

Sending SAS® Data Sets and Output to Microsoft Excel®

Imelda C. Go, South Carolina Department of Education, Columbia, SC

ABSTRACT

For many of us, using SAS and Microsoft Excel together is inevitable. This paper is useful for a beginner who wants to know how the two can be used together and how to send SAS data and/or output to Excel. A number of SAS features facilitate the sharing of data between the two. There are tools in SAS Enterprise Guide (EG) that help send output data into Excel. With SAS/ACCESS, PROC EXPORT can convert a SAS data set into an Excel file. The Output Delivery System (ODS) offers a number of destinations, such as the Excel destination that facilitates the transfer of SAS output into Excel. Using Dynamic Data Exchange (DDE) is a way for a user to populate a preexisting or preformatted Excel file with SAS data. SAS traffic lighting (color coding of SAS output elements) in a report can also be applied to SAS output such that the traffic lighting will appear in Excel.

Note: The following examples are based on using SAS EG in a Windows 7 environment with Microsoft Office 2010. The DDE examples were based on using PC SAS.

According to Microsoft, there are more than 1.2 billion people who use some kind of Microsoft Office product (<https://news.microsoft.com/bythenumbers/planet-office>) in 140 countries and 107 languages. Microsoft Excel is definitely widely available, and it is not any wonder that transferring data into Excel is an essential skill for a SAS programmer.

These short examples will hopefully inspire the beginner to incorporate the techniques into their work and to read more about the many different options available to SAS users.

PROC EXPORT

Here is code that exports a SAS data set into a CSV file, which can be subsequently imported into Excel. The use of the code below does not require SAS/ACCESS Interface to PC Files.

```
proc export data=finalexport
  outfile="c:\test.csv"
  dbms=dlm
  replace;
  delimiter=',';
run;
```

The finalexport data set is exported to a CSV file (c:\test.csv). The DBMS= option is dlm, which means the output file is a delimited file. The REPLACE option is used to indicate that if the output file already exists, the file will be replaced/overwritten.

Here is code that exports a data set into an Excel file. The Excel file format and many other file formats are available with SAS/ACCESS Interface to PC Files.

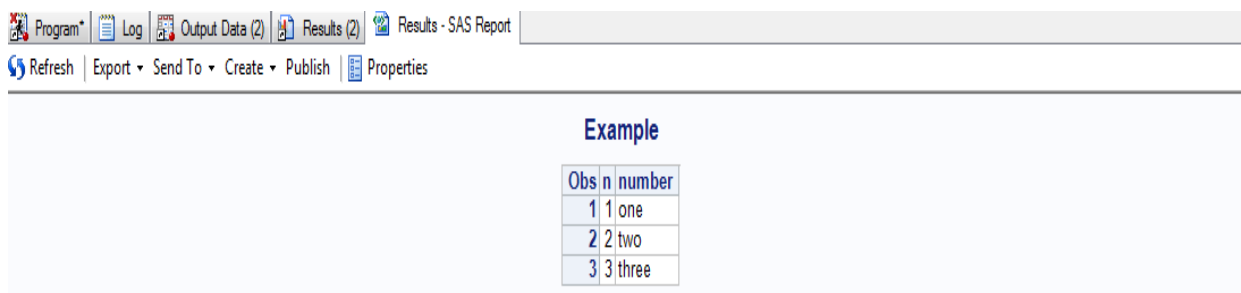
```
proc export data=finalexport
  outfile='c:\test.xlsx'
  dbms=excel replace;
  sheet='data';
run;
```

The finalexport data set is exported to an Excel (c:\test.xlsx). The DBMS= option is excel, which means the output file is an Excel file. The REPLACE option is used to indicate that if the output file already exists, the file will be replaced/overwritten. The data are written into a worksheet called data.

