ABSTRACT
To create a report, you need requirements and data. Your hardest task may be finding the data and learning how to use it properly. Two programs are presented here to aid your quest. One finds all of your SAS datasets, and the other shows you what your datasets look like on the inside. SAS dictionary tables provide basic information about the structure and contents of your data. The MEANS and SQL procedures profile the dataset, and show values and statistics for any reporting column.

INTRODUCTION
I was writing a report on a specific column that belonged to several SAS datasets. To find the reporting datasets, I searched the columns dictionary table in SAS. Then I generalized that solution because we may need to answer similar requests for finding data and producing reports. We had no useful metadata documents, and no profiling tools like SAS® Data Integration, DataFlux® dfPower® software, or JMP® software. So I created a profiling tool that used only the Base SAS® language.

There are many good papers that cover all the basic facts about SAS dictionary tables. But few papers describe sophisticated uses of SAS meta-information to solve tough problems like data discovery. This paper describes a practical SAS solution that may guide your deeper dive into the data repository.

USES OF DATASET PROFILE INFORMATION
If you have any doubts about the correctness of your data, then you need to profile the data, learn its quirks, and decide how you should use it. You may want to save that knowledge to help others understand how to use the data. Knowledge about data is imperfect under these conditions:

- Datasets may exist outside of the data warehouse.
- Metadata documents may be missing, invalid, or incomplete.
- Dataset owners may not be available to provide advice on proper usage of the data.
- Keys may be unknown, and dataset joins may require guesswork and experimentation.
- Internal reorganizations, mergers, and acquisitions may create gaps in your data knowledge.

PRACTICAL APPLICATIONS OF DATASET PROFILE INFORMATION
These are actual applications of dataset profile information in my work:

- You want to analyze all the datasets that contain a particular column. Use the Columns spreadsheet from the first program, Dataset_Explorer.sas, and find all SAS datasets that have that column. See Figure 2.
- You have an address column, but you want additional information from a foreign key in the dataset. Use the Columns spreadsheet to find datasets with column names similar to any column in the dataset. See Figure 3.
- You want to update the data repository and add a column to all campaign datasets. Use the libnames file, created by Dataset_Explorer.sas, and make SAS libnames for all your datasets. Then use SAS dictionary tables to create the SAS statements that will add a new column to the datasets.
- You need to plan some data analysis, but you don’t know which columns have a sufficient number of non-missing analysis variables. Create a dataset profile using the second program, Dataset_Profiler.sas, which counts non-missing and unique values for every column. See Figure 4.

Here are some other possible uses of dataset profiles:
- Profile datasets over time, and compare profiles to find added, deleted, or changed columns.
- Compare detailed column profiles to see whether variables in different datasets may contain similar data.
- Use dataset ownership from Dataset_Explorer.sas to assign responsibility, and ask dataset owners to document the proper usage of the data.
- Use owner information and LDAP to assign responsibility to the department of the dataset owner. Then create and publish departmental measures for active ownership and proper dataset documentation.
- Use the dataset documentation to create meaningful labels for all variables. In that way, the SAS dataset documents itself and describes how to use its data. Good examples of this practice are the datasets supplied by SAS Institute. You can learn how to use the data by reading the column labels.
- With good SAS labels, you can use Proc Print to create an excellent data dictionary. This was done at GlaxoSmithKline for their PRx data warehouse, and a documentation specialist maintained the labels. Note that label maintenance is quick and easy to do using PROC DATASETS with the MODIFY statement.
THE SAS DICTIONARY DEFINES THE ARCHITECTURE

The SAS dictionary contains a wealth of information about SAS itself and about every customization or definition that you have created. The SAS System has a self-defining data architecture like a relational database. One table defines all of the dictionary tables, and each of those tables defines all of the SAS objects and their current instances. The master table in SAS is the Dictionary_Dictionaries table.

This master dictionary defines the dictionary tables for each SAS object, such as options, formats, ODS styles, macros, catalogs, libnames, and datasets. All such information is metadata, or data about the data.

