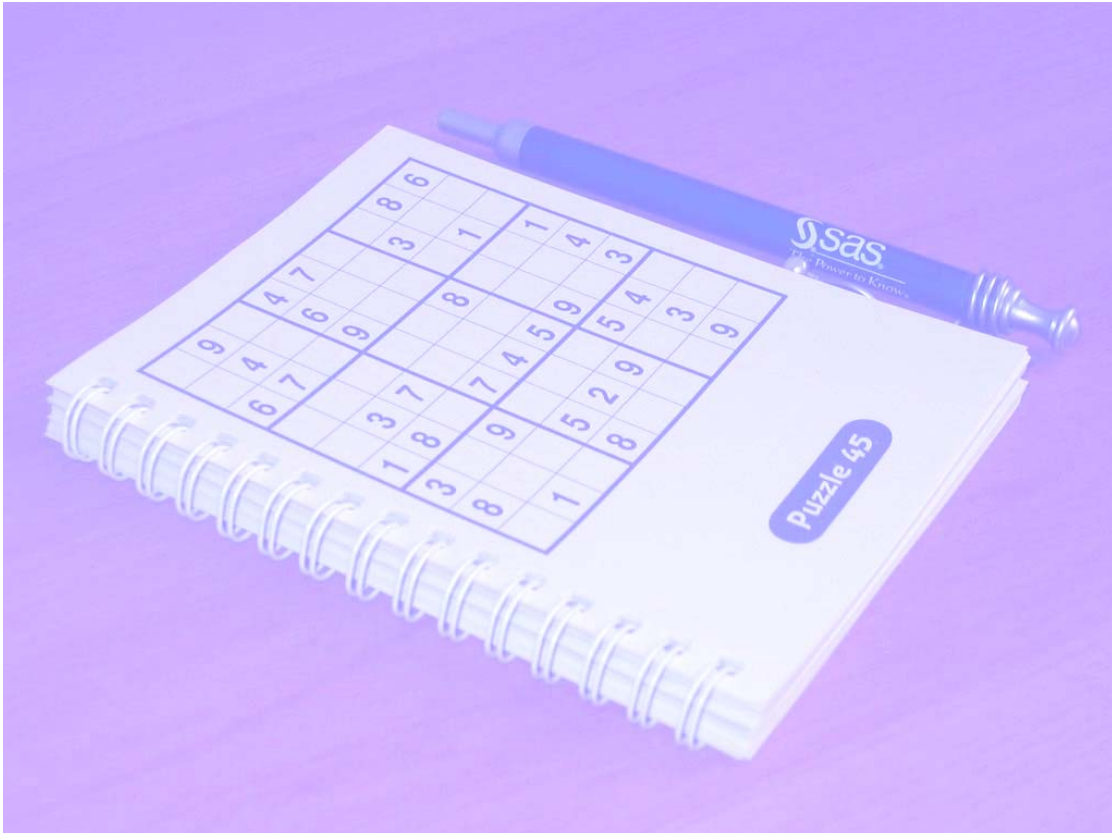


SAS, the System and Sudoku



Using SAS Software to access the Operating System, and solve other problems ... like Sudoku!

Michael Matthews
Insurance Australia Group
SNUG (SAS NSW Users Group) Q1 2006

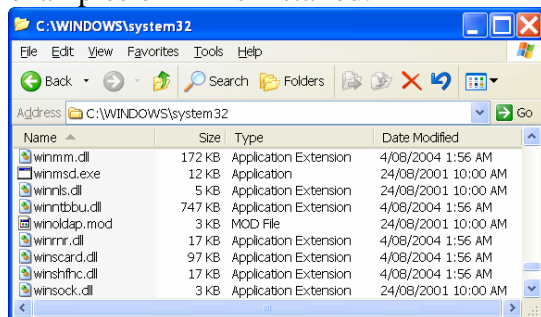
Introduction

This paper will introduce you to the concept of accessing external applications from with BASE SAS using the MODULE set of functions. It presents the topics at a high level only, so additional reading will be required to make use of these technologies.

