Automated Test Generator For Software Testing

K.L Kanthan

Chemical Bank

Abstract

A dynamic user-friendly IVP (Installation Verification program) system was designed to assure quality testing of online and batch software products, using several features of SAS . This menu system, driven by the user input, runs TPNS script simulations, generates JCL and Jobstreams for the objective of automated software testing. It facilitates browsing the results of the requested tests. Base SAS, SAS/AF, and the Display Manager software (DMS) made the development process easier, thus making the end product a flexible front-end interface to testing of software products.

1. INTRODUCTION :

A menu driven IVP facility was developed as a user-friendly front end to test installed vendor software products. This was achieved through the base SAS, SAS macro language, SAS/FSP, Display Manager(DMS), and SAS/AF. The mechanics and details of the generation of JCL, Jobstreams, running of these Batch or on-line tests are transparent to the user. The user is guided with extensive help facilities where and when needed and this makes the system self documenting.

Menu driven systems conceptually are modular, flexible, and easy to use. In principle, few responses from the user should be enough to automatically generate the test streams. Also, the system should be flexible enough to allow the more sophisticated user enough freedom, and not restrict wherever possible.

All that is needed from the user are responses to queries such as: what product is to be tested, whether to submit the job or not; if the output is preferred online or hardcopy; if SPF environment is needed. Based on these information, test jobstreams are built, jobs are submitted, user is branched to SPF environment and returned back to the menu where he came from.

This automated IVP System performs the following major functions:

- TPNS simulation to test the on-line products such as RMDS, FILE-AID, PANVALET etc.,
- Testing of software products through batch jobs such as COBOL, ASSEMBLER etc.,
- Online viewing of the resulting output from the test
- Allowing the user to branch to SPF and then returning to testing
- Extensive help facilities

The software tester is presented with a main menu, from which the needed options can be selected to perform the desired functions.

2. IVP SYSTEM STRUCTURE :

2.1 Menu system :

The IVP Menu system is a set of interconnected menus of several types of screens (CBT, HELP, MENU, PROGRAM etc.), that provide easy to use interface for the IVP test application. Some menus are shown in Appendix A . Foundation for this is laid through SAS/AF software. Very minimal or no knowledge of the SAS syntax and structure is expected of the user. User's responses are translated into several SAS macro variable values, names, or passed as parameters to some of the SAS and JCL procedures etc., Branching to different logical and functional regions also takes place based on some of the responses. Where some of the inputs are mandatory (such as which product is to be tested), but many other responses are optional (there are defaults, but it is overridden if inputs are entered). For most of these fields help facility was built in, and on any of these optional fields, if PF1 key is used, the following will be dispalyed: description of the field, whether it is mandatoy to enter input, whether it is optional, which formats to use, what are valid values etc.,

2.2 Automated Job generation :

From Batch IVP Menu a tester chooses option 1 to select the product to be tested, and another subpanel is displayed. Here the product name and other fields are entered as needed. Product name is mandatory and other fields are optional. Data field formats, and any other details needed at this point can be obtained by general or field specific help (ie. by using PF1) facility. At the end of this process (use PF3 to end this screen), user will returen back to Batch IVP menu. Now you may want to browse the generated JCL, or submit the Job, or look at the output of the test run. To look at the test output you can branch to ISPF/SDSF. When you end SPF you will be back where you left off in the IVP Menu. When you end Batch IVP Menu (ie by PF3 or option 4) you will return to IVP Primary menu.

2.3 TPNS IVPs for online products :

Option 1 (Help facility for TPNS test) describes the formats of the commands necessary, and other information necessary. In all only three commands are needed for starting and ending TPNS simulation, and to run TPNS utilities. Option 2 runs the TPNS simulation. When the test is in progress, messages are received at the terminal regarding the

status of the test every time enter key is hit. A message also is sent when the test is completed. Another option with the help of a TPNS utility makes the output available for hardcopy or online.

3. METHODOLOGY :

This section describes the underlying logic necessary for generating the Jobstreams, TPNS testing, and other test environment. To use the menu system, one does not need to be familiar with the details of this section. Default values are used where the user input is not mandatory (user has the option to override some of these), and this keeps the required user responses to a minimum.