To view the definitions of all of your SAS dictionary tables, you can submit this code:

```sas
proc sql noprint;
%*-- Create Describe commands for each table in the Master Dictionary. --;
select distinct 'describe table ' || compress('DICTIONARY.' || memname)
   into :Desc_Dictionary_Tables
   separated by "; "
   from Dictionary.Dictionaries
   order by memname;
%*-- Print the description for each table in the Master Dictionary. --;
&Desc_Dictionary_Tables;
quit;
```

For example, these are some well-labeled dictionary tables in my local SAS System:

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Dataset Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOGS</td>
<td>Catalogs and catalog-specific information</td>
</tr>
<tr>
<td>CHECK_CONSTRAINTS</td>
<td>Check constraints</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>Columns from every table</td>
</tr>
<tr>
<td>CONSTRAINT_COLUMN_USAGE</td>
<td>Constraint column usage</td>
</tr>
<tr>
<td>CONSTRAINT_TABLE_USAGE</td>
<td>Constraint table usage</td>
</tr>
<tr>
<td>DICTIONARIES</td>
<td>DICTIONARY tables and their columns</td>
</tr>
<tr>
<td>ENGINES</td>
<td>Available engines</td>
</tr>
<tr>
<td>EXTFILES</td>
<td>Files defined in FILENAME statements, or implicitly</td>
</tr>
<tr>
<td>FORMATS</td>
<td>Available formats</td>
</tr>
<tr>
<td>GOPTIONS</td>
<td>SAS/Graph options</td>
</tr>
</tbody>
</table>

*Figure 1. Some Dictionary Tables in the SAS System*

EXAMPLE USES OF THE COLUMNS SPREADSHEET

The Columns spreadsheet looks like a stripped down PROC CONTENTS listing. It contains information on all columns of all SAS datasets in your system. After you open the CSV file, and save the data as an Excel 2007+ workbook, you can filter the columns to find your analysis datasets. This example shows the results of Excel filtering on `name` to show any column that contains the string, `curr_rate`. The Directory column shows the location of the dataset, and Memname shows the SAS dataset name. See *Figure 2*.

The next example demonstrates looking for a foreign key to get more information about your data. The name column was filtered for any string that contained LPO. The SAS dataset, WFF_75MR, was eventually used for the analysis, after looking up the dataset owner and confirming proper usage of the data. See *Figure 3*. 
<table>
<thead>
<tr>
<th>Directory</th>
<th>memname</th>
<th>name</th>
<th>type</th>
<th>length</th>
<th>modate</th>
</tr>
</thead>
<tbody>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI07A</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:29:53</td>
</tr>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI07B</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:29:57</td>
</tr>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI07C</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:30:03</td>
</tr>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI08A</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:30:08</td>
</tr>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI08B</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:30:12</td>
</tr>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI08C</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:30:17</td>
</tr>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI09A</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:30:22</td>
</tr>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI09B</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:30:23</td>
</tr>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI10A</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:30:29</td>
</tr>
<tr>
<td>/cart/dart/contact_hist/dev/master_file</td>
<td>REFI10B</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>27OCT09:14:30:39</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>GROUP2_CONV</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>06OCT09:12:51:50</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>GROUP2_CONV_REFI09B</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>18SEP09:09:36:18</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>GROUP2_GOVT</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>06OCT09:12:51:50</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>GROUP2_GOVT_REFI09B</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>18SEP09:09:36:19</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>NEWREFI10A_NOCONTROL_OCT6</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>08OCT09:18:35:26</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>ORIGREFI10A_NOCONTROL_OCT6</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>08OCT09:20:53:59</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>ORIGREFI10B_NOCONTROL_OCT13</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>13OCT09:11:12:04</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>ORIGREFI10B_NOCONTROL_OCT19</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>21OCT09:14:31:11</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>ORIGREFI12A_NOCONTROL_NOV02</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>04NOV09:14:04:26</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>ORIGREFI12B_NOCONTROL_NOV23</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>03NOV09:08:22:05</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>ORIGREFI12B_NOCONTROL_DEC07</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>09DEC09:16:54:48</td>
</tr>
<tr>
<td>/cart/dart/refi</td>
<td>REFI01A_FINAL_W_CTRL</td>
<td>curr_rate</td>
<td>char</td>
<td>200</td>
<td>22DEC09:19:53:07</td>
</tr>
</tbody>
</table>