SAS and External Executables

The techniques described in this paper are applicable to Microsoft Windows, UNIX and OpenVMS, however Microsoft Windows was the only platform available to the author for testing.

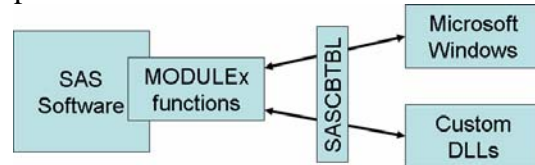
SAS Software can be used to access external executable programs (DLLs in Microsoft Windows terminology). Microsoft Windows uses DLLs extensively as for storing common routines that can be used by any application, including Windows itself. The operating system has many examples of DLLs installed.



The BASE SAS Module functions (MODULE / MODULEC / MODULEN) access any type of DLL, including custom built DLLs.

Before you can access these DLLs, you need to create a SASCBTBL file to describe to SAS the function call and parameters.

The MODULE functions can be called from the DATA STEP, SAS Macro (via %sysfunc) and SAS/IML. There are limitations via the Macro interface, as returned values cannot be updated in place.



Microsoft Windows Example 1

A common question for users is how to get a list of files in a directory imported into SAS. It is possible to use the SCL functions dopen(), dread() etc. or read a PIPE of a “DIR” command, however a solution also exists using MODULE functions direct to the Windows operating system.

```
%ut_dirlist('C:\Windows\System32\*.*', work.dirlist);
```

This macro returns more than the filename; it also returns details on the size, readonly status, hidden status, creation and modification date, and more.

FILE	DT_MOD	DT_CREATE	DT_ACCESS	FILE_SIZE	FILE_READONLY	FILE_HIDDEN
localac.dll	04AUG2004 00:56:44	04AUG2004 00:56:44	20AUG2006 11:16:29	221936	N	N
localapi.dll	04AUG2004 00:56:44	04AUG2004 00:56:44	20AUG2006 12:42:18	341904	N	N
localclb.dll	04AUG2004 00:56:44	04AUG2004 00:56:44	20AUG2006 10:40:21	11776	N	N
localcse.dll	04AUG2004 00:56:52	04AUG2004 00:56:52	20AUG2006 11:16:29	27064	N	N
localcse.exe	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 11:16:26	5120	N	N
logagent.exe	11AUG2004 01:45:04	04AUG2004 00:56:52	20AUG2006 11:16:29	96768	N	N
logonui.dll	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 10:40:21	50176	N	N
logonui.cnt	17JUL2004 11:42:38	17JUL2004 11:42:38	20AUG2006 11:16:26	488	N	N
logonui.exe	04AUG2004 00:56:52	04AUG2004 00:56:52	20AUG2006 11:16:29	59392	N	N
logoff.exe	23AUG2001 22:00:00	25AUG2005 09:14:52	20AUG2006 10:40:56	15360	N	N
logonui.exe	04AUG2004 00:56:58	04AUG2004 00:56:58	20AUG2006 13:53:43	226672	N	N
logonui.exe	04AUG2004 00:56:52	04AUG2004 00:56:52	20AUG2006 11:16:29	514560	N	N
logonui.exe.manifest	25AUG2005 09:17:49	25AUG2005 09:17:49	20AUG2006 11:16:29	488	Y	Y
lp.dll	04AUG2004 00:56:44	04AUG2004 00:56:44	20AUG2006 13:49:24	22016	N	N
lp.exe	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 11:16:29	8144	N	N
lp.exe	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 11:16:29	8192	N	N
lpshelp.dll	04AUG2004 00:56:44	04AUG2004 00:56:44	20AUG2006 10:40:21	15040	N	N
lpshelp.dll	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 10:40:21	3016	N	N
lsasrv.dll	2002CT 2004 11:21:01	04AUG2004 00:56:44	20AUG2006 12:42:18	721304	N	N
lsassvc.exe	04AUG2004 00:56:52	04AUG2004 00:56:52	20AUG2006 12:42:18	13312	N	N
LUser.dll	20JUL2005 11:40:39	20JUL2005 11:40:34	20AUG2006 11:16:29	75	N	N
lsimgui.exe	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 11:16:29	42168	N	N
ls32.dll	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 12:42:18	2560	N	N
lsapi.dll	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 10:40:21	8096	N	N
Lexcept.rls	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 11:16:29	168	N	N
Lurl.rls	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 11:16:29	7046	N	N
lsapi.exe	03JUL2005 16:50:19	25AUG2005 09:16:43	20AUG2006 13:42:37	0	N	N
lsapi.exe	04AUG2004 00:56:52	04AUG2004 00:56:52	20AUG2006 11:16:30	72704	N	N
lsapi.hook.dll	23AUG2001 22:00:00	23AUG2001 22:00:00	20AUG2006 10:40:21	8192	N	N

