaise instruments. There was no easy way to look at data during instrument development without a tool to export Blaise data into a readable format.

5. Importance of Identifying Data Issues in Early Stages

5.1 Data Integrity

Identifying data processing requirements early in development helps maintain data integrity. Subsequently, data retention issues can be reduced if sponsors and data processors are able to verify data during unit testing. Our instrument development process consists of several rounds of unit or module (mini-model) testing, in which each module is placed on the TMOusers Menu for sponsors to test. Extracting data from mini-models is possible using the TransSAS application. Some critical issues can be identified during this process and can be brought to the attention of developers. It is possible at this stage to reorganize the blocks before integrating the instrument. This process is like normalizing a relational database to create data output in a format that the processors expect. Not only is the data dependable, but this process of re-organizing the blocks helps create more compact instruments, which enhances performance.

5.2 Time Efficiency for Developers, Sponsors and Processors

One of the main goals for development teams is maintaining timely releases of the deliverables to the client. Maintaining the quality of our products is another important goal. Prior to the development of TransSAS, there was no standard method of providing output to the data processors until the systems test. A systems test is conducted to verify that the instrument is able to interface with our control systems. At this stage of development the instrument is almost ready for production. The complexity of our surveys makes it difficult to quickly debug and fix a problem if it is identified at this stage. Although it can still be fixed, it can be time- consuming for developers and might create a setback in our development process causing a chain of delays for other areas as well. Therefore getting a timely feedback from the sponsors regarding data issues help in fixing major problems before instruments go to systems test. Sometimes data that is output from one interview serves as input for other follow-on interviews with the same respondent. TransSAS can be used to help create test input data for follow-on instruments on the testing system.

6. Common Tools Used to build TransSAS

The Technologies Management Office provides their sponsors with three output options to receive production data from Blaise instruments:

1. Blaise database
2. ASCII files
3. ASCII Relational files.

[1]Xiaodong Guan, at U.S. Census Bureau, has discussed in detail these three output options in his paper “Output Processing for Consumer Expenditure Survey “ and how his output data from a highly complex instrument. His paper concluded that using ASCII relational output was the best method for their office.

Consequently, the Demographic Surveys Division requested that TMO use the ASCII Relational option when implementing TransSAS. We create a Blaise-to-ASCII relational export script using Manipula, which generates the ASCII Relational files, and subsequently use Cameleon scripts to generate setup files for running SAS. The Cameleon Translator files (.CIF), can be used to generate SAS

[1] Output Processing for Consumer Expenditure Survey Xiaodong Guan, U.S. Census Bureau Howard R McGowan, U.S. Census Bureau

code for selected blocks in the instrument. We used a Cameleon script that was developed at Census to generate SAS macros, which can be easily run by data processors.

The Cameleon scripts can also generate setup files to the Oracle client server environment, Paradox, or any other database. The WinBatch programming tools are used to link all the programs to create a standalone application for distribution in a multi-user testing environment. TransSAS helps to achieve our goal of exporting data from Blaise and to generate SAS code for analysts.