**Figure 2. The Columns Spreadsheet – Filtered on name contains curr_rate**

<table>
<thead>
<tr>
<th>Directory</th>
<th>memname</th>
<th>name</th>
<th>type</th>
<th>length</th>
<th>modate</th>
</tr>
</thead>
<tbody>
<tr>
<td>/cart/dart/data_repository</td>
<td>ALL_WFF_50MR_1006_EXCLUSIONS</td>
<td>LPO</td>
<td>num</td>
<td>8</td>
<td>07OCT09:12:37:59</td>
</tr>
<tr>
<td>/cart/dart/data_repository</td>
<td>ALL_WFF_50MR_1006_EXCLUSIONS</td>
<td>New_LPO</td>
<td>num</td>
<td>8</td>
<td>07OCT09:12:37:59</td>
</tr>
<tr>
<td>/cart/dart/data_repository</td>
<td>ALL_WFF_50MR_1006_EXCLUSIONS</td>
<td>Old_LPO_No</td>
<td>num</td>
<td>8</td>
<td>07OCT09:12:37:59</td>
</tr>
<tr>
<td>/cart/dart/data_repository</td>
<td>ALL_WFF_75MR_1007_INCLUSION</td>
<td>new_LPO</td>
<td>num</td>
<td>8</td>
<td>08OCT09:11:00:53</td>
</tr>
<tr>
<td>/cart/dart/data_repository</td>
<td>WFF_75MR</td>
<td>New_LPO</td>
<td>num</td>
<td>8</td>
<td>04NOV09:16:21:01</td>
</tr>
<tr>
<td>/cart/dart/MAP_Maint/op10</td>
<td>HPC_BRANCHES</td>
<td>LPO</td>
<td>num</td>
<td>8</td>
<td>2JAN10:15:06:33</td>
</tr>
<tr>
<td>/cart/dart/MAP_Maint/op10</td>
<td>MAIL_OP10_W_SEEDS</td>
<td>LPO</td>
<td>num</td>
<td>8</td>
<td>11FEB10:16:24:53</td>
</tr>
<tr>
<td>/cart/dart/non_responder10</td>
<td>NONR1002A_SEEDS_FINAL_HPCX</td>
<td>LPO</td>
<td>num</td>
<td>8</td>
<td>02FEB10:14:04:38</td>
</tr>
<tr>
<td>/cart/dart/non_responder10</td>
<td>MAIL_NBR_W_SEEDS</td>
<td>Old_LPO</td>
<td>num</td>
<td>8</td>
<td>92DEC09:14:40:49</td>
</tr>
<tr>
<td>/cart/dart/non_responder10</td>
<td>MAIL_NBR_W_SEEDS</td>
<td>Old_LPO</td>
<td>num</td>
<td>8</td>
<td>92DEC09:14:40:49</td>
</tr>
<tr>
<td>/cart/dart/non_responder10</td>
<td>RR8_FINAL_MAILFILE_W_SEEDS_SUPP1</td>
<td>LPO</td>
<td>num</td>
<td>8</td>
<td>02DEC09:17:02:41</td>
</tr>
<tr>
<td>/cart/dart/non_responder10</td>
<td>RR8_FINAL_MAILFILE_W_SEEDS_SUPP1</td>
<td>Old_LPO</td>
<td>num</td>
<td>8</td>
<td>97DEC09:18:28:16</td>
</tr>
<tr>
<td>/cart/dart/non_responder10</td>
<td>RR8_FINAL_MAILFILE_W_SEEDS_SUPP2</td>
<td>LPO</td>
<td>num</td>
<td>8</td>
<td>92DEC09:14:40:49</td>
</tr>
<tr>
<td>/cart/dart/non_responder10</td>
<td>RR8_FINAL_MAILFILE_W_SEEDS_SUPP2</td>
<td>Old_LPO</td>
<td>num</td>
<td>8</td>
<td>97DEC09:18:28:16</td>
</tr>
<tr>
<td>/cart/dart/non_responder10</td>
<td>PDQ_FINAL_MAILFILE_W_SEEDS_SUPP1</td>
<td>LPO</td>
<td>num</td>
<td>8</td>
<td>92DEC09:12:01:34</td>
</tr>
<tr>
<td>/cart/dart/non_responder10</td>
<td>PDQ_FINAL_MAILFILE_W_SEEDS_SUPP1</td>
<td>Old_LPO</td>
<td>num</td>
<td>8</td>
<td>97DEC09:12:01:34</td>
</tr>
</tbody>
</table>