The following Windows functions are called within the macro:

- FindFirstFileA() - Starts a search for files and returns the first one (including wildcards).

- FindNextFileA() - Loop calling this to find subsequent files.
- FindClose() - When finished.

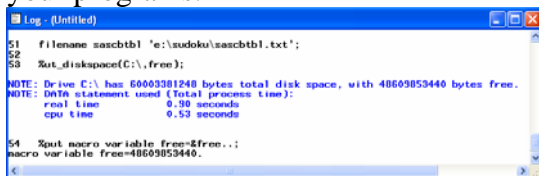
The source code for this example is included in the appendices at the end.

Microsoft Windows Example 2

Another common question is how much space is left on a particular disk? This can be useful before copying files, and also could be used to monitor the amount of WORK space remaining before starting a large job.

```
%ut_diskspace(C:\,free);
%ut_diskspace(%sysfunc(pathname(WORK)),
              freevar,totalvar);
```

The macro returns the amount of free space to the LOG, and also updates the macro variables for later analysis by your programs.



The following Windows function is called within the macro:

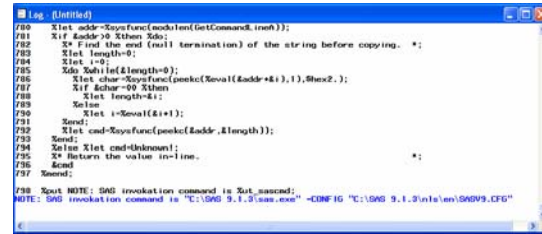
- GetDiskFreeSpaceA() - Returns details on free and total space for the given directory. Function works on both local and network disks.

Microsoft Windows Example 3

The SAS Command line is mainly available to you via examination of SAS options, however it can be useful to determine the exact string used to invoke SAS.

```
%put NOTE: SAS invokation command is
%ut_sascmd;
```

This function returns a string that contains the command line used to invoke SAS.



The following Windows function is called within the macro:

- GetCommandLineA() - Returns a pointer to the string used to invoke SAS.

Microsoft Windows Possibilities

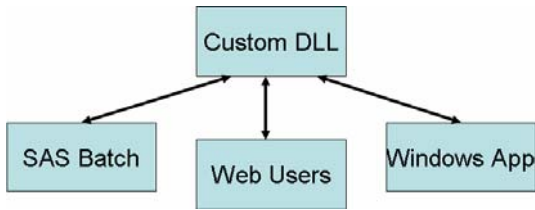
Microsoft's MSDN website includes all of the functions that you could use. One additional example is the MessageBoxA function, which provides all of the message boxes that you normally see in Windows. This function could be used to improve your SAS/AF applications, for example.

Why write your own DLL?

In the previous sections, Windows DLLs have been used. It is also possible for you to write your own DLLs, as some functions are easier to write in languages other than SAS.

DLLs are reusable between other applications. Other technologies for sharing application functionality have different pros and cons, and should be examined before developing a DLL, although the DLL overheads are low, so are generally fast.

