Paper 080-2010

Using Data Set Labels and Format Descriptions to Simplify Application Debugging

Rob Russell, The Hartford, Hartford, CT

ABSTRACT

Hansel and Gretel left breadcrumbs as they wandered through the forest to leave themselves a trail back home. This paper will show you simple ways to leave pointers for yourself in SAS® data sets and SAS formats that will help you "get back home" when you are trying to trace a problem back through various programs. The advantage of these pointers vs. breadcrumbs is that these pointers won't be eaten by birds.

INTRODUCTION

You have a production problem to debug. Where do you start your debugging efforts? In simple applications where all of the processing is contained in a single program, deciding where to start may not be that difficult. In larger, more complex and modularized applications, tracing back to the source(s) of a problem can be significantly more challenging. Fortunately, there are some simple ways that you can help yourself by setting up some pointers in your application that will help you trace backwards in your application.

An analogy for the techniques presented in this paper is what Hansel and Gretel did in the Grimm Brothers' story. When Hansel and Gretel went into the woods, they left a trail of bread crumbs behind so they could find their way back home (ignore that fact that the birds ate the crumbs). We will leave a trail – or more accurately, pointers - from our data sets and formats back to their creating programs.

The techniques that we'll look at are:

- Use data set labels to point back to the programs that created them;
- Use catalog descriptions on SAS® formats to point back to the programs that created them; and,
- Include a "production date" in the SAS data set label as a simple means of dating the file.

Note – for purposes of this paper, the term "pointers" will be used to refer to both data set labels and SAS format catalog descriptions.

WHY DO THIS?

Why go to the effort of creating these pointers? The answer is, quite simply, because the information contained in the pointers makes it easier to debug problems. By creating these pointers, you are adding metadata (information about the data) to the data itself. We are all used to using some common types of metadata – file size in megabytes, number of observations in a SAS data set or whether or not the SAS data set is indexed. If you know what program created a data set or format, you have a direct path back to the most likely place to look for a problem. The ultimate cause of a problem will frequently be somewhere other than the program that created the data set or format, however knowing where to start looking for the problem definitely saves time.

Size does matter. Our application is comprised of hundreds of SAS programs, hundreds of permanent formats (used as mapping tables) and thousands of data sets. While our application isn't that large, it would be a rare programmer who could remember all of the relationships between these objects. The pointers are a debugging aid and are also an excellent tool for new programmers in the area when they need to trace data flows through various aspects of our application.

A key point about these techniques is that you must make the effort up front to create these pointers. You can't decide that you want to retroactively create the pointers when you suddenly have a production problem to debug. It doesn't take much time or code to create the pointers. The most difficult aspect of

creating the pointers is training yourself to make that creation an automatic part of the code that you write. When you type "DATA *data set name*", automatically think of the DATA statement as being DATA *data set name* (label="*my label*"). When you create a new format that will be permanently stored in your SAS format catalog, think about the *description* that you will put on the format.

CREATING DATA SET LABELS

Creating a SAS data set label is simply a matter of including the SAS data set LABEL option statement in your SAS code. The LABEL option can be added to any SAS data set that you are creating.

The syntax for the data set LABEL option is **(LABEL="data set label")**. The descriptive label can be up to 40 characters long in SAS V6, 256 characters in V8 and higher.

Following are some code examples of the LABEL option added to a few different snippets of SAS code. One comment about the SAS code examples in this paper – most of the macro code is removed for purposes of clarity. For example, the date shown in the following code would really be a macro variable so that the appropriate production date for that cycle would be automatically inserted into the label.

```
/* Example of a data set label added to a DATA step. */
DATA  RESRVSRC.XB2020Y0
      (label="SB0007PC: Full AYR 200912 Homeowners");
set RESRVSRC.XB2020C0;
run;

/* Example of a data set label added to a PROC SORT. */
PROC  SORT
      data=RESRVSRC.XB2020C0
      out=RESRVSRC.XB2020Y0
            (label="SB0007PC: Full AYR 200912 Homeowners")
      nodupkey;
      by    VAR1 VAR2;
run;

/* Example of a data set label added to a PROC SQL. */
PROC  SQL;
      create table RESRVSRC.XB2020Y0
            (label="SB0007PC: Full AYR 200912 Homeowners")
      as select * from RESRVSRC.XB2020C0;
run;
quit;
```