**Figure 3. The Columns Spreadsheet – Filtered on name contains LPO**
THE DATASET EXPLORER PROGRAM

Dataset_Explorer.sas provides information about all SAS datasets in a directory tree. The SAS program performs these tasks:

1. Find all SAS datasets in the directory tree, and also get the dataset ownership and permissions.
2. Create SAS libnames for every directory that contains any SAS datasets.
3. Query the SAS dictionary tables for information on all tables and all columns, using these tables:
   a. Dictionary.Tables, which contains information on all SAS datasets.
   b. Dictionary.Columns, which contains information on all SAS variables.
4. Create Excel and CSV reports that can be filtered and searched for relevant SAS datasets and variables.

The reports are automatically date tagged. If today is June 30, 2010, then the two reports would be:

1. Dataset_Explorer_20100630.xls – an Excel workbook with information on all SAS datasets and their file permissions.
2. Dataset_Explorer_Columns_20100630.xls – a CSV file with information on all SAS variables. Figures 2 and 3 show what the CSV file looks like after being converted to Excel.

CAVEATS ABOUT THE SAS PROGRAM ENVIRONMENT
The SAS program was written for PC SAS. The SAS data repository is on UNIX, and remote access is by SAS/CONNECT®. The code for remote UNIX is between the RSUBMIT and ENDRSUBMIT statements.

If you are not using SAS/CONNECT, then you would omit the RSUBMIT and ENDRSUBMIT statements, as well as the %sysinput macros. If your SAS repository is not on UNIX, then you would need an equivalent command to search a directory for all SAS datasets.

HOW THE PROGRAM WORKS
At the top of the program, two definitions control how the program works. Set these to appropriate values for your system:

```sas
/*-- Start at this UNIX tree --*/
%let UNIX_Tree = /cart/dart;
filename DSNs PIPE
   "cd &UNIX_Tree; /usr/bin/find &My_Dirs -name "*.sas7bdat" -exec ls -l {} \;" ;
```

The %let defines the directory tree to search. The DSNs filename opens a UNIX pipe that searches for any SAS dataset (*.sas7bdat) in the directories (&My_Dirs, which is a subset of all directories). For every SAS dataset that is found, a full listing (ls –l) is produced, so that ownership and permissions can be captured. The result of those commands go to a PIPE, which is read by a SAS data step. If you need further details, the complete program is well commented and available for download.

THE DATASET PROFILER PROGRAM

Dataset_Profiler.sas is a much more complex program than the previous one. It was developed for the same SAS environment, and uses more tricks to process the data. The Excel spreadsheet is tagged by a constant inside the program, and its name is Dataset Profiler_CH.xls.

The following explanations may be easier to follow if you see the final results of a heavily filtered excerpt in Figure 4.

This program profiles any SAS dataset, and shows values and statistics for all the report columns in the dataset. A report column is defined as any column that has fewer than 300 unique values, but you may want to adjust that number for your own system.

1. PROC SQL counts the number of rows, the number of missing values per column, and the number of unique values per column.
2. PROC MEANS summarizes the reporting columns, and produces a wide range of statistics for each unique value of each reporting column.

Figure 4 shows an excerpt from a dataset profile report. Column A has the name of the column, column B has the values for that column, and the other columns are statistics from the PROC MEANS output dataset. Note that all the possible column values are shown for the offer_type and offer_pct columns, along with statistics on each value of those columns. The autolabel option names the columns based upon the statistic, which is N, a count of non-missing values. For example, refl_payment_ben_flag has non-missing values only when offer_type equals "" or "P", in which case it has 55,876 and 55,295 respectively.
Figure 4. The Dataset Profiler Spreadsheet – a Heavily Filtered excerpt

HOW THE PROGRAM WORKS
Several programming tricks are used in this program, and they will be covered from top to bottom. Please ignore the references to REFI; the program was originally developed for that type of analysis.