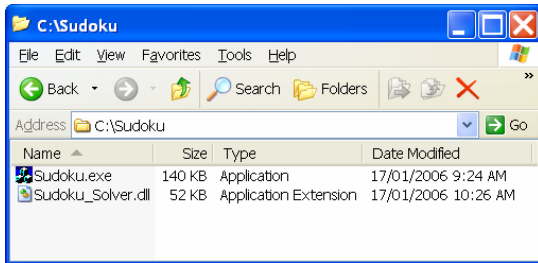
The following diagram shows a single DLL being called by SAS (using the MODULE functions), by Web applications, and also by a windows application.



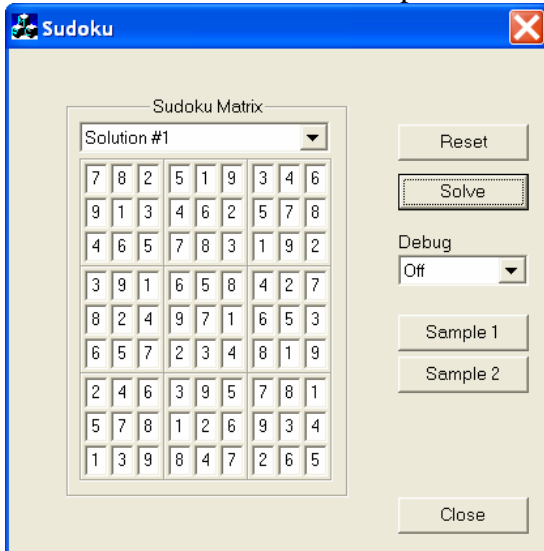
SAS and Sudoku

Sudoku is challenge enjoyed by many people around the world, and as such a good generic application to show how the technologies can be used.

In the example below, Sudoku.exe is a windows application that contains a display, but no intelligence on how to solve Sudoku. The intelligence is all contained within the Sudoku_Solver.dll file.

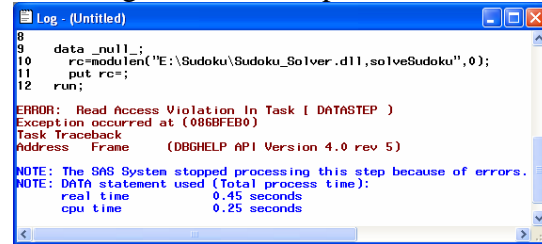


Starting the Sudoku.exe file allows you to enter values into the cells, and click the 'Solve' button to solve the puzzle.



The Sudoku_Solver.dll can then be called from SAS by creating the appropriate entries in the SASCBTBL

file. It should be noted that incorrect entries in this file can cause spectacular problems, with messages like the following a minor example!



First, you must assign the SASCBTBL fileref to the file you have created. SASCBTBL is a reserved name, similar to SASAUTOS and LIBRARY.

```
filename sascbtbl
        'E:\Sudoku\sascbtbl.txt';
```

```
data _null_;
    array sudoku(81) 8;
```

The MODULEN function is used here to initialise the table. The source code be from a SAS dataset instead.

```
rc=modulen("setSudokuValues",
           7,0,0,5,1,0,3,0,0,
           9,0,0,0,6,0,5,7,0,
           0,6,5,0,8,0,0,0,2,
           0,0,1,0,5,0,0,2,0,
           0,0,4,0,7,1,6,5,0,
           0,5,7,0,0,4,0,0,9,
           2,4,6,0,9,0,7,8,1,
           0,0,0,1,0,6,0,0,4,
           0,3,0,0,4,0,0,0,0);
put 'NOTE: Called setSudokuValues()'
    rc=;
```

The MODULEN function then asks the DLL to Solve the Sudoku grid.

```
rc=modulen("solveSudoku",2);
put 'NOTE: Called solveSudoku()' rc=;
rc=modulen("getNumSudokuResults");
put 'NOTE: getNumSudokuResults()'
    rc=;
```