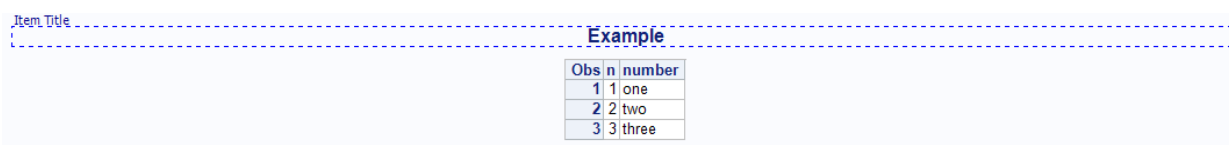
MANUALLY COPYING OBJECTS IN SAS ENTERPRISE GUIDE (EG) AND PASTING THEM INTO EXCEL

When inside the SAS Report window, users can select objects and copy them. They can then go to an Excel session and paste the object into Excel. To select an object, first click on it. Selected objects will appear outlined in dashed instead of solid borders. To copy the selected object, right-click on it and then select Copy. Another way is to press CTRL+C while the object is selected.

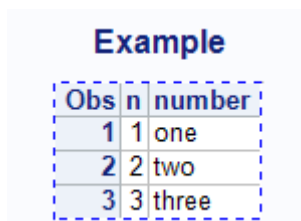
Here is an example of PROC PRINT output with a title in the SAS EG Results window:



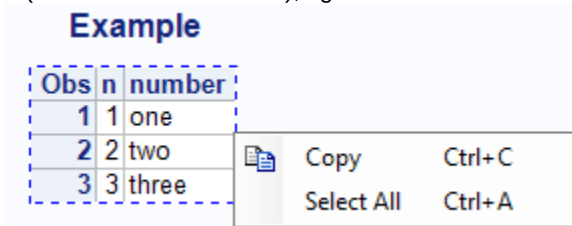
When you click on the title Example, you will see the following where the title is now inside a box with a dashed perimeter, which indicates you have selected the Item Title. Note that the Item Title now appears at the upper lefthand corner of the box.



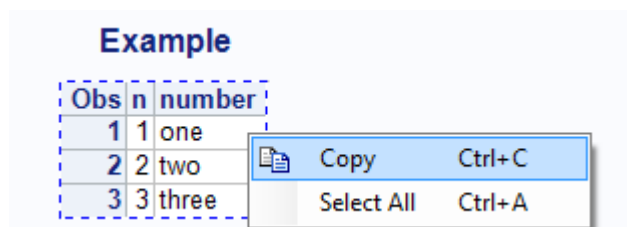
When you click on the PROC PRINT output, you will see the following where the solid border line is now a dashed border line.



While the object remains selected (dashed border is visible), right-click on the selected object to see dropdown options:



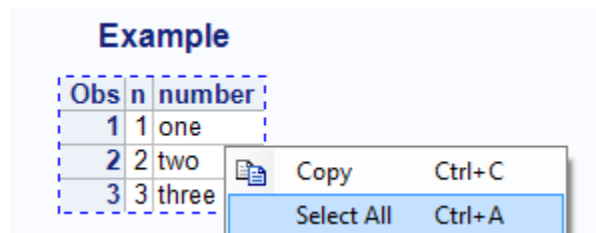
Select Copy in the dropdown options:



After copying the desired object, open Excel, put your cursor in the cell where you want to paste the object, and then paste with Ctrl+V. In the example below, the cursor was placed in cell B2 before pasting the object.

	A	B	C	D
1				
2		Obs	n	number
3		1	1	one
4		2	2	two
5		3	3	three

The dropdown list has Select All as an option.



When you choose Select All, you will select all objects in the Results window:

Item Title	Example												
	<table><tr><th>Obs</th><th>n</th><th>number</th></tr><tr><td>1</td><td>1</td><td>one</td></tr><tr><td>2</td><td>2</td><td>two</td></tr><tr><td>3</td><td>3</td><td>three</td></tr></table>	Obs	n	number	1	1	one	2	2	two	3	3	three
Obs	n	number											
1	1	one											
2	2	two											
3	3	three											

Clicking anywhere in the Results window and then right-clicking to choose Select All will select all the objects in the window. Another way to select all the objects is to click anywhere inside the Results window and then press Ctrl+A to select all the contents.

You are not limited to selecting one object or all objects from the Results window. You may select more than one object by selecting an object in the Results window and then selecting other objects by pressing on the Ctrl key at the same time.