You can also run the DATASETS procedure to add a label to your SAS data set. Here's an example of that code.

```
/* Example of a PROC DATASETS to add a label to a data set. */
PROC  DATASETS ddname=RESRVSRC nolist;
      modify XB2020Y0 (label="SB0007PC: Full AYR 200912 Homeowners");
run;
quit;
```

ADDING DESCRIPTIONS TO SAS FORMATS

If you issue a CATalog command against a SAS format catalog or dump said catalog via the CATALOG procedure, you will see the description associated with each catalog entry. In the case of SAS formats, the FORMAT procedure creates a description for each format ("MAXLEN = …."). That description has never been particularly useful to us. Unfortunately, there is no LABEL= or DESCRIPTION= option for PROC FORMAT. PROC FORMAT does not give you any way to replace that "MAXLEN= …" description with something more useful.

PROC CATALOG, however, does give you a way to update the description on an entry in a catalog. This technique works perfectly well against formats in format catalogs. After the format is created, use the MODIFY statement to update the description on the catalog entry (format). The description is limited to 40 characters in SAS V6, 256 characters in V8 and higher.

Following is an example of the PROC CATALOG code to update the description on a format.

```
/* Run PROC CATALOG to add a description to a format. */
PROC  CATALOG catalog=RESRVTBL.FORMATS;
      modify CSYMNAM.FORMATC
            description="SB8002FV-Table K: Claim Symbol Descrip");
run;
quit;
```

Note that the MODIFY statement needs to identify both the format name and type. The type will be either “FORMAT” or “FORMATC”, respectively, for numeric or character formats.

How do we use this technique? We simply embed a PROC CATALOG step after PROC FORMAT in the programs that create our permanent formats. An example of that code follows.

```
%put ************************************************************;
%put SB8002FV-3.0  Create Claim Symbol Description Format CSYMNAM....;
%put ************************************************************;
/* Read table to form CNTLIN data set. */
DATA SB8002D1 (keep=FMTNAME START LABEL);
    length LABEL $30;
    array  ROLLUP(*) $2 RSUF1-RSUF4;
    set &tblelib..TABLEK;
    START=CLMSYMBL;
    FMTNAME="$CSYMNAM";
    LABEL=CSYMNAME;
    output;
    return;
run;
%put ************************************************************;
/* Sort CNTLIN data set into FORMAT sort order. */
PROC SORT data=SB8002D1;
    by FMTNAME LABEL;
run;

%put ************************************************************;
/* Build FORMAT. */
PROC FORMAT cntlin=SB8002D1 library=&fmtlib.;
RUN;
%put ************************************************************;
/* Add Description to FORMAT. */
PROC CATALOG catalog=&fmtlib..FORMATS;
   modify CSYMNAM.FORMATC
          (description="SB8002FV-Table K: Claim Symbol Descrip");
run;
quit;
```

POINTERS – WHAT INFORMATION TO INCLUDE?

We now know how to update the SAS data set label and format descriptions. What do we want to put on those pointers for information that will be useful to us? There are at least 2 pieces of information that we want to put in the data set and format pointers. These are:

- the name of the program that created the object, and
- some sort of useful description about what the object is.

For SAS data sets, we also include the "production date", so we can easily see when the data set was created. Our production cycles are monthly, so we use a format of YYYYMM or YYYY-MM for the dates we put in our data set labels. One significant advantage to putting a production date into your SAS data set label is that moving or copying the data set around with operating system-based commands doesn't modify the label. This gives you a way of determining the age of a data set without being dependent on the operating system maintained create or last update date.

We use a common format for our pointers. That format is:

Positions	Information
1 - 8	Name of the program that created the data set
9 – 40	Data set description, including production date

Data set label example: **SB0002PC: Countrywide datamart Leg 0 2009-12.**

Creating program name
Some sort of meaningful description
Include the date of the production cycle that created the file

The production date isn't always the last part of the label on our SAS data sets. Sometimes the date is embedded in the middle of the label. This would occur when we have a common "stem" description across a group of SAS data sets with something that differentiates the files at the end of each label.

For descriptions on formats, we do not include the production date. SAS maintains an "Updated" date in the catalog directory. We don't copy formats around much and would rather use the space in the description for more detailed descriptions of the format.

