Creating Code to Convert Blaise to ASCII for SAS Input (Using the Blaise API within Visual Basic)

Roger Schou, National Agricultural Statistics Service, USA

1. Background

At the National Agricultural Statistics Service (NASS), we often have a need to create an ASCII file from a Blaise data set, in order to read it into SAS. We very rarely want to read out every field from the Blaise data set, so a very tedious, hand-coding effort was required to create a Manipula setup as well as a SAS input statement. The introduction of the Blaise API provided tools to create SASOut.EXE, an application using Visual Basic to easily create these two files that were once so labor intensive. Throughout this paper, this application will be referred to as SASOut.

SASOut has much of the same basic functionality as the Manipula Wizard, which will be part of Blaise 4.6. However, there are many additional functions available with SASOut.

2. NASS Standards

SASOut assumes that certain NASS standards are being used. If the standards should change, minor changes to the Visual Basic code would be necessary. The following discussion addresses the standards that SASOut uses.

2.1 NASS Folder Structure

At NASS, we have a standard folder structure for all CASIC applications. The instrument is placed in the h:\hqapps\casic*surveyfolder* folder, the survey specific Manipula setups are in the h:\hqapps\casic*surveyfolder*\Manipula folder, the Blaise data set is in h:\data\casic*surveyfolder*, and any generated ASCII output is placed in the h:\data\casic*surveyfolder*\asciiout folder, where *surveyfolder* reflects the survey. These four folders play a role in SASOut, because the optional SAS input statement will be placed in the instrument folder. The Manipula setup will be placed in the Manipula folder. This Manipula setup will look for the Blaise data set in the data set folder and place the output in the asciiout folder.

2.2 NASS Field Description

A coding standard has been introduced into the Blaise instrument to ease the processing steps needed with SASOut. The standard is simply the addition of a field description to each field to be included in the output. The syntax is SASVar: followed by the variable name (the colon is required). An example of a field in Blaise follows:

```
FIELDS
    ID “Please enter the id.” / “SASVar: RecordID” : 1..5000.
```

This standard is by no means required, it just simplifies the SASOut processing steps. SASOut will use the Variable name following the SASVar: notation, if it is present in the instrument. If it is not present, SASOut will prompt to use the Blaise field name for each field chosen. There is also a button available that will use the Blaise field name as the Variable name for all of the remaining selected fields.

2.3 Output Files

There are potentially two output files created by SASOut: SASOutVB.MAN and *Instrumentname*VB.SAS, where *Instrumentname* is the name of the Blaise instrument.
The Manipula file is always created, and the SAS file is optionally created. Two small examples of these output files are included at the end of this paper.

2.4 Field Conversions

It may be desirable to convert some fields when they are output. SASOut addresses some of these conversions. “Don’t Know” and “Refusal” responses may be converted to –1 for all of the non-string fields (integer, real, and enumerated fields). If the field to be converted to a –1 is a one-digit field, then it will automatically be output as a two-digit field, in order to handle the minus sign. This conversion is an all or none conversion, meaning that it will be true for all non-string fields.

Any string field may optionally be converted to an integer. This conversion is based on each individual string field. A prompt will appear for each string field, asking whether it should be converted to an integer.

Date type and time type fields are output as strings.

SASOut will not output open fields. It simply gives a warning that it cannot output open fields, and then continues.

3. The SASOut Interface

The buttons and windows of SASOut are dynamic, only appearing when they are valid. It was written to be straightforward, while obtaining all the necessary information to create the Manipula setup and the optional SAS input statement.

3.1 Selecting the Blaise Meta File

When invoking SASOut, a window is displayed that has only one active green button. It is used to identify the Blaise data model to be used. Clicking on the green button will allow the user to point to the Blaise meta file (BMI file). Once the BMI file has been selected, a prompt asking for the Blaise data set name appears. The name of the Blaise data set should be entered without an extension.

Figure 1. Blaise Data Model

3.2 Selecting Fields

Once the Blaise data set name has been identified, the structure browser of the data model is displayed. The structure is collapsed to the data model level and may be manually expanded block by block. The *Expand Full Tree* button is also available, which will expand the full structure tree. However, there is no Collapse Full Tree button, so once the full tree is expanded, it can only be collapsed manually. A *Load from Files* button will only be visible if the selected fields and variable names were saved from a previous session. This will be discussed later, but it is worth pointing out that the view (or field selections) may be saved and reloaded at a later time.