Here is an example of three out of four objects selected (the second title is not selected):

Item Title

Example

Obs	n	number
1	1	one
2	2	two
3	3	three

Page Break

Example

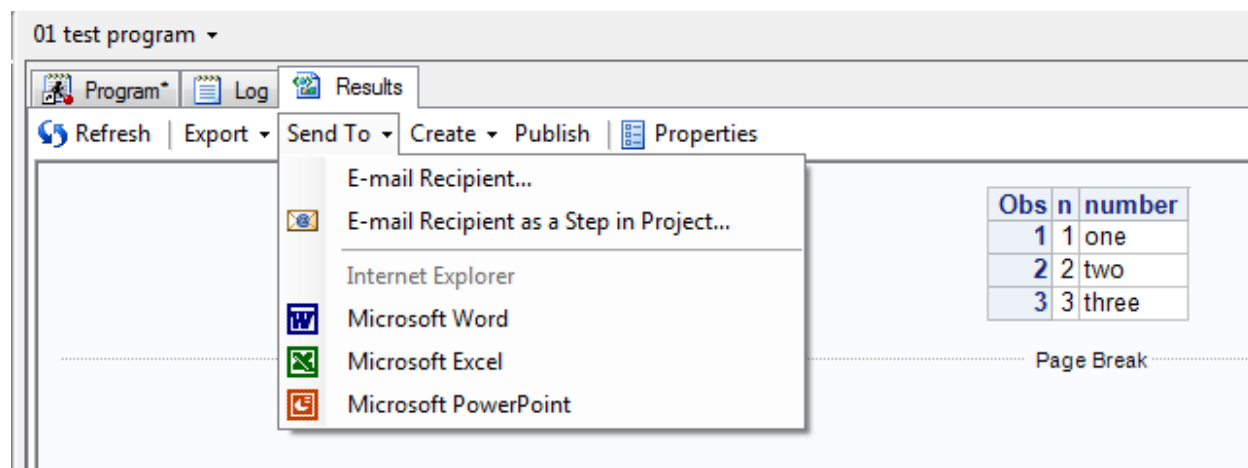
Obs	n	number
1	1	one
2	2	two
3	3	three

Page Break

Selecting an object while pressing the Ctrl key acts like a toggle. The first selection of an object selects it. The second selection of the same object while the Ctrl key is pressed deselects the object.

SAS ADD-IN FOR MICROSOFT OFFICE: MANUALLY SENDING RESULTS TO A MICROSOFT PRODUCT

Users have the option to install the SAS Add-In for Microsoft Office, which works together with SAS Enterprise Guide. The Add-In will enable users to send the contents of the Results window to Microsoft products (Word, Excel, PowerPoint).



ODS EXCEL DESTINATIONS: SENDING SAS RESULTS INTO EXCEL WORKBOOKS WITH SAS CODE

Use ODS Excel destination to send SAS output into one or more worksheets in an Excel file.

```
ods excel  
FILE="&dirname\output.xlsx";  
  
ods excel  
options(sheet_name='Summary1'  
embedded_titles='yes' );  
  
proc print data=data1;  
run;  
  
ods excel  
options(sheet_name='Summary2'  
embedded_titles='yes' );  
  
proc print data=data2;  
run;  
  
ods excel  
options(sheet_name='Summary3'  
embedded_titles='yes' );  
  
proc print data=data3;  
run;  
  
ods excel close;
```

The first ODS statement contains the FILE= option, which specifies the name of the Excel file.

The next ODS EXCEL statements are strategically placed so that:

- the PROC PRINT output for data1 is written to the Summary1 worksheet;
- the PROC PRINT output for data2 is written to the Summary2 worksheet;
- and the PROC PRINT output for data3 is written to the Summary3 worksheet.

The ODS EXCEL CLOSE statement ends the transmission of output to the ODS EXCEL destination.

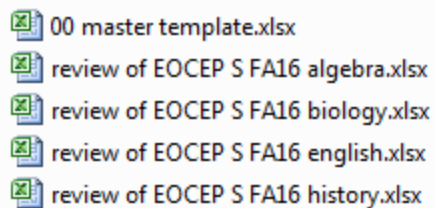
DYNAMIC DATA EXCHANGE (DDE): SENDING SAS DATA INTO EXCEL WITH SAS CODE

Note: Not all SAS installations support the use of Dynamic Data Exchange. Unfortunately, the author no longer has access to Dynamic Data Exchange through SAS. Therefore, the sample code with X commands below could not be tested. However, all the code was functioning when they were heavily used some time ago.