You may decide that there is different information that you want to include in your pointers. An example of different information would be the name of the MVS batch job that created a data set or updated a format, if that information would be helpful.

USING THE POINTERS

We have discussed the technical (SAS code) aspects of how to create these pointers and what types of information you should put in the pointers. Are we done yet? No. There are a couple of small details about the pointers that we need to discuss.

TURNING THE "DETAILS" OPTION ON TO ACCESS DATA SET LABELS

In fact, the first detail is just that – the SAS system option "DETAILS". It doesn't do any good to create the SAS data set labels if you can't see them and we must turn the DETAILS option on. The syntax to turn on the DETAILS option is as follows:

```
OPTIONS DETAILS;
```

It's easier to turn the option on with the OPTIONS statement than it is to navigate down through the System Options. If you want to use the navigation method, the DETAILS option is under the System Options, Log and procedure output control, SAS log and procedure output as shown in Figure 1.

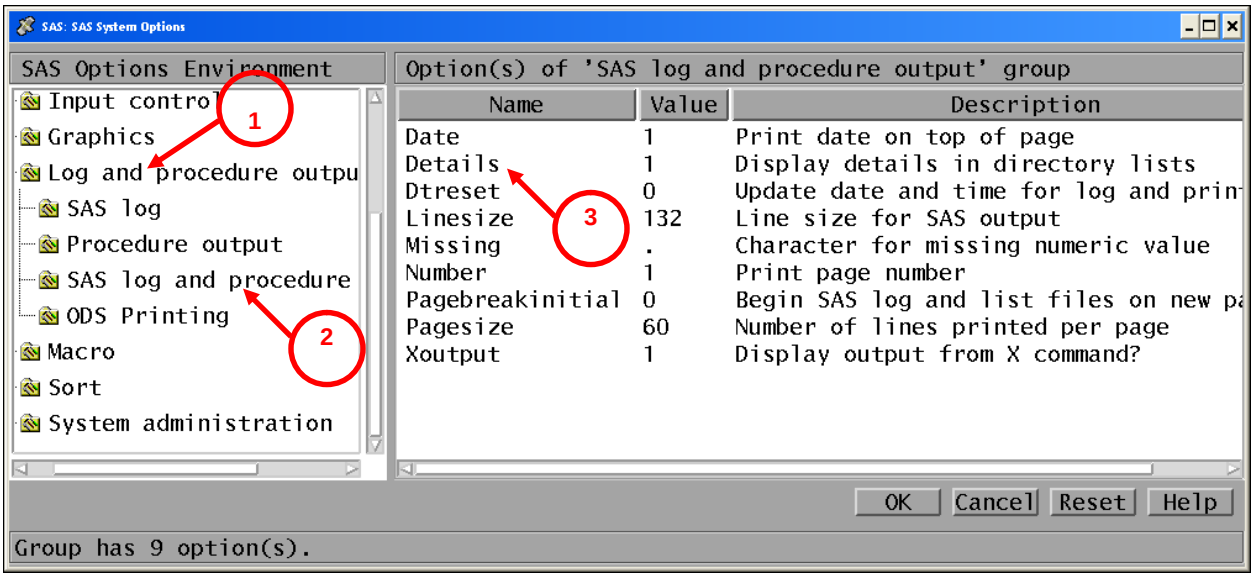

Figure 1.
The DETAILS option using the SAS System Options menus.

The syntax to turn off the DETAILS options is as follows:

```
OPTIONS NODETAILS;
```