The first pass of the data uses SQL in a two-step process. In the first step, the count variable query is created from the dictionary table for the analysis dataset:

```sas
%let Count_All_Vars =
proc sql noprint stimer;
%*
-- Count non-missing and unique rows in the dataset. ---------------------
%*
-- Prepare the SQL statements to perform the counts. ---------------------
%*
-- Handle issue where SAS variable name would be longer than 32 chars --
select "count(" || strip(Name)  || ") as N_"
    || substr(left(Name),1,min(29, length(strip(Name)))) ||
    ", count(distinct " || strip(Name) || ") as ND_"
    || substr(left(Name),1,min(29, length(strip(Name))))
into :Count_All_Vars separated by ", "
from dictionary.columns
where libname = upcase(&Refi_LibName) and memname = upcase(&Refi_DSN);
%put Count_All_Vars has &sqlobs rows;
quit;
```

In the second step, the query selection is executed by PROC SQL:

```sas
%syslput Check_Keys = &Check_Keys;
%syslput Count_All_Vars = &Count_All_Vars;
%syslput Means_DSN = &Refi_LibName2..&Refi_DSN;
rsubmit; /*$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$*/
proc sql noprint stimer;
%*
-- Perform the dataset counts prepared above. --;
create table key_values_0 as
    select count(*) as All_Rows, &Count_All_Vars
    from &Means_DSN;
%put Dataset has &sqlobs rows;
quit;
endrsubmit; /*$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$*/
```
You should know one SAS paradigm when reading my code. I use CARDS4 as a SAS comment statement because it skips over everything but 4 semi-columns in a row. In the code below, &Check_Keys skips SAS code up to the semi-colon line when it equals CARDS4, but executes the SAS code when &Check_Keys is blank:

```sas
/*-- If &Check_Keys ^= cards4, then the next SQL query runs --*/
data _null_;&Check_Keys;run; /* ... ... whatever code you like to be here ... ... */

The previous SQL query returns a single row of data about missing and unique counts. That data is transposed and used to create all the other basic counts for each column. Then a determination is made whether a column is a reporting column by this code:

```sas
if (Unique_Pct_All <= 0.1 and Unique < 300 and Count ^= 0) then Stats = "Y";output;
```

Further manipulation is performed on the list of reporting variables. The only important criteria for selection should be the code above, since both alpha and non-continuous numeric variables can be used as class variables for PROC MEANS. The variables lists for MEANS are produced by SQL, and these are the PROC statements:

```sas
rsubmit; /*$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$*/
/*-- Date formats cause some issues with some stats: TBD later --*/
proc means data=&Means_DSN &Mean_Drop_Num_Stmt missing noprint;
class &CH_Vars;types &CH_Vars;
output out=&Means_Output N= NMiss= max= min= mean= std= / autoname;
run;
proc download data=&Means_Output; run;
endrsubmit; /*$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$*/
```

The output data from the MEANS procedure is a sparse matrix because of the `types` clause, which limits the output to summary statistics on the variables by themselves. The statistic columns are to the right of the `_type_` column, and all the columns before `_type_` have only one column with non-missing data.

We want something like Figure 4, where the column name is on the left, then a specific value for that column, and finally all the statistics for that column value. Lots of computations are done to process the dataset so we can use the `coalesce` SQL function to get the desired results:

```sas
%put in the SQL code is a useful diagnostic to count the rows you select and make sure that you get correct results. There is a bit more processing and merging before the Excel reports are produced.

You will need some creativity when using the Excel reports. You should hide and filter columns to display what you looks interesting. You may want to print the results to guide your report development. I found these methods useful in creating or enhancing reports. Time was saved because I could see the data and the range of values for each column. I used the statistics to evaluate whether I had good enough data to create a report. Instead of spending my time running custom queries, I just browsed and selected what I liked.
CONCLUSION
These two SAS programs provided a great starting point for analytical reporting. They have been used in a real work environment to solve real problems. I believe this tool has many other uses in data quality, data audits, and data governance. The results give you a window into your datasets, and use minimal system resources.

Like any new program, a lot will be learned as these programs get into the field and solve real business problems. I am interested in your results and your suggestions. Please let me know what you think.

ACKNOWLEDGMENTS
Thanks to Brent and Jessy for letting me spend a little time on this project.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:

Phillip Julian
Bank of America
101 S. Tryon St.
Charlotte, NC 28255
(980) 387-0507
julianp@acm.org
www.acm.org/~julianp

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