Using Dynamic Data Exchange and PROC EXPORT will write data into Excel files. Here are some of the differences between the two methods:

Dynamic Data Exchange (DDE)	PROC EXPORT
- Writes data into an existing Excel file. - The existing Excel file may contain pre-existing objects (e.g., formulas, charts) and may also have preformatted qualities (e.g., conditional formatting of cells, pages are already preset for printing). - The Excel file has to be open when SAS writes data into it. - The user must specify exactly which worksheets and which cells SAS will write the data into. - The user must program which variable values go into the Excel worksheet cells.	- Creates a data file specified by the user. - SAS takes the data set and outputs the data according to the order of the variables in the data set. - SAS can only write one data set into one Excel file. Therefore, the Excel file will only contain one worksheet.

Here is an example of how Dynamic Data Exchange can be useful. The list of variables show five Excel workbooks:



The first file is a template (00 master template.xlsx), which contains two worksheets with conditional formatting and which have been pre-formatted for printing. The other four files are the same as the template except that the file names are different.

Here is a screenshot of one of the worksheets in the template:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1			2016-17 FA16				* = answer key	tor(s) had a larger %							* = answer key		X=Distractor(s) had a positive value					
2			Item ID	Item Num	KEY	Count	P-value (Check c.4 or c.95)	%A	%B	%C	%D	Flag	%Omit	%Mult	Corr A	Corr B	Corr C	Corr D	Corr Flag	Item Discrim (Check c.2)	Grade + DIF	Ethnic + DIF
3																						
4																						
5																						
6																						
7																						
8																						
9																						
10																						

Here is an example of SAS code that populates one of the worksheets. This code is taken from a macro called processsubject that contains code to populate both worksheets in the workbooks.

```
%macro processsubject(subject,dsname,filename)

filename excel dde "Excel|[%filename]MC All Groups!r3c1:r8000c50" notab;

data _null_;
set &dsname;
file excel dlm='09'x dsd;
    put group
    form
    ItemID
    ItemPos
    KEY
    Count
    ItemDiff
    txtpctA
    txtpctB
    txtpctC
    txtpctD
    commentpct
    PCTOmits
    PCTMults
    txtCorrA
    txtCorrB
    txtCorrC
    txtCorrD
    commentcorr
    ItemDisc
    GenderDIF
    EthnicDIF
    modeDIF ;
run;
%mend processsubject;
```

The goal is to populate four Excel workbooks with data from four subjects: algebra, biology, English, and history. The SAS programmer can manually create these four Excel workbooks and then open all of them in Excel. Dynamic Data Exchange will only work if the Excel workbooks you are attempting to write into are open.

Here is an example of macro calls to the processsubject macro that processes data by subject. It will take the data for each subject (specified by dsname=data set name macro variable) and send the data to a workbook (specified by filename macro variable).

```
%processsubject(subject=English,
dsname=dt.English,filename=review of EOCEP S fa16 English.xlsx);
%processsubject(subject=Algebra,
dsname=dt.Algebra,filename=review of EOCEP S fa16 algebra.xlsx);
%processsubject(subject=history,
dsname=dt.History,filename=review of EOCEP S fa16 history.xlsx);
%processsubject(subject=biology,
dsname=dt.Biology,filename=review of EOCEP S fa16 biology.xlsx);
```

After the program runs, we see worksheets that have been populated by SAS for you. Once the data are loaded, we see below the conditional formatting (yellow cells) that has been activated with the loading of the data since the template was already programmed for conditional formatting.