Finally, the results are returned and displayed in the SAS Log.

```
rc=modulen("getSudokuResults",1,
           Sudoku( 1), ... , Sudoku(81));
put 'NOTE: Called getSudokuResults()'
    rc=;
```

```

do i=1 to 81;
  put sudoku(i) @;
  if mod(i,9)=0 then put;
end;
run;

```

```

Log - (Untitled)
642   put sudoku(i) @;
643   if mod(i,9)=0 then put;
644   end;
645   run;

NOTE: Called setSudokuValues() rc=0
NOTE: Called solveSudoku() rc=0
NOTE: Called getNumSudokuResults() rc=1
NOTE: Called getSudokuResults() rc=0
7 8 2 5 1 9 3 4 6
9 1 3 4 6 2 5 7 8
4 6 5 7 8 3 1 9 2
3 9 1 6 5 8 4 2 7
8 2 4 9 7 1 6 5 3
6 5 7 2 3 4 8 1 9
2 4 6 3 9 5 7 8 1
5 7 8 1 2 6 9 3 4
1 3 9 8 4 7 2 6 5

NOTE: DATA statement used (Total process time):
      real time           22.75 seconds
      cpu time            0.45 seconds

```

Conclusion

Accessing Windows functions directly is extremely powerful, enhancing the abilities of BASE SAS.

It is possible to develop custom code in DLLs, to then be accessed from SAS Software and other applications. These functions are very reliable once developed and debugged

References

The main source of reference is the SAS OnlineDoc 9.1.3, and

<http://support.sas.com/>. Additional resources exist on the InterNet, however I found that some of the SASCBTBLs were incorrect and caused crashes – so beware!

Microsoft's MSDN website <http://msdn.microsoft.com/> has extensive information on the Windows operating system and function definitions.

Appendix 1: SASCBTBL used in these examples

```
*****;
*
* CONFIGURATION PROGRAM:
*   SASCBTBL.TXT
*
* DESCRIPTION:
*   This configuration file allows SAS to access routines from
*   within external DLL files.
*
*****;
*
* AUTHOR : Michael Matthews
*
*****;

routine GetTempPathA
  module=KERNEL32
  minarg=2
  maxarg=2
  stackpop=called
  returns=long;
  arg 1 input num byvalue format=piB4.;
  arg 2 update char format=$cstr200.;

routine Beep
  module=KERNEL32
  minarg=2
  maxarg=2
  stackpop=called
  callseq=byvalue;
  arg 1 input num byvalue format=piB4.;
  arg 2 input num byvalue format=piB4.;

routine FindFirstFileA
  module=KERNEL32
  minarg=14
  maxarg=14
  stackpop=called
  returns=long;
  arg 1 input char format=$cstr260.;
  arg 2 update num fdstart format=piB4.;
  arg 3 update num format=piB4.;
  arg 4 update num format=piB4.;
  arg 5 update num format=piB4.;
  arg 6 update num format=piB4.;
  arg 7 update num format=piB4.;
  arg 8 update num format=piB4.;
  arg 9 update num format=piB4.;
  arg 10 update num format=piB4.;
  arg 11 update num format=piB4.;
  arg 12 update num format=piB4.;
  arg 13 update char format=$cstr260.;
  arg 14 update char format=$cstr14.;

routine FindNextFileA
  module=KERNEL32
  minarg=14
  maxarg=14
  stackpop=called
  returns=long;
  arg 1 input num byvalue format=piB4.;
  arg 2 update num fdstart format=piB4.;
  arg 3 update num format=piB4.;
  arg 4 update num format=piB4.;
  arg 5 update num format=piB4.;
  arg 6 update num format=piB4.;
  arg 7 update num format=piB4.;
  arg 8 update num format=piB4.;
  arg 9 update num format=piB4.;
  arg 10 update num format=piB4.;
  arg 11 update num format=piB4.;
  arg 12 update num format=piB4.;
  arg 13 update char format=$cstr260.;
  arg 14 update char format=$cstr14.;

routine FindClose
  module=KERNEL32
  minarg=1
  maxarg=1
  stackpop=called
  returns=long;
  arg 1 input num byvalue format=piB4.;

routine LoadLibraryA
  module=KERNEL32
  minarg=1
  maxarg=1
  stackpop=called
```

```
returns=long;
arg 1 input char byaddr format=%cstr256.;

routine FreeLibrary
module=KERNEL32
minarg=1
maxarg=1
stackpop=called
returns=long;
arg 1 input num byvalue format=ib4.;

routine GetLastError
module=KERNEL32
minarg=0
maxarg=0
stackpop=called
returns=long;

routine FormatMessageA
module=KERNEL32
minarg=7
maxarg=7
stackpop=called
returns=long;
arg 1 input num byvalue format=pi4.;
arg 2 input num byvalue format=pi4.;
arg 3 input num byvalue format=pi4.;
arg 4 input num byvalue format=pi4.;
arg 5 update char byaddr format=%cstr2000.;
arg 6 input num byvalue format=pi4.;
arg 7 input num byvalue format=pi4.;

routine MessageBoxA
module=USER32
minarg=4
maxarg=4
stackpop=called
returns=short;
arg 1 input num byvalue format=pi4.;
arg 2 input char byaddr format=%cstr2000.;
arg 3 input char byaddr format=%cstr2000.;
arg 4 input num byvalue format=pi4.;

routine GetDiskFreeSpaceA
module=KERNEL32
minarg=5
maxarg=5
stackpop=called
returns=long;
arg 1 input char byaddr format=%cstr200.;
arg 2 output num byaddr format=pi4.;
arg 3 output num byaddr format=pi4.;
arg 4 output num byaddr format=pi4.;
arg 5 output num byaddr format=pi4.;

routine GetCommandLineA
module=KERNEL32
minarg=0
maxarg=0
stackpop=called
returns=long;
```

