Electricity Forecasting Full Circle

- Database Creation
- Libname Functionality with Excel
- VBA Interfacing

Allows analysts to develop procedural prototypes

By: Kyle Carmichael

Disclaimer

The entire presentation and associated views are solely that of the analyst and not of ATCO Power.

Permission has been granted by ATCO Power for the analyst to share some examples from his work at ATCO Power.

Forward-Looking Information:

- Certain statements contained in this presentation may constitute forward-looking information. Forward-looking information is often, but not always, identified by the use of words such as "anticipate", "plan", "estimate", "expect", "may", "will", "intend", "should", and similar expressions. Forward-looking information involves known and unknown risks, uncertainties and other factors that may cause actual results or events to differ materially from those anticipated in such forward-looking information. The Corporation believes that the expectations reflected in the forward-looking information are reasonable, but no assurance can be given that these expectations will prove to be correct and such forward-looking information should not be unduly relied upon. Any forward-looking information contained in this presentation represents the Corporation's expectations as of the date hereof, and is subject to change after such date. The Corporation disclaims any intention or obligation to update or revise any forward-looking information whether as a result of new information, future events or otherwise, except as required by applicable securities legislation.

Agenda

- Database Creation
  - Looping Macro (Based on Time)
  - Import engine (XML)
  - DOS commands (X Commands)
- Program Integration
  - Libname functionality (With Excel)
  - SAS Add-in (Controlling with VBA)
- Example Procedure

The Database – Looping Macro

```
%macro database_loop;

%let starttime = %sysfunc(date());            /*Define Start (Today)*/
%let endtime = %sysevalf(&start + 7);/*Define End (Today+7 Days)

%do %until( &time > %sysevalf(&endtime)); /* Start of Macro*/

%let time = %sysfunc(date());                 /* Current time

%XML_IMPORT_MACRO

data  _null_ ;                                /* Sleep for 30 seconds*/
Call sleep(5,1);
Run;

%end;                                         /*End of Macro*/
```

1. Determine Filenames

```
%macro get_filenames(location);
  filename pipedir pipe   "dir ""%unquote(&location)"" /b " lrecl=32767;
  data filenames;
    infile pipedir truncover;
    input files $char1000.;
  run;
%mend;

%get_filenames(location);
```

2. Count the Number Files

```
proc sql noprint;
select count(files) into: count
from filenames;
quit;
```

3. Record the Filenames as macro variables

```
proc sql;
select files into :File1 - :File&count
from filenames;
quit;
```

4. Use a SAS engine to import the data (XML, Proc Import, ect.)

```
%do i=1 %to &count;

libname in xml (location\&&file&i);

data scrape&i;
set in.anything  /*Only the in is important, the name does not matter*/
run;
```

5. Merge the different data files into a file in a permanent library

%if the first file %then %do → Create initial permanent data table (ex: sasuser.input)
%else %do → Append to the created table (proc append)

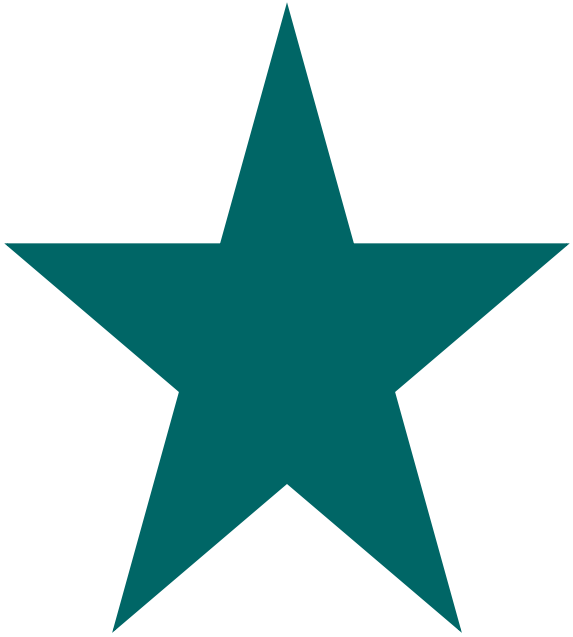
6. Use X Command to archive and clear xml files and proc datasets to clear work library

```
X Copy "location\&&file&i" "New_Location\NewName"

X Del /f /s /q "location\&&file&i"

proc datasets lib=work;
delete scrape&i;
run;
```