2216-17 FA16				* = answer key tor(s) had a larger %				* = answer key				X=Distractor(s) had a positive value						
Item ID	Item Num	KEY	Count	P-value (Check c.4 nr >.95)	zA	zB	zC	zD	zE	zF	zG	zH	zI	zJ	zK	zL	zM	zN
1/ 330 A2Z14129	D1-01	W	9598	0.8780	2.6	6.5	3.0	* 87.8										
1/ 330 MSCZ277	D1-02	Z	9598	0.7670	9.5	5.1	* 76.7	8.6										
1/ 330 A2Z14232	D1-03	Z	9598	0.6340	13.8	8.8	* 63.4	13.8										
1/ 330 A2Z14254	D1-04	S	9598	0.7200	* 72.0	7.5	6.0	14.4										
1/ 330 A2Z15241	D1-05	W	9598	0.6120	6.2	14.6	17.8	* 61.2										
1/ 330 MSCZ232	D1-06	S	9598	0.8300	* 83.0	6.0	6.3	4.5										
1/ 330 A2Z14212	D1-07	Z	9598	0.6620	11.1	12.2	* 66.2	10.4										
1/ 330 MSCZ714	D1-08	W	9598	0.7650	15.7	3.1	4.6	* 76.5										
1/ 330 A2Z14271	D1-09	S	9598	0.5740	* 57.4	24.9	11.5	6.1										
1/ 330 A2Z15297	D1-10	S	9598	0.5800	* 58.0	26.7	9.3	6.0										
1/ 330 A2Z14272	D1-11	X	9598	0.3530	9.6	* 35.3	15.4	39.6	X									
1/ 330 A2Z14293	D1-12	S	9598	0.5630	* 56.3	22.5	14.6	6.5										
1/ 330 A2Z14246	D1-13	X	9598	0.5150	13.2	* 51.5	23.6	11.5										
1/ 330 A2Z14255	D1-14	S	9598	0.5090	* 50.9	32.1	7.5	9.4										
1/ 330 MSCZ498	D1-15	X	9598	0.6850	9.9	* 68.5	4.2	17.3										
1/ 330 A2Z14263	D1-16	X	9598	0.4730	15.0	* 47.3	17.1	20.6										
1/ 330 A2Z14297	D1-17	Z	9598	0.6660	8.5	19.1	* 66.6	5.7										
1/ 330 MSCZ712	D1-18	W	9598	0.6810	6.7	9.1	16.0	* 68.1										
1/ 330 MSCZ527	D1-19	W	9598	0.7140	6.9	7.2	14.2	* 71.4										
1/ 330 A2Z14211	D1-20	S	9598	0.5230	* 52.3	24.6	15.6	7.4										
1/ 330 MSCZ632	D1-21	W	9598	0.4870	30.2	17.9	3.0	* 48.7										
1/ 330 MSCZ386	D1-22	Z	9598	0.6670	6.4	23.5	* 66.7	3.4										
1/ 330 A2Z14279	D1-23	W	9598	0.4610	11.3	18.6	23.7	* 46.1										
1/ 330 MSCZ693	D1-24	W	9598	0.5830	6.0	14.3	21.2	* 58.3										
1/ 330 A2Z14111	D1-25	X	6	0.0000	33.3	* 0.0	50.0	16.7	X									
1/ 330 A2Z14237	D1-26	S	9598	0.4520	* 45.2	18.2	23.4	13.1										
1/ 330 A2Z14248	D1-27	Z	9598	0.4640	11.3	30.1	* 46.4	12.1										
1/ 330 A2Z14252	D1-28	S	9598	0.4350	* 43.5	21.2	13.9	21.2										
1/ 330 MSCZ263	D1-29	X	9598	0.4970	11.3	* 49.7	16.9	21.9										
1/ 330 A2Z15236	D1-30	Z	9598	0.4440	10.8	23.6	* 44.4	21.1										
1/ 330 A2Z14224	D1-31	X	9598	0.4790	17.1	* 47.9	13.5	21.5										
1/ 330 MSCZ559	D1-32	S	9598	0.5150	* 51.5	27.7	11.8	8.9										
1/ 330 MSCZ297	D1-33	S	9598	0.5340	* 53.4	17.0	22.1	7.3										

In the above example, the four worksheets were created manually in the Windows system by copying and pasting the template four times and then renaming each of the four copies to the desired names. It is possible to program even that part of the work. You can get this done by using the X command to submit Windows commands. The X command enables you to submit a Windows command through SAS.

Let us suppose that you have to create a set of Excel workbooks for 100 schools. That is, data for each of the 100 schools will be loaded in a copy of the template, such that the Excel workbook's name for each school will include the school's name.