Appendix 2: %UT_DIRLIST source

```
/******  
/*  
/* AUTOCALL MACRO: *  
/* %UT_DIRLIST *  
/* *  
/* DESCRIPTION: *  
/* Find the list of files in a given directory (using optional *  
/* wildcards). *  
/* *  
/* INPUT: *  
/* PATH - Path to search (quoted). Eg: 'C:\*.*', 'D:\FRED\*.SAS' *  
/* DSNAME - SAS Dataset name. *  
/* *  
/* OUTPUT: *  
/* SAS Dataset created. *  
/* *  
/* USAGE: *  
/* %ut_dirlist('C:\*.*',work.file_list); *  
/* *  
/******  
/* AUTHOR : Michael Matthews *  
/* *  
/******  
%macro ut_dirlist(path,dsname);  
  %local FILE_ATTRIBUTE_READONLY FILE_ATTRIBUTE_HIDDEN FILE_ATTRIBUTE_SYSTEM FILE_ATTRIBUTE_DIRECTORY  
        FILE_ATTRIBUTE_ARCHIVE FILE_ATTRIBUTE_DEVICE FILE_ATTRIBUTE_NORMAL FILE_ATTRIBUTE_TEMPORARY  
        FILE_ATTRIBUTE_SPARSE_FILE FILE_ATTRIBUTE_REPARSE_POINT FILE_ATTRIBUTE_COMPRESSED  
        FILE_ATTRIBUTE_OFFLINE FILE_ATTRIBUTE_NOT_CONTENT_IDX FILE_ATTRIBUTE_ENCRYPTED;  
  
  %* Constants from WINNT.h *;  
  %let FILE_ATTRIBUTE_READONLY =00000001x;  
  %let FILE_ATTRIBUTE_HIDDEN =00000002x;  
  %let FILE_ATTRIBUTE_SYSTEM =00000004x;  
  %let FILE_ATTRIBUTE_DIRECTORY =00000010x;  
  %let FILE_ATTRIBUTE_ARCHIVE =00000020x;  
  %let FILE_ATTRIBUTE_DEVICE =00000040x;  
  %let FILE_ATTRIBUTE_NORMAL =00000080x;  
  %let FILE_ATTRIBUTE_TEMPORARY =00000100x;  
  %let FILE_ATTRIBUTE_SPARSE_FILE =00000200x;  
  %let FILE_ATTRIBUTE_REPARSE_POINT =00000400x;  
  %let FILE_ATTRIBUTE_COMPRESSED =00000800x;  
  %let FILE_ATTRIBUTE_OFFLINE =00001000x;  
  %let FILE_ATTRIBUTE_NOT_CONTENT_IDX =00002000x;  
  %let FILE_ATTRIBUTE_ENCRYPTED =00004000x;  
  
  data &dsname(keep=NA_FILE DT_MOD DT_CREATE DT_ACCESS NM_SIZE  
        FL_READONLY FL_HIDDEN FL_SYSTEM FL_DIRECTORY FL_COMPRESSED FL_ENCRYPTED);  
    length NA_FILE $ 260 DT_MOD DT_CREATE DT_ACCESS NM_SIZE 8  
           FL_READONLY FL_HIDDEN FL_SYSTEM FL_DIRECTORY FL_COMPRESSED FL_ENCRYPTED $ 1;  
  
    format DT_MOD DT_CREATE DT_ACCESS datetime19. ;  
  
    %* Declare and initialise the arguments to the MODULE functions. *;  
    length handle 8 dwFileAttributes ftCreationTime1 ftCreationTime2 ftLastAccessTime1 ftLastAccessTime2  
           ftLastWriteTime1 ftLastWriteTime2 nFileSizeHigh nFileSizeLow dwReserved0 dwReserved1 8  
           cFileName $ 260 cAlternateFileName $ 14;  
    dwFileAttributes=.;  
    ftCreationTime1=.;  
    ftCreationTime2=.;  
    ftLastAccessTime1=.;  
    ftLastAccessTime2=.;  
    ftLastWriteTime1=.;  
    ftLastWriteTime2=.;  
    nFileSizeHigh=.;  
    nFileSizeLow=.;  
    dwReserved0=.;  
    dwReserved1=.;  
    cFileName=' '  
    cAlternateFileName=' '  
  
    %* Find the first file (and check the HANDLE returned). *;  
    handle=modulen('FindFirstFileA',&path,dwFileAttributes,ftCreationTime1,ftCreationTime2,  
           ftLastAccessTime1,ftLastAccessTime2,ftLastWriteTime1,  
           ftLastWriteTime2,nFileSizeHigh,nFileSizeLow,dwReserved0,  
           dwReserved1,cFileName,cAlternateFileName);  
  
    if handle>0 then do;  
      rc=1;  
      do while(rc=1);  
        %* Convert the results to SAS dates etc. *;  
        na_file=cFileName;  
        nm_size=nFileSizeLow+nFileSizeHigh*(2**32);  
        dt_mod=(ftLastWriteTime1+ftLastWriteTime2*(2**32))/1000000+'01jan1601:00:00'dt+'10:00't;  
        dt_create=(ftCreationTime1+ftCreationTime2*(2**32))/1000000+'01jan1601:00:00'dt+'10:00't;  
        dt_access=(ftLastAccessTime1+ftLastAccessTime2*(2**32))/1000000+'01jan1601:00:00'dt+'10:00't;  
        FL_READONLY=substr('NY',(band(dwFileAttributes,&FILE_ATTRIBUTE_READONLY)>0)+1,1);  
        FL_HIDDEN=substr('NY',(band(dwFileAttributes,&FILE_ATTRIBUTE_HIDDEN)>0)+1,1);  
        FL_SYSTEM=substr('NY',(band(dwFileAttributes,&FILE_ATTRIBUTE_SYSTEM)>0)+1,1);  
      end;  
    end;
```

```
FL_DIRECTORY=substr('NY',(band(dwFileAttributes,&FILE_ATTRIBUTE_DIRECTORY)>0)+1,1);
FL_COMPRESSED=substr('NY',(band(dwFileAttributes,&FILE_ATTRIBUTE_COMPRESSED)>0)+1,1);
FL_ENCRYPTED=substr('NY',(band(dwFileAttributes,&FILE_ATTRIBUTE_ENCRYPTED)>0)+1,1);
output;
%* Loop for subsequent files.
rc=modulen('FindNextFileA',handle,dwFileAttributes,ftCreationTime1,ftCreationTime2,
           ftLastAccessTime1,ftLastAccessTime2,ftLastWriteTime1,
           ftLastWriteTime2,nFileSizeHigh,nFileSizeLow,dwReserved0,
           dwReserved1,cFileName,cAlternateFileName);

end;
%* Close the HANDLE neatly.
rc=modulen('FindClose',handle);
end;
run;
%mend;
```

