

```
*****;
* PURPOSE: Controlling the Execution of SAS Code *;
* Without Using Macros or External %INCLUDES *;
* LIBRARY: TSO.DAOSPE.WISUGLIB(MULTSTEP) *;
* UPDATED: 11/14/07 by David Oesper *;
*****;
```

```
options nocenter source2 linesize=73;
```

```
%macro savestep(stepn);
  filename &stepn temp;
  data _null_;
    file &stepn;
    input sasline $72.;
    put sasline;
  %mend savestep;
```

```
%savestep(step010); datalines4;
proc sql;
  connect to db2...;
  create table outsas.orders as
  select *
  from connection to db2
  (select...
  from...
  where...
  for fetch only with ur);
  %put sqlxrc=&sqlxrc sqlxmsg=&sqlxmsg;
  disconnect from db2;
quit;
```

```
proc print data=outsas.orders (obs=23);
  title 'Orders';
  ;;;
```

```
%savestep(step020); datalines4;
proc sql;
  connect to db2...;
  create table outsas.cust as
  select *
  from connection to db2
  (select...
  from...
  where...
  for fetch only with ur);
  %put sqlxrc=&sqlxrc sqlxmsg=&sqlxmsg;
  disconnect from db2;
quit;
```

```
proc print data=outsas.cust (obs=23);
  title 'Customers';
  ;;;
```

```
%savestep(step030); datalines4;
proc sort data=outsas.orders;
  by hh_num;
```

```
proc sort data=outsas.cust;
  by hh_num;
```

```
data orders2;
  merge outsas.orders (in=a) outsas.cust;
  by hh_num;
```

```
proc freq data=orders2;
  tables ord_cat*cust_seg;
  title 'Order Category - Customer Segment';
  ;;;
```

```
%include step010; %sysexec echo 'multstep step010 completed';
%include step020; %sysexec echo 'multstep step020 completed';
%include step030; %sysexec echo 'multstep step030 completed';
```

In porting SAS mainframe code to Unix, one of the challenges was how to replicate multiple JCL job steps in the Unix environment. In both environments, it is advantageous to isolate long-running processes (such as some pass-through SQL data pulls) in separate steps, turning them on or off as needed. While not equivalent to mainframe jobs steps with its separate invocations of SAS in each step, this process allows one to control the execution of blocks of code within a single SAS invocation, in both the Unix and PC SAS environment.

```
//STEP010 EXEC SASCURR
//OUTSAS DD DSN=DXJT.G30.DAOESPE.ERMDLIB.EOPEN.STEP010,
//      DISP=(NEW,CATLG,KEEP),
//      SPACE=(CYL,(3000,200),RLSE)
//SASCURR.SYSIN DD *

OPTIONS MSTORED SASMSTORE=OUTSAS;

%MACRO PARMS / STORE;
%GLOBAL BEG_SEND_DT
        BEG_OPEN_DT
        END_OPEN_DT
        FSCL_YR
        ;
%LET BEG_SEND_DT = '2007-06-30';
%LET BEG_OPEN_DT = '2007-07-07';
%LET END_OPEN_DT = '2007-08-03';
%LET FSCL_YR     = 2007;
%MEND PARMS;

%PARMS;

PROC SQL;
  CONNECT TO DB2...);
  CREATE TABLE OUTSAS.OPENS AS
    SELECT *
    FROM CONNECTION TO DB2
    (SELECT...
    FROM...
    WHERE...OPEN_DT BETWEEN &BEG_OPEN_DT AND &END_OPEN_DT
    FOR FETCH ONLY WITH UR);
  %PUT SQLXRC=&SQLXRC SQLXMSG=&SQLXMSG;
  DISCONNECT FROM DB2;
QUIT;

/*
//STEP020 EXEC SASCURR
//OUTSAS DD DSN=DXJT.G30.DAOESPE.ERMDLIB.EOPEN.STEP020,
//      DISP=(NEW,CATLG,KEEP),
//      SPACE=(CYL,(3000,200),RLSE)
//SASCURR.SYSIN DD *

LIBNAME EOPENS 'DXJT.G30.DAOESPE.ERMDLIB.EOPEN.STEP010' DISP=SHR;

OPTIONS MSTORED SASMSTORE=EOPENS;

%PARMS;

PROC SQL;
  CONNECT TO DB2...);
  CREATE TABLE OUTSAS.SENT AS
    SELECT *
    FROM CONNECTION TO DB2
    (SELECT...
    FROM...
    WHERE...SEND_DT BETWEEN &BEG_SEND_DT AND &END_OPEN_DT
    FOR FETCH ONLY WITH UR);
  %PUT SQLXRC=&SQLXRC SQLXMSG=&SQLXMSG;
  DISCONNECT FROM DB2;
QUIT;
/*
```