Figure 2. Select Fields from Structure Browser

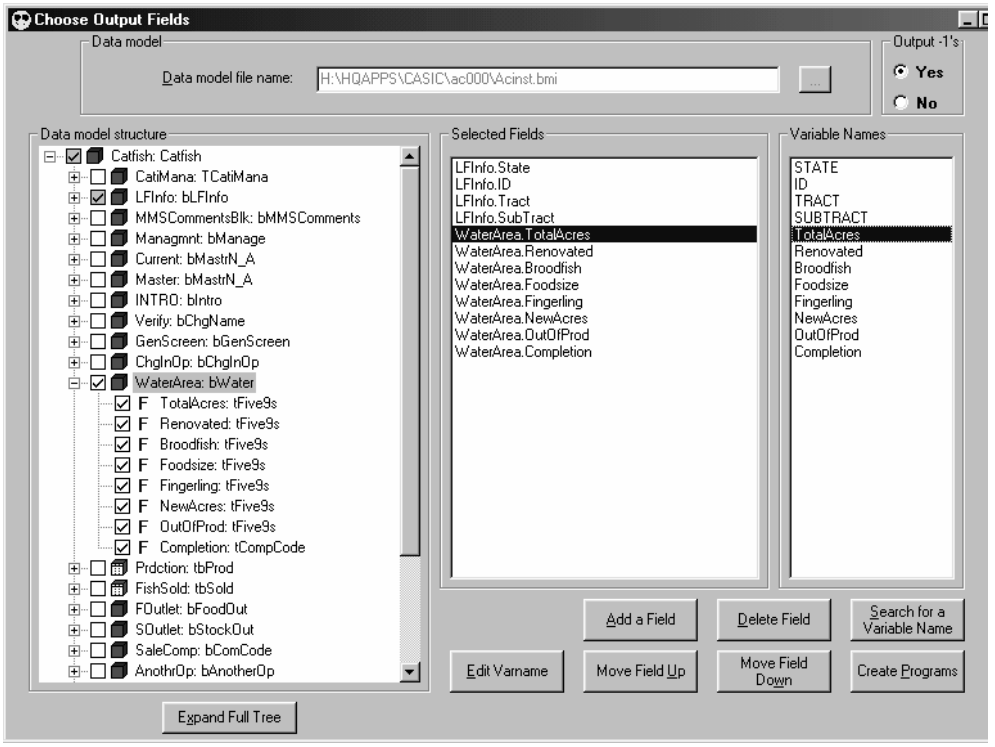
When the first field is selected from the structure browser, a prompt to output a –1 for all "Don't Know" and "Refusal" responses for all non-string fields appears. Once that question has been answered, it will be noted at the top right of the window. This answer may be changed by clicking on either the *Yes* or *No* option. The *Create Varnames* button also becomes visible. The selection of fields may continue, until all desired fields have been selected. The order in which the fields are selected will determine the order the variable names will be listed; however, this order may be changed.

3.3 Creating Variable Names

The next step is to click on the *Create Varnames* button. All selected fields that have the SASVar: notation in the field description in the instrument will appear with the designated name in the *Selected Fields* and *Variable Names* list boxes. The full Blaise field name will be in the *Selected Fields* list box, and the variable name will be in the *Variable Names* list box. If the selected field has a string type, a prompt will be displayed asking if the value should be converted from a string to an integer. For every selected field that does NOT have the SASVar: notation, a prompt will be displayed asking for the variable name to be used for the selected field. This variable name will default to the Blaise field name, but may be changed. In fact, there is an *Accept ALL Blaise Field Names* button available which will use the Blaise field name for all of the remaining selected fields. Once the *Create Varnames* button is clicked, it will no longer be visible, and the structure browser will also be disabled.

There are two new buttons that become visible. The *Create Programs* button will create the Manipula setup and the optional SAS input statement. This button will be discussed later. The second new button is the *Add a Field* button. If a field was omitted from the original selection, clicking the *Add a Field* button will enable the structure browser so more fields may be selected. After clicking the *Add a Field* button, the *Create Programs* button will become invisible, and the *Create Varnames* button will become visible again.

Figure 3. Manipulating the Variable Names



3.4 Manipulating the Variable Names

Once the *Create Varnames* button has been clicked, it is possible to select a field in the *Selected Fields* or the *Variable Names* list boxes. Selecting a field in either of these list boxes will cause up to five new buttons to be visible. The *Edit Varname* button offers the ability to change the selected variable name. The *Delete Field* button will remove the selected variable name from the list, and thus it will not be output. The *Search for a Variable Name* button provides the ability to search the *Variable Names* list box. The search can be from the beginning of the list down or from the selected field down. The search is a "whole word search," but it is not case sensitive. The *Move Field Up* button will be visible for all fields except the first one, and it will move the selected field up one. The *Move Field Down* button will be visible for all fields except the last one, and it will move the selected field down one.