Data Is Now In Database



The Libname Statement

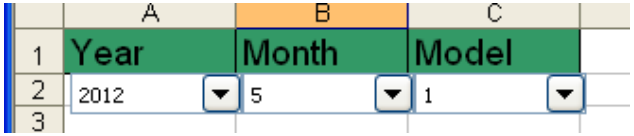
1. Pull in filtering parameters from an excel workbook

Filename parms excel **location**\Parameters.xls

Data parms;

Set parms.'Sheet1$'n;

Run;



2. Convert the parameters to macro variables

```
Proc sql;
select year into: year
from parms;
run;
```

```
Proc sql;
select month into: month
from parms;
run;
```

```
Proc sql;
select model into: model
from parms;
run;
```

3. Filter the data and run analysis – histogram, distribution, means, etcetera

```
Data filtered;
Set sasuser.input (where=(year = &year month=&month model=&model));
Run;
proc means data=filtered;
by year month model;
output out = mean_output mean= max= p75= \autoname;
run;
```

4. Send the data to an output excel file

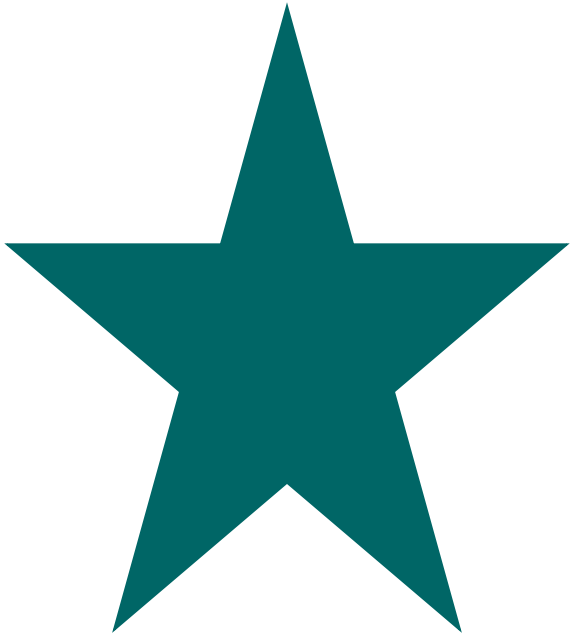
```
libname output excel "location\Model_Output.xls"

Proc datasets lib=output;
Delete exceloutput;
Run;

Data output.exceloutput
Set mean_output;
Run;

Libname output clear;
```