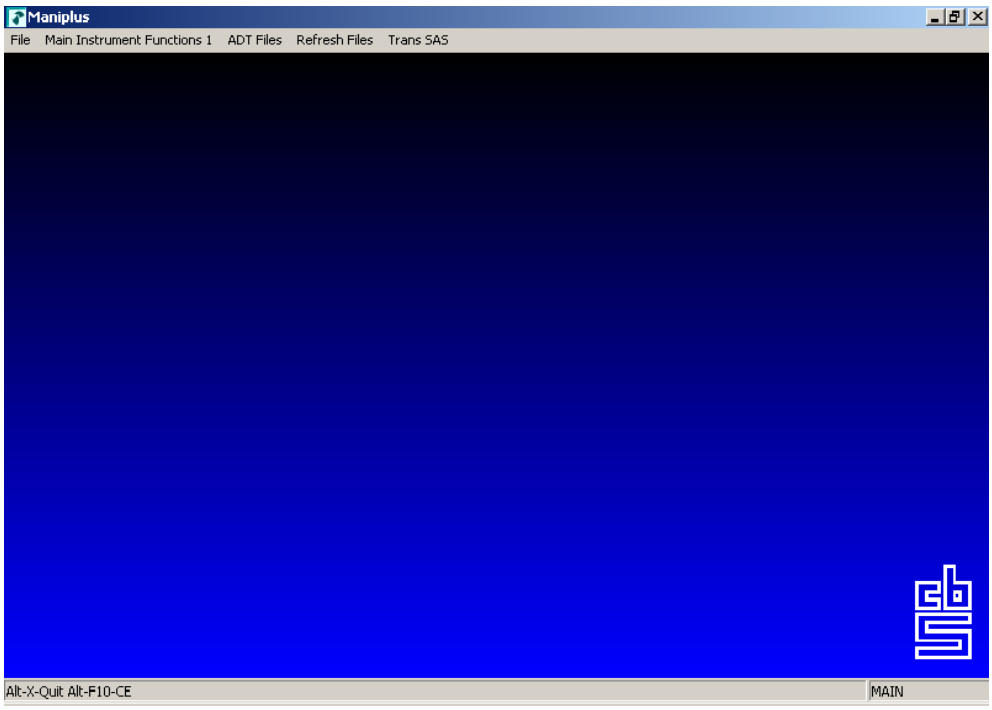
To sum up, TransSAS was built using the following software and tools.

1. Blaise
2. Manipula
3. Cameleon
4. WinBatch

7. How TransSAS works

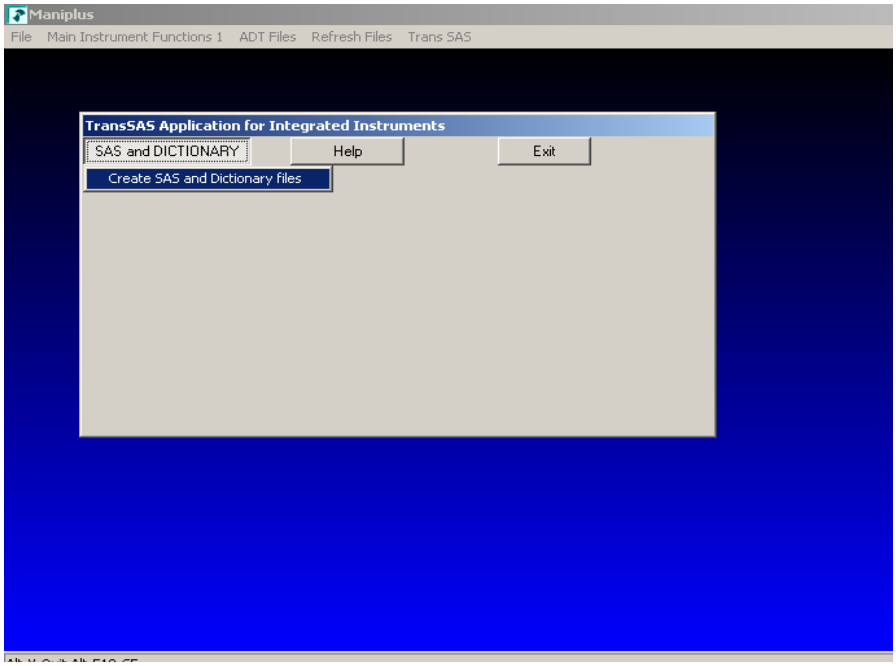
We have seen some of the common tools that have been used to create TransSAS. Now we will see how it works. The TransSAS executable is located at the survey level folder of our instruments on the TMOusers Menu. The testers and sponsors are given rights to access their respective surveys to test the instruments. They go through the preliminary screens on the TMOusers Menu (see Figures 3.1-3.3) to get to this Maniplus interface. The Maniplus interface shows several drop-down menus on the menu bar, which are based on survey specific requirements. They include options to run the instrument by selecting a case id, view audit trails, and refresh the databases. TransSAS is seen as the last option (see Figure 7.1).