Appendix 3: %UT_DISKSPACE source

```
/******  
/*  
/* AUTOCALL MACRO: *  
/* %UT_DISKSPACE *  
/* *  
/* DESCRIPTION: *  
/* Returns the amount of free space on a given drive. *  
/* *  
/* INPUT: *  
/* DRIVE : Drive to check for space (unquoted with \ - eg: C:\) *  
/* FREEVAR : (optional) Macro variable to be created to contain *  
/* amount of free space in bytes *  
/* TOTALVAR : (optional) Macro variable to be created to contain *  
/* the total amount of space in bytes on the disk *  
/* *  
/* OUTPUT: *  
/* Macro variables created and updated, and a note returned to the *  
/* SAS Log. *  
/* *  
/* USAGE: *  
/* %ut_diskspace(K:\,free,total); *  
/* *  
/* %ut_diskspace(%sysfunc(pathname(WORK))\,free); *  
/* %put free=&free; *  
/* *  
/******  
/*  
/* AUTHOR : Michael Matthews *  
/* *  
/******  
%macro ut_diskspace(drive,freevar,totalvar);  
  data _null_;  
    /* Call the Windows function to get the disk space values. *  
    call module('GetDiskFreeSpaceA',"&drive",sectors,bytes,free,total); *  
    /* Change to a number of bytes. *  
    free=free*sectors*bytes;  
    total=total*sectors*bytes;  
    put "NOTE: Drive &drive has " total "bytes total disk space, with " free "bytes free."; *  
    /* Create macro variables for output, as required. *  
    %if &freevar ne %then %do;  
      %if %symexist(&freevar)=0 %then %global &freevar;  
      call symputx("&freevar",free);  
    %end;  
    %if &totalvar ne %then %do;  
      %if %symexist(&totalvar)=0 %then %global &totalvar;  
      call symputx("&totalvar",total);  
    %end;  
  run;  
%mend;
```