3.1 Batch IVPs :

A simple SAS code builds the JCL and Jobstream for the batch testing. Input to this program is a set of JCL and SAS statements. The SAS code is mostly made up of conditional statements with a do-end block. The do-end blocks are made of JCL statements, embedded with SAS macro variable names.

A sample code is as follows:

```
If (some condition) then do;
//&JOBNAME   JOB (&ACCT1,&ACCT2),
//      &ACTINFO,CLASS=&CLASS,
//      MSGCLASS=&MSG,NOTIFY=&NOTIFY
//***
end;

IF "&SYST"  = 'ASM' THEN DO;
//ASM     EXEC  xxxxx
//ddn1      DD  DSN=dsname.&n1&n2,
//          DISP=SHR
//          .....
//          DD  DSN=dsname.xxx(&name),
//          DISP=SHR
end;
```

On the execution of these embedded SAS code in the input, the JCL statements are added, or modified (by resolving the macro variable), through SAS PUT statements, written to dataset named in ddname JOBSUB. The SAS macro code for this (1), and file allocations (2), and how this code is called and executed (3) are given below :

File allocations under TSO:

```
X ALLOC DA('dataset.INP(INP)')
              FI(INP) SHR;
X ALLOC DA('dataset.SASCDE')
              FI(SASCDE) OLD;        (2)
X ALLOC DA('dataset.JOBSUB')
              FI(JOBSUB) OLD;
X ALLOC DA('dataset.INP')
              FI(SOURCLIB) SHR;
RUN;
```

Where datasets named in DDNAMEs INP has the SAS code and JCL statements, SACDE the intermediate SAS PUT statements, JOBSUB the final JCL generated respectively.

```
options mprint macrogen ;
%inc sourclib(macs);
%inc sourclib(jobgen);
data _null_;  %jobgen;
data _null_;                         (3)
file jobsub notitle;
%inc sascde ; run;
```

Note the notitle option on the file statement. Also the SASCDE if allocated to internal reader, the JCL created in this dataset will be submitted at the end of SAS.

The SAS macro JOBGEN is as follows:

```
%macro JOBGEN;
infile INP EOF=EOF; file SASNJCL;
do while (1 = 1); input COL1 $1. @;
 if COL1  = '/'  then    do;        (a)
   input @1 INPLINE $CHAR72.;
   put "PUT "
   """" INPLINE """" ";"  ;  (b)   (1)
end;
else do;
   input @1 INPLINE $CHAR72.;
   put INPLINE;   end;    end;      (c)
EOF : stop;
%mend JOBGEN;
```

Note:

(a) If col 1 has '/' then it is JCL, then
(b) make it a SAS PUT statement
(c) If col 1 is not '/' then it is a SAS statement, then pass it as it is.

The SAS code (transparent to the user), reads dataset referred in the DDNAME INP and checks the first column. If it is a '/', then it is a JCL statement, otherwise it is a SAS statement. JCL statements are converted to PUT statements, others (SAS statements) are output in their unmodified form. This logic is represented in the SAS Macro named JOBGEN. Through the SAS FILE statement, these put statements are written to the dataset for SASCDE (ddname). At the end of this phase, the temporary file SASCDE is read and executed when the user selects the particular option. Also during this the SAS macro variable names are resolved. This method has several advantages: viz.

- from a common set of JCL statements, through program logic we can tailor the job stream.
- through conditional logic, a subset of of the JCL can be added, or modified etc., This means conditional generation of Job/s, job step/s
- ddname/s and individual JCL statements. The conditions can be based on date, site information, or data values during the program execution
- etc.,
- indirect referencing the SAS macro variables make it even more powerful.

Sample inputs and the resulting outputs are given on page 1, 2 of the appendix C. Even though very simple conditional SAS statements are used here, one can make it as complex as needed, combine with the macro variable resolution, direct and indirect, parameter passing for the JCL procedures, one can see how powerful this can be.