Figure 7.1. Maniplus interface to test the instrument



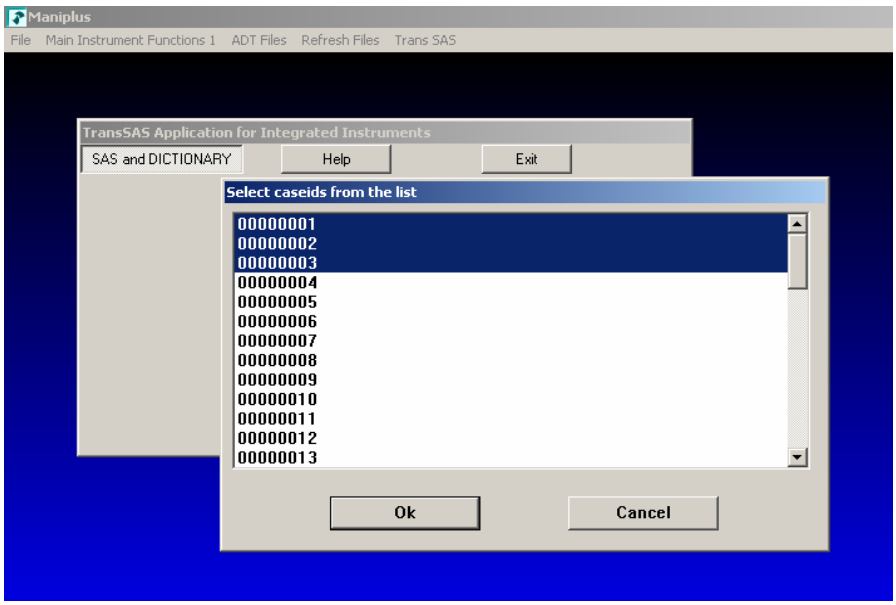
When users are at this screen they can perform several functions, such as picking a case and entering data, looking at the audit key files and so on. Now with TransSAS being an option, they can choose to output the data as well. When the TransSAS option is picked, the following screen is displayed (see Figure 7.2).

Figure 7.2



From this point on the control is turned over to WinBatch. When users pick the first option to create the SAS and dictionary files, a screen with a list of caseids is displayed (see Figure 7.3). These are cases that are available to test and therefore the cases that could contain data for output. The program allows users to select multiple cases from the list by holding down the shift key. It is important that the user selects specific cases to output data. Otherwise, if no cases are identified, the program will produce output for all cases.

Figure 7.3



Once the cases are identified, the program creates a consolidated database by combining the selected cases into one database. The ASCII relational files are output for the selected cases based on the meta data.

The Manipula script generates the ASCII Relational data files for each block. It connects the block name, block number, data model name, and ASCII data set name.

The Cameleon script has been customized to create the SAS files for these cases. It also produces a dictionary file, which contains a list of variables that are present in the instrument.

Since all the scripts are run from the WinBatch application, control does not pass back over to Maniplus until WinBatch terminates.