Here is an outline of what can be coded with SAS macros to automate this process:

- Open Microsoft Excel using an X command.
- Have a copy of your Excel workbook template that you want to use for each data subset of interest. Open the template in Excel using SAS. Transfer the data for the subset into the Excel workbook using DDE.
- Save the template populated with new data as an Excel workbook with a new name.
- Create another macro to generate the 100 macro call lines you need to execute the process 100 times. This assumes that you have a data set that contains the necessary information for each of the 100 macro calls

<pre> %macro startxl; filename sas2xl dde 'excel system'; data _null_; file sas2xl; run; options noxwait noxsync; %if &syserr ne 0 %then %do; x "c:\program files\microsoft office\office\excel.exe"; data _null_; x=sleep(10); run; %end; data _null_; file sas2xl; put '[open("c:\template.xlsx")]'; run; %mend startxl; </pre>	This macro starts/opens Excel from within SAS. The X command is used to run Excel. After Excel is opened, the template is opened.
<pre> %macro writedata(school); filename xl dde "Excel [template.xlsx]by group!r4c1:r60000c80" notab lrecl=9000; data _null_; set &school; file xl dlm='09'x dsd; put school lastname firstname; run; data _null_; length test \$100.; file sas2xl; put '[error(false)]'; test='[save.as("c:\school data for' "&school" '.xlsx")]'; put test; put '[file.close(false)]'; run; %mend writedata; </pre>	The macro parameter for the writedata macro is school, which is the SAS data set name for the data you want to send to an Excel file. The FILENAME statement with DDE option identifies the template and the cells in the template that data should be written into. In the example, data can be written into the grid of cells with corners at (1) 1st column and 4th row, and at (2) the 60,000th row and 80th column. The first DATA _NULL_ step takes the data in the data set specified by the school macro variable and puts it into the template. The second DATA _NULL_ step saves the template loaded with the school data into a file. The file has the school's name in the file name.

<pre> proc freq data=one; tables school/noprint out=macrodata; filename runthis 'C:\school macro calls.txt'; data _null_; length test \$100.; set macrodata; file runthis; test='%writedata(school=' trim(school) ')'; put test; run; </pre>	For this example, creating a SAS data set using PROC FREQ will provide a data set (macrodata) with the 100 school names. For this example, the school name is the same as the data set name for each school. The FILENAME statement assigns runthis to the text file (c:\school macro calls.txt) that will contain all 100 macro calls needed to create the 100 files. The DATA _NULL_ step takes the information from the macrodata data set and writes the macro call for each school. These macro calls are written into the runthis file. <pre> %writedata(school=ABC); %writedata(school=CDE); . . . %writedata(school=ZZZ); </pre>
<pre> %startxl; %include runthis; </pre>	The first statement starts the Excel session and opens the template (%startxl). The %INCLUDE statement includes the contents of the runthis file for execution in the SAS program.

TRAFFIC LIGHTING IN SAS AND IN EXCEL

Traffic lighting is a powerful reporting technique. Here is sample code that shows you how the same traffic lighting looks like in the SAS Results window and in an ODS EXCEL destination.

<pre> proc format; value color 1='red' 2='yellow' 3='green'; ods excel file="&dirname\example.xlsx"; ods excel options(sheet_name='color' embedded_titles='yes' sheet_interval='none'); proc print data=sample; var n ; var n / style={background=color.}; var n / style={foreground=color.}; run; ods excel close; </pre>	Use PROC FORMAT to specify a format that corresponds to colors. This example uses the traditional red, yellow, green colors for traffic lighting. A value of 1 will correspond to red, 2 to yellow, and 3 to green. The variable n is listed once in three separate VAR statements. The first VAR statement renders n values using defaults and has no traffic lighting. The second VAR statement renders the traffic lighting color as a background color for cell shading. The third VAR statement renders the traffic lighting color as a foreground color for text.
--	--

This is how it looks like in the SAS Results window:

Obs	n	n	n
1	1	1	1
2	2	2	2
3	3	3	3

Here is how it looks like in the Excel workbook produced by ODS EXCEL.

	A	B	C	D
1	Obs	n	n	n
2	1	1	1	1
3	2	2	2	2
4	3	3	3	3
5				

CONCLUSION

This paper serves as a quick overview on how to quickly transfer SAS data or output into Microsoft Excel. Some methods are done manually, while other methods are programmatically executed. With so many different ways to do things and so many different options to choose from in SAS, there is bound to be a method or configuration that will suit your needs.

TRADEMARK NOTICE

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Microsoft, Encarta, MSN, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other brand and product names are trademarks of their respective companies.

CONTACT INFORMATION

Imelda C. Go
Office of Assessment
SC Department of Education
1429 Senate St.
Columbia, SC 29201