Data Is Now In Excel



VBA Interfacing

- VBA can control SAS Stored Processes

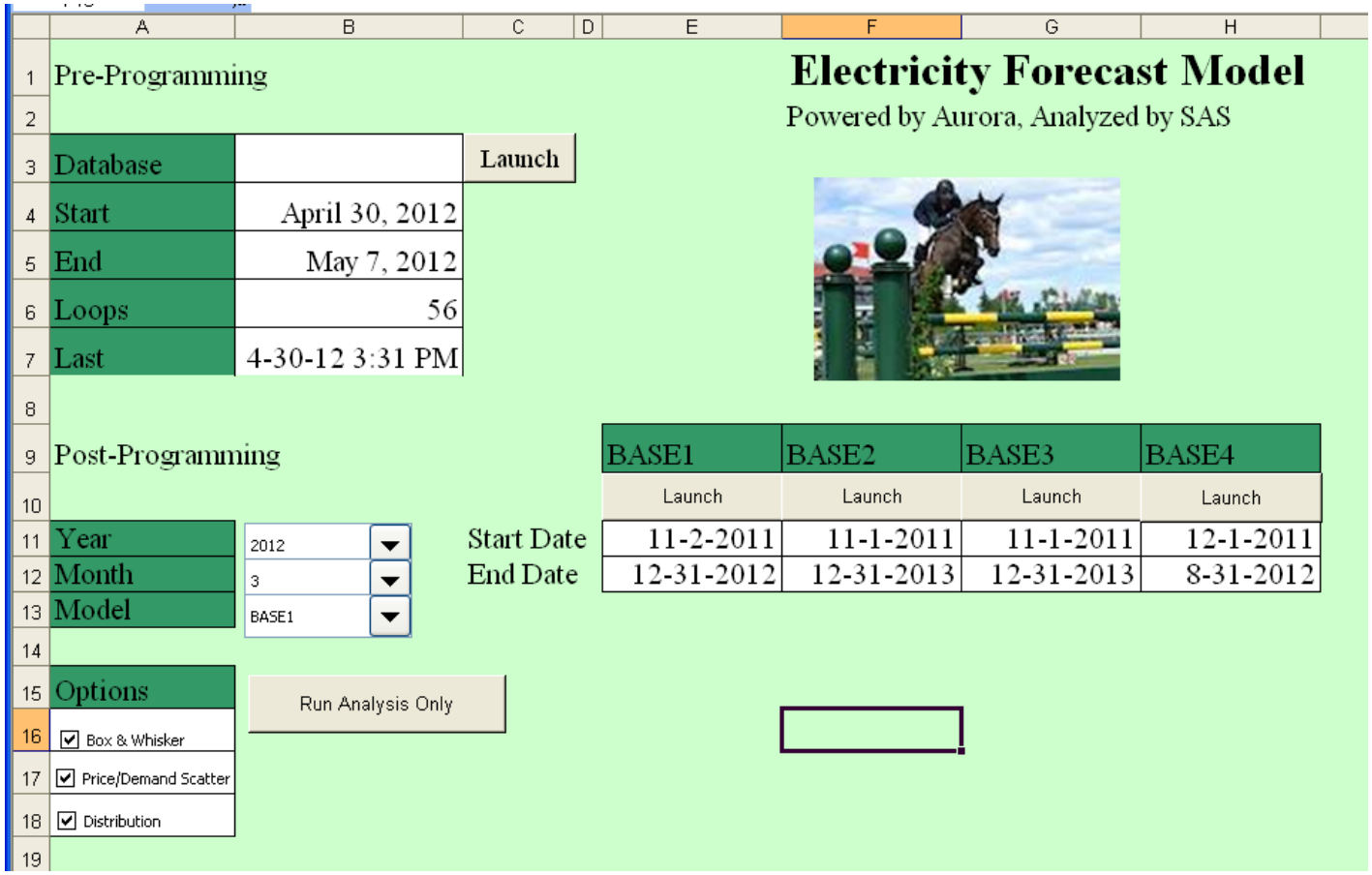
```
Dim SAS As Object
Set SAS =Application.COMAddIns.Item("SAS.OfficeAddin.Loader.ConnectProxy").Object
SAS.Refresh "Analysis_Stored_Process"
```

- Only the name of the stored process has to be used
- In Visual Basic Editor you may have to go to Tools→ References to update library for SAS

VBA SAS Function to see if connection is working

```
Dim SAS As Object
Set SAS =Application.COMAddIns.Item("SAS.OfficeAddin.Loader.ConnectProxy").Object
SAS.HelloWorld
```