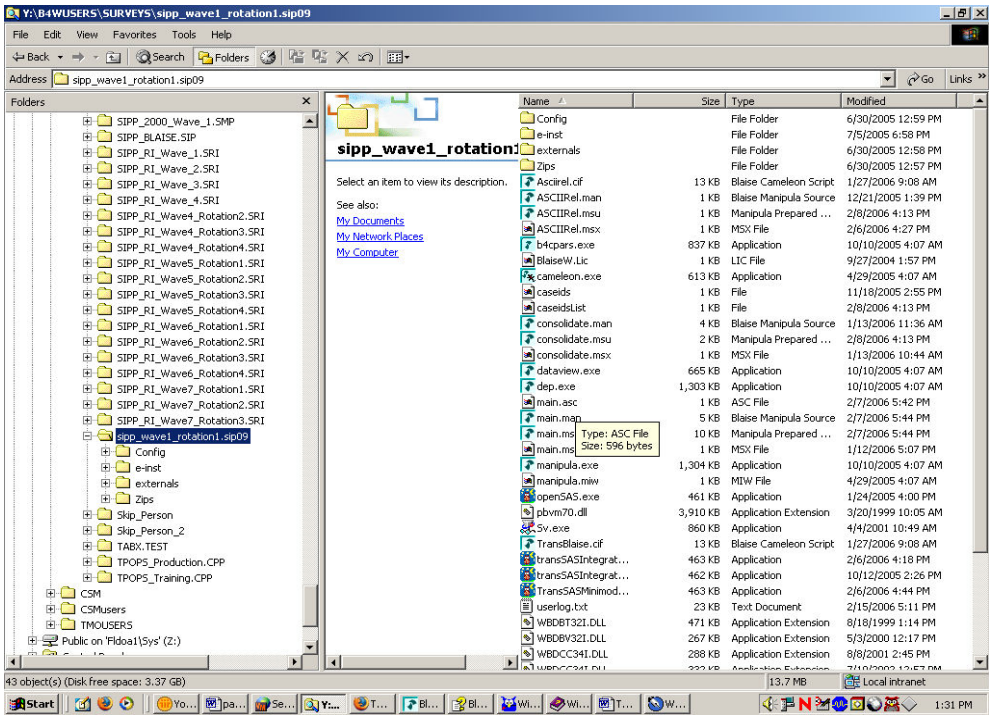
We have integrated all the scripts into one application and used the Blaise parser and executables so it can run in any environment (e.g., desktop, laptop, network etc.). We have seen the functionality of how TransSAS works. Next we will see how TransSAS is packaged and deployed on our network.

8. Packaging and Deployment

The TransSAS application was created using WinBatch, which is a very powerful general-purpose programming language. Its main purpose was to interact with Manipula and Cameleon. The WinBatch application was used to call the Manipula and Cameleon scripts using command line parameters and to allow user input. The WinBatch script is compiled to generate the TransSASIntegrated.exe. The command line parameter used is the name of the folder at the survey level for a project on the TMOusers Menu. Since we also conduct module testing, a similar executable is available for testing mini model output on the TMOusers Menu. The access to the TransSAS application for mini-models is the same as it is for integrated instruments. Testing output from mini models is important because it is one way data inconsistencies can be identified during early stages of development.

Here is some background information about the TMO standardized file names and directory structure. TMO uses 'inst' for its main project files, for instance the inst.bdb, inst.bmi, inst.bxi etc. These files are stored under the e-inst directory for the integrated instruments, and for the mini-models they are stored under the name of the mini-model. Figure 8.1 shows an example of our directory structure on the TMOusers Menu.

Figure 8.1



8.1 Scripts and Other Files Required for Integrated Instruments

As already mentioned, the required files will have to reside at the execution level of each survey. Table 8.1.1 shows the name of each program used and a detailed explanation for each. Mini-models have a similar structure except for consolidate.man file.

Table 8.1.1

Program name	Purpose	Output
AsciiRel.man	Creates the ASCII relational files The ASCIIrel.man is compiled and run inside the WinBatch script using the parser.	One data set per block in the instrument. E.g. A0, A02, A03.
ASCIIRel.cif	Creates the SAS code and dictionary files. The ASCIIRel.cif is compiled and run from inside the WinBatch script using the parser.	One data set per block is created.
Consolidate. man	At present we generate one record per caseid, (caseid.bdb). The consolidate.man combines the data for these caseids into one consolidated database.	Output is Inst.bdb
Main.man	Main.man is the script that launches the instruments on the TMOusers Menu. This is the screen where control is turned over to Manipula from the WinBatch interface. See Figure 7.1.	
Main.asc	Menu options are defined in this ASCII file.	Main.man uses this ASCII file as input.
Caseid	A text file with a list of caseids It displays what cases are available to pick.	

8.2 Executables Required for Running TransSAS on Integrated Instruments

Table 8.2.1

Programs	Purpose
B4cpars.Exe	Parser to compile the scripts
Cameleon.exe	Needed for running Cameleon
Manipula.exe	Needed for running Manipula
Sv.exe (Optional)	Needed to open SAS viewer
Wordpad.exe (Optional)	Needed to open SAS and dictionary files.
TransSASIntegrated.exe	Executable used by integrated instruments.