3.2 Online IVPs using TPNS scripts :

TPNS (TeleProcessing Network Simulator) is an IBM software that enables building of script to simulate onlines systems. TPNS can be used for functional, regression, and performance testing. TPNS can execute as an ACF/VTAM application program, and logical units(LU) can also be simulated. TPNS scripts can be

designed to generate the transactions for applications. For the menu driven IVPs TPNS Scripts were setup for the software products such as PANVALET, FILE-AID, SAS on-line, etc., TPNS (version 2, release3) was used. Only the most used functions of these products were represented in the scripts.

User needs to know only three commands in regard to TPNS testing:

- To bring up the test for the product
- To end the test and
- To look at the output from test

These commands are discussed below. When the user selects the on line product testing from the TPNS IVP menu, a TPNS clist is executed under SPF. At this point he/she enters the network name corresponding to the scripts defined for the particular software product to execute. During the test, user is informed of the status of the test at frequent intervals. At the end of the test, control is transfered back into the menu.

Each of the test scripts can be logically divided into three parts: (1) Logon, (2) processing of the product, and (3) Logoff. During all these phases, messages are sent regarding the status of the test. A copy of the Network, Logon, Script, and Logoff message decks for the INFO/SYS is given in the appendix B.

The TPNS network for this example works with LUTYPE LU2. VTAM APPLIDs TPNS2001 - TPNS2006 are used, and the LOGON IDs used are UPGMTS1, UPGMTS2, and UPGMTS3. The following libraries contain the network configuration, message decks, clist etc., that are necessary for the ON-LINE TPNS TESTS.

```
PPG1.TPNS.CNTL :   This contains the JCL
       for the preprocessor runs, runlog
       reformat, and related members.
TSO.USRCLIST   :   contains clists for
       preprocessor TPNS run, and Log
       formatting.
PPG1.TPNS.INITDD :  Network definitions
PPG1.TPNS.MSGDD  :  Message decks
```

To test an on-line product, the user selects option 2 on the TPNS IVP Menu. This response is translated into the execution of a TPNS clist. At this point, the user enters the name of the TPNS network (TPNS network name for the product) in the following format:

```
i network,s     <===  to run the script

where network: the name of the network
          defined on the NETWRK card of
          the TPNS scripts.
```

At the end of the TPNS test, user receives a message to that effect, and at this point, to end the TPNS net, the following command is enetred:

```
zend            <=== to stop the TPNS run
```

The tester will be in SPF at this point, and when exiting from SPF, he/she will be back in the TPNS IVP Menu. Then options are available to browse the formatted log data from these tests, either online, or hardcopy. Online option allows the user to execute a clist under SPF environment, and for the hardcopy, a Batch job is submitted. User also has the option to go into the SPF environment to check the output.

Using some of the TPNS utilities, the log is formatted, and this facilitates the user to look at the results of the online tests. This is achieved through option 2 of the TPNS log menu. This results in a branch to SPF/TSO. Now the user enters a clist name:

```
tpnsklog       <=== to format and browse
```

At the end of the browse, when the user exits SPF, returns back to the same IVP test menu.

3.3 Test results :

User has the option to browse the output of the tests at every logical point. Eg. From the Batch IVP menu, after the job has been submitted, user can look at the test results with SPF/SDSF. Similar option is available under TPNS testing. After each of these branches to SPF, the user is returned back to the IVP test system, so that he/she can continue any further testing.

4. Conclusion

SAS based systems are very powerful for many of the applications. Here it is shown how flexible, and easy it can be to use it in testing the installed software products. SAS/AF makes it very attractive through the menus and Help facility, and the processing behind screens can be transparent to the user. The System shown here is simple enough, but it can be enhanced through further exploitation of the SAS macro language and SAS/FSP.

Acknowledgements :

The author wishes to thank Joan Payne who reviewed the manuscript and made valuable comments.

*SAS is the registered trademark of SAS Institute, Inc., Cary, NC, USA.