3.5 Creating the Output Files

Once all of the desired fields to be output have been selected and the variable names have been assigned, the *Create Programs* button should be clicked. Once clicked, SASOut will check the variable names to insure that each one is unique and contains no spaces. If it finds a duplicate name or a variable name that contains a space, a message is displayed and the programs are not created. Once SASOut determines that there are no duplicate variable names and none contain a space, a few additional prompts are displayed.

3.5.1 Saving the Selected Fields

The first prompt presents the option to save the selected fields and variable names. Clicking *Yes* will cause SASOut to save a series of little ASCII files so that the selected fields and corresponding variable names may be reloaded at a later time. Clicking *No* will not save the selected fields and variable names, and the next time someone selects the same instrument, they will have to reselect all of the desired fields and assign the variable names again.

3.5.2 SAS Input Statement Option

The second prompt presents the option to create a SAS input statement in addition to the Manipula setup. Clicking *No* will result in only the Manipula setup being created.

3.5.3 Delimited File Option

The third prompt presents the option to create an ASCII delimited file instead of a fixed field file. Clicking *Yes* indicates that an ASCII delimited file is desired, and a dialog box will be displayed, where the delimiter of choice may be selected.

3.5.4 Naming the SAS Data Set

The fourth prompt will only appear if a SAS input statement was requested by answering *Yes* to the second prompt (the SAS Input Statement Option). This last prompt asks for the name of the SAS data set to be created. This will be the name used in the Data step in SAS.

4. Conclusion

Examples of a Manipula setup and a SAS input statement, both generated by SASOut follow. These selected examples were shortened in order to conserve space, but it is evident that these programs could become very large.

Before SASOut existed, creating these large programs was very time consuming. Cameleon was used to generate the SAS input statement, but all of the fields in the instrument were included in the input statement. This file then had to be pared down by hand. Next, a corresponding Manipula setup had to be written by hand that would be used to create the ASCII file for SAS to read. This entire process took hours to complete, and the programs often contained typographical errors. With the new SASOut application, the same process can be completed in 15 minutes to ½ hour depending on the number of variables desired in the output. Without the Blaise API, SASOut would have never been possible.

Example 1. SASOutVB.MAN – A generated Manipula setup

```
SETTINGS
   DATEFORMAT = MMDDYY
USES
   InstFmt '\HQAPPS\CASIC\ac000\Acinst'

   DATAMODEL SASOutFmt
      FIELDS
         STATE : INTEGER[2]
         ID : INTEGER[9]
         TRACT : INTEGER[2]
         SUBTRACT : INTEGER[2]
         TotalAcres : INTEGER[5]
   ENDMODEL

Inputfile BlzData : InstFmt ('\data\CASIC\ac000\ac000', BLAISE)

Outputfile SASOut : SASOutFmt ('\data\CASIC\ac000\asciiout\BlzData.ASC',
ASCII)
   SETTINGS
      SEPARATOR = ','

MANIPULATE

   SASOut.STATE := BlzData.LFInfo.State
   SASOut.ID := BlzData.LFInfo.ID
   SASOut.TRACT := BlzData.LFInfo.Tract
   SASOut.SUBTRACT := BlzData.LFInfo.SubTract
   IF (BlzData.WaterArea.TotalAcres = DK) OR
      (BlzData.WaterArea.TotalAcres = RF) THEN
      SASOut.TotalAcres := -1
   ELSE
      SASOut.TotalAcres := BlzData.WaterArea.TotalAcres
   ENDIF

   SASOut.WRITE
```

Example 2. *InstrumentName*VB.SAS – A generated SAS input statement

```
TITLE Catfish;
DATA CATFISH;
INFILE '\data\CASIC\ac000\asciiout\BlzData.ASC' LRECL=62 DELIMITER=',';
INPUT
    STATE                   /* LFInfo.State */
    ID                              /* LFInfo.ID */
    TRACT                   /* LFInfo.Tract */
    SUBTRACT                        /* LFInfo.SubTract */
    TotalAcres              /* WaterArea.TotalAcres */
;
LABEL
    STATE = 'LFInfo.State'
    ID = 'LFInfo.ID'
    TRACT = 'LFInfo.Tract'
    SUBTRACT = 'LFInfo.SubTract'
    TotalAcres = 'WaterArea.TotalAcres'
;
```