It is important to note that the scripts must be in sync with the Blaise data model. Hence, there is a need to compile them for every release of the instrument where the data structure has changed. This process can be time-consuming for developers. Taking some of these issues into consideration, we have automated these processes in the TransSAS application. Every time we run the TransSAS application, it will compile the scripts.

Note: The Manipula and Cameleon scripts will have to be compiled in the version of Blaise that is being used by the survey instrument. It is important for programmers to place the correct parser and executables for that particular version at the execution level or the root directory.

9. Code Examples

The TransSAS executable is called from within Main.man. This is our intermediate step before turning control over to WinBatch. A call to the WinBatch executable is placed inside the Main.man, which launches the Maniplus interface (see Figure 7.1).

- **Main.man**
  Add the MenuID to the main.asc file to select the TranSAS program.
  Use the same MenuID number to condition inside the code. Change the survey name without adding any spaces.

```
IF MenuID = '3.01' THEN
 Reslt := run('TransSASIntegrated.exe Maricopa_County')
ENDIF
```

Figure 9.1

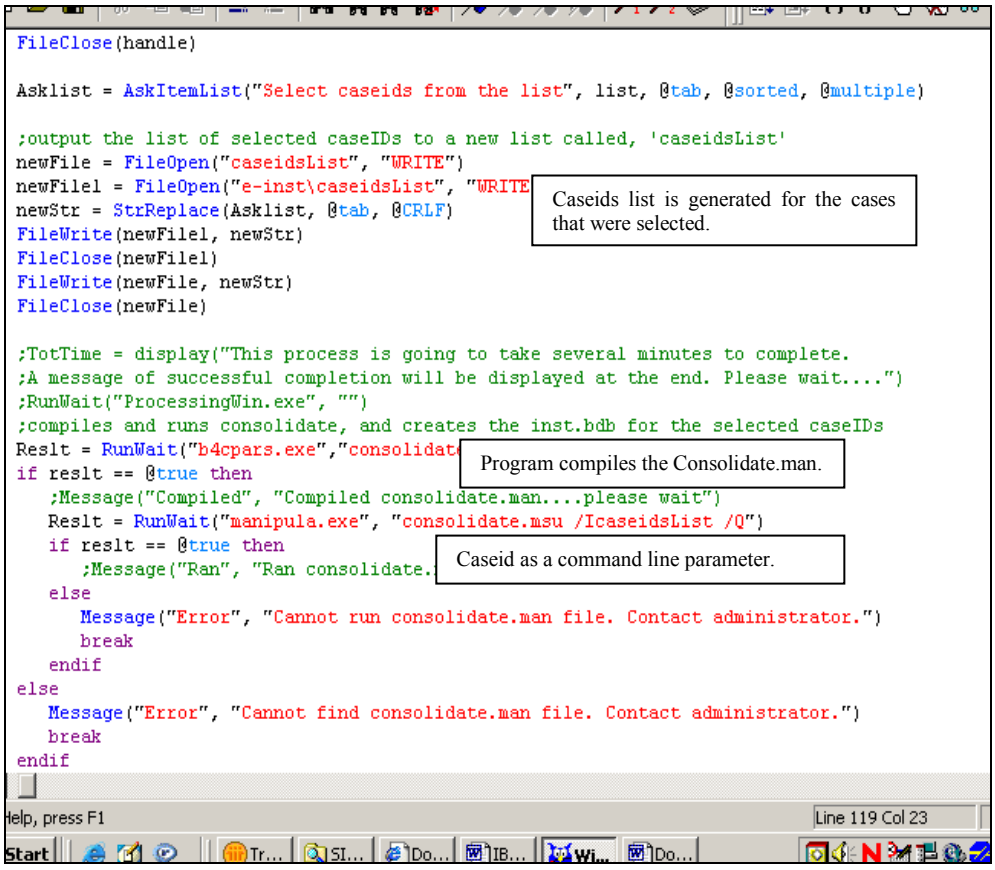


Figure 9.1 shows the WinBatch code, which was used to build TransSAS. The list of caseids is the first display (see Figure 7.3). We have also implemented functionality in the program, where a new list of caseids is stored in as ASCII file called caseidlist. The caseidlist file helps testers identify cases for which data has been extracted.