The Control Panel & Process



Pre-Programming

Database	
Start	April 30, 2012
End	May 7, 2012
Loops	56
Last	4-30-12 3:31 PM

Launch

Electricity Forecast Model

Powered by Aurora, Analyzed by SAS

Post-Programming

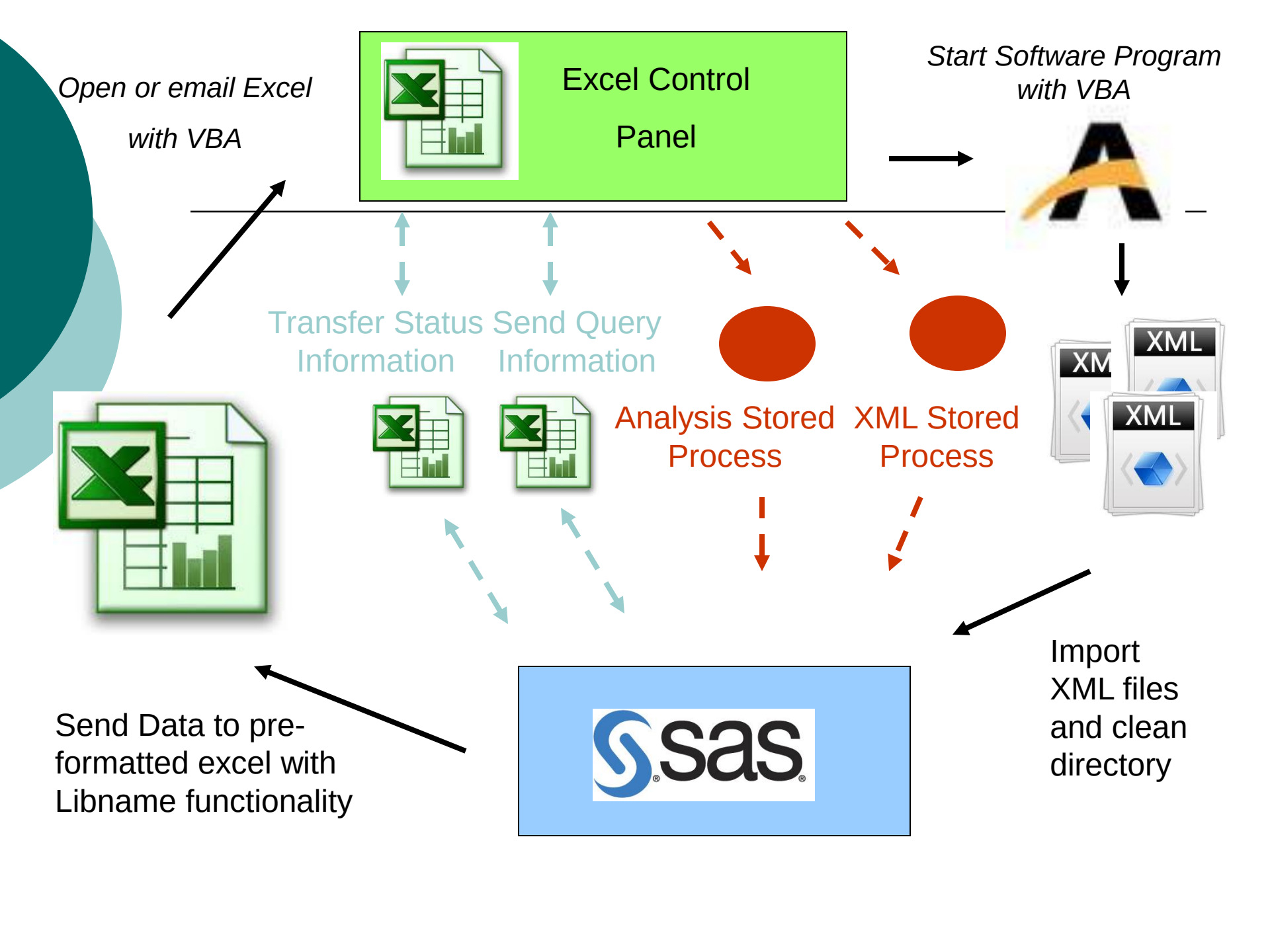
Year	2012
Month	3
Model	BASE1

	BASE1	BASE2	BASE3	BASE4
	Launch	Launch	Launch	Launch
Start Date	11-2-2011	11-1-2011	11-1-2011	12-1-2011
End Date	12-31-2012	12-31-2013	12-31-2013	8-31-2012

Options

- ☑ Box & Whisker
- ☑ Price/Demand Scatter
- ☑ Distribution

Run Analysis Only


Open or email Excel with VBA
Excel Control Panel
Start Software Program with VBA
Transfer Status Information
Send Query Information
Analysis Stored Process
XML Stored Process
XML
XML
XM
Import XML files and clean directory
Send Data to pre-formatted excel with Libname functionality
sas

Questions & Contact Information

Kyle Carmichael
Business Analyst
ATCO Power

Phone:(403) 245 -9972
E-mail: kyle.carmichael@atcopower.com