You can always run a PROC DATASETS or PROC CONTENTS to display the data set labels. Figure 2 shows some output from a PROC DATASETS run against one of our production libraries with the system option DETAILS turned on. Note that we have data sets with a production date of 2006-09. It might be time to question whether or not those are still being used, although they could still be valid as a "point in time" snapshot of some data. Those files were likely copied between directories sometime during the past 3 years. We can still tell when those files were created - 3 years later - because the production date is in the data set label.

```
              RESERVING AUTOMATION SUPPORT                  09:45 Friday, January 15, 2010   1

                                DATASETS PROCEDURE

                               -----Directory-----

                          Libref:         RESRVSRC
                          Engine:         V9
                          Physical Name: $1$DGA124:[RESERVE.SBUPD06]

                             Obs or
     #  Name      Memtype   Entries  Vars  Label                                          Indexes
  ------------------------------------------------------------------------------------------------
     1  AAHSACRM  DATA       245001    22  SB0023PC-CMO: A.A.H.S.A. 200609
     2  AAHSACRQ  DATA        97394    21  SB0023PC-CQT A.A.H.S.A. 200609
     3  AAHSACRY  DATA        30233    21  SB0023PC-CYR A.A.H.S.A. 200609
     4  AAHSADIA  DATA        22914    43  KG0038PC-DIAG 200609 AAHSA Loss Data
     5  AAHSAORM  DATA       633954    14  SB0024PC-OSM: A.A.H.S.A.200609
     6  AAHSAORY  DATA        53466    14  SB0024PC-OSQ A.A.H.S.A. 200609
     7  AAHSARCX  DATA        76607    41  KG0030PC: 200609 Triangle fr AAHSARSV
     8  AAHSARMX  DATA      2290465    30  KG0031PC-AMO Triangle fr AAHSARS 200609
```

Figure 2.
Sample Output from PROC DATASETS

The above technique is, however, not the only way to get at the data set label and it's frankly not the most convenient. We like to be able to see the data set label as part of DIRectory commands. By turning on the SAS option DETAILS, the data set labels are included in the output from DIRectory commands.

Figure 3 shows an example of the display from a DIRectory command. This particular example is from SAS V9 running on OpenVMS. In this particular example, the column width needs to be resized to show all of the descriptive label, but that wouldn't fit in the screen shot.

The information shown in Figure 3 is also displayed in the right pane in the SAS EXPLORER if you prefer to use a mouse to navigate.

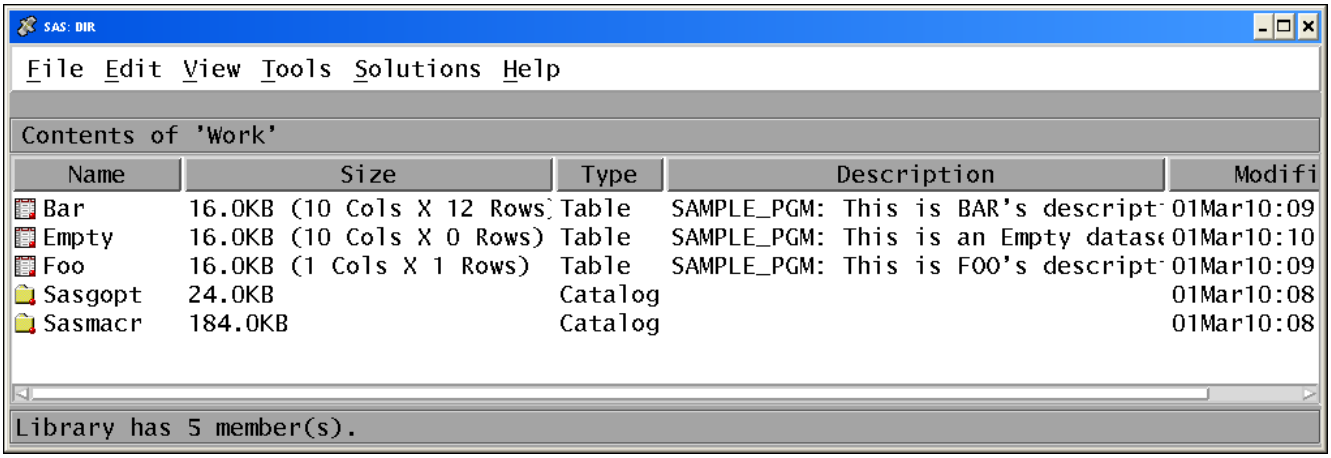

Figure 3.
Sample of DIRECTORY command showing data set labels

A LITTLE BONUS

One other useful piece of information that is displayed when you turn on the DETAILS option is data set size. You get both the number of columns (variables) and rows (observations) in the data set. In Figure 3, note that data set EMPTY has 0 Rows – this can be very useful as a quick way to identify empty data sets.