10. Snippet of the SAS code (Blaiseoutput.SAS)

- Below is an example of the SAS macro inside the Blaiseoutput.SAS file that was created by Cameleon.

```
%MACRO A76(dataname); /* corresponds to the data file number generated by
                        the Manipula script.*/
DATA &dataname.;      /* SAS dataset name  */
    Input
        FPRIMARY  $ 8.
        INSTANCENUMBER  $ char5
```

- The example below shows the output for a subroutine in the instrument, which is not part of the database. It is useful to identify and remove subroutines.

```
/*FileNumber:= 76   Block Name = RoundToNearest*/
```

- The location of the ASCII Relational files for the INFILE statement inside the SAS macros has been parameterized. See sample code from Cameleon below.

```
[Spaces:= 2]
[/]%MACRO[:3][FileNumber](dataname,InputPath,InputFile,ext=[FileNumber],
                                                    FILESTAMP=0);
[/]DATA &dataname.; /*FileNumber:= [BLOCKNUMBER] [Block Name=blockname]*/
[Spaces:= Spaces + 2]
[:Spaces]INFILE "&InputPath.&InputFile..&ext" MISSOVER PAD
                                                    LRECL=RecordLength];
```

11. Current Limitations

It is time-consuming to run TransSAS on the network. Things that affect the performance of TransSAS may be the following:

- Number of cases selected.
- Size of the instrument
- The network speed.
- Consolidating the individual databases.
- Compiling the scripts
- Cases locked by another user

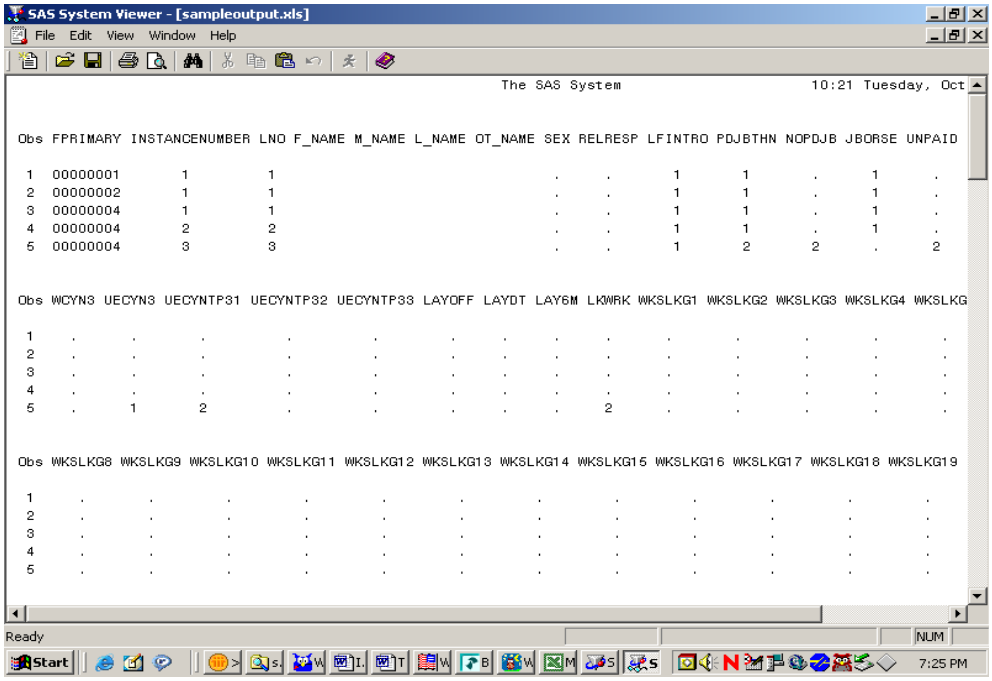
12. Benefits

- The program is run in "quiet" mode. Therefore, it does not interfere with other work. The users will see a message box that asks them to choose a destination folder for the output once the program is completed.
- Compiling scripts is beneficial as it saves on setup time for authors: They do not have to compile to keep the scripts in sync. TMO has instruments in several different versions of Blaise. Therefore, compiling the scripts is highly recommended.
- The users are given the ability to choose their output destination folder.
- Reduces user errors: The program automatically deletes the SAS, dictionary and data files from the network drive. By doing this, we can avoid users from accidentally deleting any other important file from the network.