Appendix A

```
I V P  Primary Menu

Select one of the options and press ENTER

  1   I V P  Batch Testing
  2   I V P  TPNS online Testing
  3   SPF Environment
  4   Using the  I V P  Test System
  5   EXIT
```

```
Batch  I V P  menu

Select one of the options and press ENTER

  1   Select the Product for testing
  2   Execute the test
  3   Browse the generated JCL
  4   Test Output - SPF/SDSF
  5   Exit
```

```
General description of IVP Menus

Type the number of your choice on the command
line and press the ENTER key.

 1  General Description of the Menu System
 2  I V P  Batch Testing
 3  I V P  TPNS  online Testing
 4  SPF Environment
 5  Types of screens in the Menu System
 6  Keys and commands
 7  Getting on-line help
 8  Return to I V P  Primary Menu
```

```
TPNS  I V P  menu

Select one of the options and press ENTER

(Look at option 1 - before using options 2-4)

     1   Select (online) Product - Help(PF1)
     2   Execute the TPNS test
     3   Browse TPNS log - online
     4   Hard copy of TPNS formatted log
     5   Exit
```

Appendix B

```
NETFIL4  NTWRK HEAD='MODEL TSOxx NETWORK',
               DELAY=F1,
               DISPLAY=(24,80),
               BUFSIZE=8192,
               FRSTTXT=TSOLOG34,
               ITIME=5,
               INIT=SEC,
               LUTYPE=LU2,
               MAXSESS=(0,1),
               MAXCALL=24,
               LOGDSPLY=BOTH,
               MSGTRACE=YES,
               OPTIONS=(CONRATE,MONCMND),
               PATH=(0),
               THKTIME=UNLOCK,
               USERAREA=100,
               SAVEAREA=(2,8),
               UTI=100
0        PATH  NETFAID,TSOLOGOF
         VTAMAPPL APPLID=TPNS9999
UPGMTS1  LU
TSOLOG34 MSGTXT
*        SET NC2=+1
         DATASAVE AREA=1,TEXT=($ID,7$)
0        IF LOC=RU+0,TEXT=('01'),THEN=E-INIT,
              STATUS=HOLD
1        IF LOC=RU+0,TEXT=('A0'),THEN=E-SDT,
              STATUS=HOLD
2        IF LOC=RU+0,TEXT=('32'),THEN=E-UNBIND,
              STATUS=HOLD
3        IF LOC=RU+0,TEXT=('31'),THEN=E-BIND,
              STATUS=HOLD
         CMND COMMAND=INIT,RESOURCE=TSOxx
*
* CHECK FOR ACF82003  ACF2, ENTER LOGON ID -
*
4        IF LOC=B+0,SCAN=YES,TEXT=(ACF82003),
            THEN=B-LOGONID,ELSE=WAIT,DELAY=CANCEL
W001     WAIT
         BRANCH LABEL=W001
LOGONID  LABEL
         TEXT    ($ID,7$)
         ENTER
*
* CHECK FOR ACF82004  ACF2, ENTER PASSWORD -
*
4        IF LOC=RU+0,SCAN=YES,TEXT=(ACF82004),
            THEN=B-PSWD,DELAY=CANCEL,ELSE=WAIT
W002     WAIT
         BRANCH LABEL=W002

PSWD     LABEL
         TEXT    (xxxxxx)
         ENTER
*
* CHECK FOR ACF01012  PASSWORD NOT MATCHED
*
4        IF LOC=RU+0,SCAN=YES,TEXT=(ACF81012),
                    THEN=B-BPWD,DELAY=CANCEL
```

```
*
* CHECK FOR ACF82012  ACF2, ENTER ACCOUNT -
*
5       IF LOC=RU+0,SCAN=YES,TEXT=(ACF82012),
           THEN=B-ACCT,DELAY=CANCEL,ELSE=WAIT