DRAWBACK TO THE DETAILS OPTION

Turning the DETAILS option on does have a cost, otherwise you'd expect that SAS would have just made that the way SAS always behaves. To get the information from the data set label, SAS needs to open the dictionary block on every data set in a library. That takes time. If you have lots of data sets in your directories, you may see a noticeable delay in the response to a DIRectory command before the directory display opens up. We have 1000+ data sets in our main production library and a DIRectory command issued against that LIBNAME takes more than a minute to come back. We consider that a small price to pay to have the data set labels displayed.

ACCESSING CATALOG DESCRIPTIONS

If you want to see the descriptions on your formats, you can run a PROC CATALOG. Following is sample code to dump the contents of your format catalog.

```
PROC CATALOG catalog=LIBRARY.FORMATS;
contents;
run; quit;
```

Figure 4 shows a portion of the output from such a PROC CATALOG run.

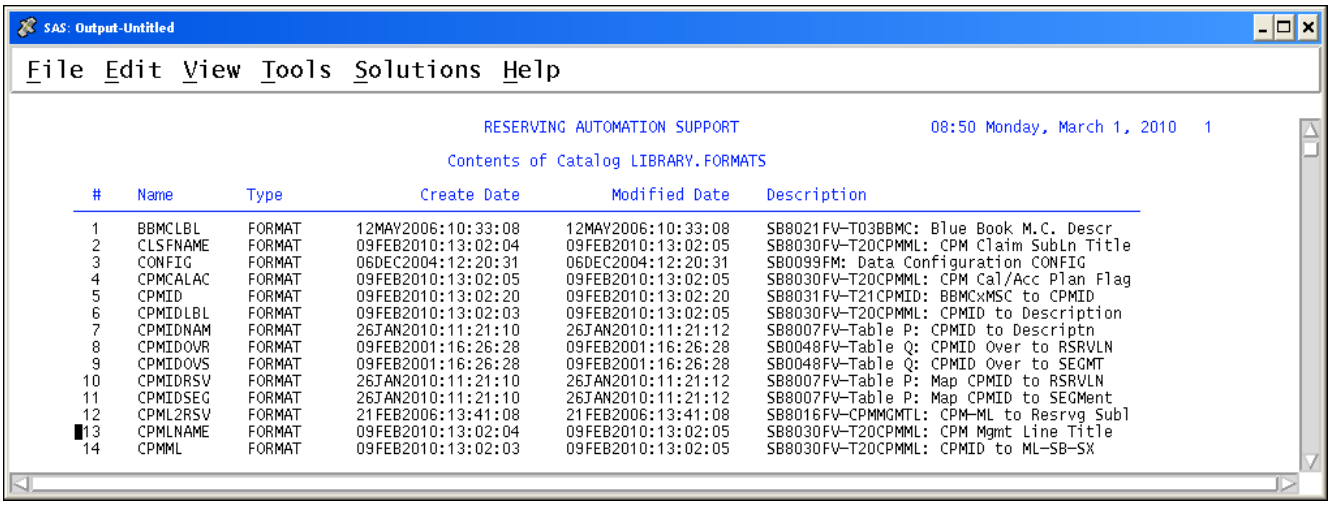

Figure 4.
Sample output from PROC CATALOG

While running PROC CATALOG certainly produces an accurate report of what is on your format catalog, it is far easier to simply use the CATalog command. The syntax of the CATALOG command is:

CAT[ALOG] ***libname.catalogname***.

Issuing the command *CATALOG LIBRARY.FORMATS* produces a display as shown in Figure 5.