13. Sample SAS Output

Figure 13.1 shows a sample of SAS output ".lst" files which was generated by running one of the SAS datasets which was output from Cameleon.

Figure 13.1



Some sponsors have tested their instruments using TransSAS and have also run the SAS macros to verify the data. So far we have got positive feedback from them. Table 13.1 is an example of the SAS output created by the sponsors using SAS code that was generated by Cameleon.

Table 13.1

b06_varname	n05_varname	b06_length	n05_length	b06_type	n05_type	b06_blockname	b06_filename	Outcome
AFUEL	AFUEL	1	1	C	C	BlkBAC	b12	OK
AFUELQ	AFUELQ	8	1	N	C	BlkBAC	b12	OK
AFUELQSP	AFUELQSP	30	30	C	C	BlkBAC	b12	OK
AFUELV	AFUELV	1	1	N	C	BlkBAC	b12	OK
	AFUR		1		C			Drop
AGE	AGE	8	3	N	N	BRELATION	a62	OK

14. Conclusion

The TransSAS application can be deployed in a multi-user testing environment, but it is not mandatory for everyone to set up their environment in this manner. Since it is an executable, it can be deployed on a client-server environment, packaged on a laptop, or placed on a FTP site for external sponsors to download. Other languages such as VB, Delphi, C, and C++ can be substituted for WinBatch to run TransSAS.

To summarize, implementing TransSAS for instrument testers and data processors to retrieve test output from Blaise instruments provides the following benefits:

1. The application provides an easy way for sponsors to access their output data to ensure better data quality.
2. The application improves time efficiency for programmers and data processors.
3. The application is low maintenance and cost effective

15. Future Plans

There is room to improve the TransSAS application. We are looking at making some enhancements in the future. We can build an application similar to TransSAS using different object-oriented languages where we can explore the following possibilities:

1. Deploying it in client-server environment.
2. Develop the application using the Blaise Component Pack along with any language that supports the Microsoft COM (Common Object Model) specification like Visual Basic, Visual Basic for Applications, Delphi, C++, et cetera.
3. To improve the speed of the application we have considered the following methods:
   a) Selectively compile the scripts by incorporating the version info and the date changed into the instrument so that it can be part of the meta data. The WinBatch program can then be modified to read this and selectively compile the scripts by comparing the dates.
   b) Explore the possibility of modifying the Cameleon/Manipula scripts, such that it eliminates creating empty data files.
   c) Lastly, create a batch program to run this application behind the scenes

References

Guan, Xiaodong & McGowan, Howard R. (2001) : Output Processing for Consumer Expenditure Survey. Presentation for the International Blaise Users Conference.

Jellema, T.R. (2001) : Blaise to OLE-DB-Relational Data Migration? Yes! Presentation for the International Blaise Users Conference.

Schou, Roger. (2003) : Creating Code to Convert Blaise to ASCII for SAS Input (Using the Blaise API within Visual Basic). Presentation for the International Blaise Users Conference.

Wensing, Fred. (2004) : Instrument assembly, documentation and release. Presentation for the International Blaise Users Conference.

Acknowledgements

I would like to acknowledge the following people from our TMO Authoring staff. I would like to thank Karen, Rob, Tom, and Ed for reviewing and editing this 10th International BLAISE Users Conference presentation paper. I also would like to thank Yogeeta who has jointly programmed the TransSAS application and has provided valuable input into this paper. I would like to thank Emily for providing valuable input about our TMO testing environment.

Karen Bagwell, U.S. Census Bureau, [Assistant Division Chief]
Rob Wallace U.S. Census Bureau, [Team Leader]
Thomas Spaulding, U.S. Census Bureau, [Team Leader]
William Edward Dyer, U.S. Census Bureau [Section Chief]
Yogeeta Purohit, Contractor at U.S. Census Bureau [Programmer/Analyst]
Emily Johnson, U.S. Census Bureau [Computer Specialist]