Appendix 4: %UT_SASCMD source

```
/******  
/*  
/* AUTOCALL MACRO: *  
/* %UT_SASCMD *  
/* *  
/* DESCRIPTION: *  
/* Returns the SAS Command line used to open the current SAS *  
/* session. *  
/* *  
/* INPUT: *  
/* None. *  
/* *  
/* OUTPUT: *  
/* Inline constant returned. Be warned that multiple quotes can be *  
/* present in the results making parsing tricky. *  
/* *  
/* USAGE: *  
/* %put NOTE: SAS invokation command is %ut_sascmd; *  
/* *  
/******  
/*  
/* AUTHOR : Michael Matthews *  
/* *  
/******  
%macro ut_sascmd;  
  %local addr cmd char i length;  
  
  /* Find the address of the Command Line using the windows function. */  
  %let addr=%sysfunc(modulen(GetCommandLineA));  
  %if &addr>0 %then %do;  
    /* Find the end (null termination) of the string before copying. */  
    %let length=0;  
    %let i=0;  
    %do %while(&length=0);  
      %let char=%sysfunc(peekc(%eval(&addr+&i),1),%hex2.);  
      %if &char=00 %then  
        %let length=&i;  
      %else  
        %let i=%eval(&i+1);  
      %end;  
    %let cmd=%sysfunc(peekc(&addr,&length));  
  %end;  
  %else %let cmd=Unknown!;  
  /* Return the value in-line. *  
  &cmd *  
%mend;
```