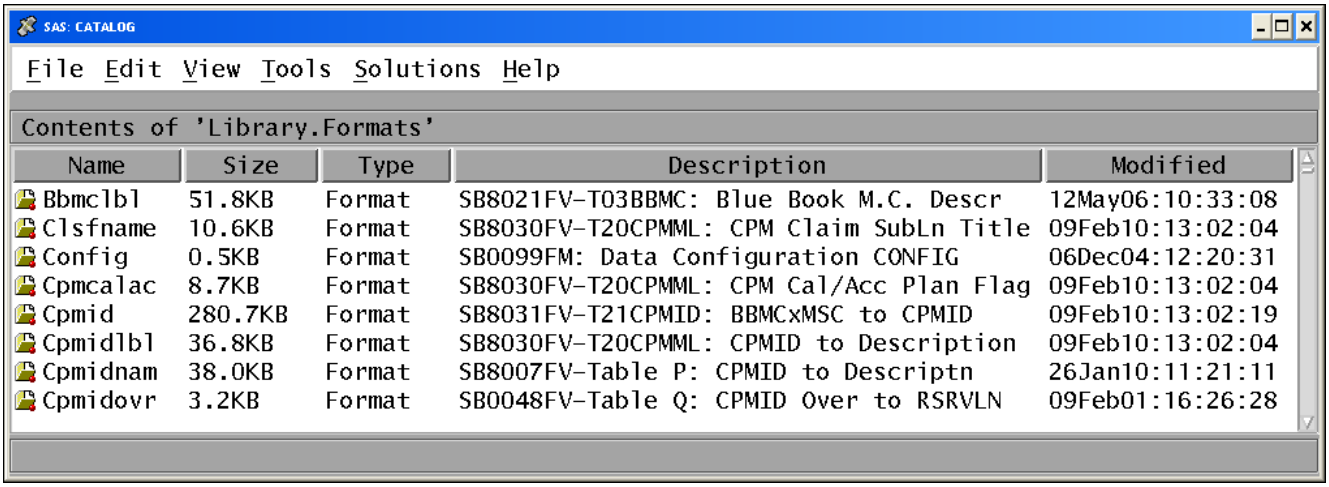

Figure 5.
Sample output from the CATALOG command

As you can see, the information provided from the CATalog command is almost the same as what is provided from the PROC CATALOG output, but it's far easier just to issue the command. Note also that displaying the catalog descriptions is not affected by the system option DETAILS – that option only affects display of the SAS data set labels.

The information shown in Figure 5 is also shown in the right pane of the SAS EXPLORER if you prefer to use a mouse to navigate.

PERSONAL PREFERENCES

My preferred technique for accessing these pointers is via commands. Why? I find it easier to type in a command and have the information popup in a new window than it is to use the mouse (this obviously applies to those environments where the mouse works). Part of my prejudice may be that I learned SAS in V6, running under Powerterm which is a character-based terminal interface.

As previously noted, in those SAS environments that do support a mouse, you can access the same information as with the command line interface. What you get back may be a little bit different. For example, as long as you have the system option DETAILS turned on, if you browse into a directory using the EXPLORER window – you'll be able to see the data set labels. I personally don't like having the directory display scrunched up in that little EXPLORER pane, don't want to size up and size back down just to see the labels. To me, it's easier to have the directory display created in a new window that I can park or close as I wish.

However you feel most comfortable accessing the pointers – the key idea is that the pointers are there and are useful. They point you from the data set or format back to the program that created them.

CONCLUSION

Debugging problems can be exasperating. Knowing where to start looking for problems, in terms of which program to start with, can be a big help in shortening your debugging time. The techniques presented in this paper are really very basic. What they do is point you back to the program that created a particular data set or format and, for data sets, give you a creation date that doesn't get blitzed when you move or copy the data set.

There are 2 comments I'd like to leave you with about the techniques:

- Adding the pointers doesn't take much work, and
- It's just a habit that you have to get into.

What's the bottom line? Start adding pointers to your data sets and formats and you'll have a trail to follow back to your programs as a start for your debugging efforts. Add the production date to your data set labels and have a permanent method for identifying when the data set was created.

REFERENCES

Shoemaker, Jack. 1998. "Creating a Self-Documenting FORMAT Catalog." *Proceedings of the 11th Annual NorthEast SAS Users Group Conference*, Pittsburgh, PA, 345-346.

ACKOWLEDGEMENTS

Ralph Leighton - who firmly believed in having data "talk to you" with descriptive labels and who got me to believe in the value of adding good labels to SAS data sets (and variables, but that's a different paper).

Jack Shoemaker - whose paper on formats at my first NESUG (in 1998) provided the generic technique for adding descriptions to format catalog entries.

DISCLAIMER

The contents of this paper express the work and opinions of the author and do not necessarily represent recommendations or practices of The Hartford. The code is supplied to illustrate possible solutions with SAS and no warranty is implied that its use will fit any particular situation.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. You may contact the author at:

Rob Russell
The Hartford
Corporate Actuarial, HO-GL-140
Hartford Plaza
Hartford, CT 06115
Work Phone: (860) 547-4777
Work Email: Rob.Russell@thehartford.com

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 registered trademarks or trademarks of their respective companies.