W003    WAIT
        BRANCH LABEL=W003
ACCT    LABEL
        TEXT    (9999)
        ENTER
*
*  IF THREE ASTERISKS, GENERATE AN ENTER KEY HIT
*
1       IF LOC=C-4,TEXT=(***),THEN=C-CLEAR,
             STATUS=HOLD,DELAY=CANCEL
*
* CHECK FOR IKJ56425I - USERID ALREADY IN USE
*
4       IF LOC=RU+0,SCAN=YES,TEXT=(IKJ56425I),
           THEN=E-DUPUSER,DELAY=CANCEL
*
*  AT READY PROMPT, BEGIN TSO TRANS PROCESSING
*
5       IF LOC=RU+0,SCAN=YES,TEXT=(READY),
           THEN=B-READY,DELAY=CANCEL,ELSE=WAIT
*
*  IF THREE ASTERISKS APPEAR (ACF2 MSG),
*       GENERATE AN ENTER KEY HIT
*
6       IF LOC=C-5,TEXT=(***),THEN=C-CLEAR,
             STATUS=HOLD,DELAY=CANCEL
        WAIT
WAIT    WAIT
        BRANCH LABEL=WAIT
        WTO (FALL THROUGH ERROR)
CLEAR   LABEL
        ENTER
        RETURN
INIT    LABEL
        WTO (SNA INIT-SELF RECVD - MSGDECK $MSGTXTID$)
        RETURN
BIND    LABEL
        WTO (BIND RECEIVED IN MSGDECK $MSGTXTID$)
        SETSW SW1=ON
        RETURN
SDT     LABEL
        WTO (START DATA TRAFFIC RECEIVED)
        IF WHEN=IMMED,LOC=SW2,THEN=SW3(ON)
        SETSW SW2=ON
        RETURN
UNBIND  LABEL
        WTO (UNBIND RECEIVED IN MSGDECK $MSGTXTID$)
        IF   WHEN=IMMED,LOC=SW3,THEN=E-UNBERR,
             ELSE=CONT
        RETURN
DUPUSER LABEL
        DEACT IFS=(5)
        DATASAVE AREA=U+0,LOC=*,LENG=80
        WTO (USER ALREADY LOGGED ON:)
        WTO ($RECALL,U+0,80$)
        RETURN
BPWD    LABEL
        DEACT IFS=(4,5)
        WTO (PASSWORD NOT MATCHED)
        TEXT (+)
        ENTER
        STOP
        QUIESCE
UNBERR  DATASAVE AREA=U+0,LOC=*,LENG=80
        WTO (UNEXPECTED UNBIND RECEIVED )
        WTO (  IN MSGDECK $MSGTXTID$ )
        WTO (LAST MSG: $RECALL,U+0,80$)
        QUIESCE
        RETURN
READY   LABEL
        SET NSEQ=+1
        SET NC1=+1
        SET NC3=+1
        WTO  (READY FOR TSO COMMANDS)
        DEACT IFS=ALL
ENDLOG34 ENDTXT

NETISYS  MSGTXT
*
* CHECK FOR IKJ56470I UIVPCT1 LOGGED OFF AT ...
*
0       IF LOC=RU+0,TEXT=(IKJ56470I),SCAN=YES,
           STATUS=HOLD,THEN=ETSOLOGOF-LOGOFF
*
1       IF LOC=RU+0,TEXT=('32'),
           THEN=ETSOLOGOF-UNBIND1,STATUS=HOLD
*
*  IF THREE ASTERISKS, GENERATE ENTER KEY HIT
*
2       IF LOC=C-4,TEXT=(***),
           THEN=CTSOLOGON-CLEAR,STATUS=HOLD
*
*  WE ARE AT THE READY PROMPT,
*  BEGIN TSO TRANSACTION PROCESSING
*
        TEXT (se ''Hello! From TPNS . u{{xxx}})
        ENTER
        STOP
        EREOF
        TEXT (se ''Test INFO/SYS '' u{{Uxxxx}})
        ENTER
        STOP
READY   LABEL
*
* Select  INFOSYS  from  spf menu
*
        TEXT (spf)
        ENTER
        STOP
        TEXT (i)
        ENTER
        STOP
          .
          .
          .
          .
*
        WTO   ( End of INFOSYS Test - QUIT )
*
        TEXT (x)
        ENTER
        STOP
*
* End of Infosys
*
        STOP
3       IF    LOC=RU+0,TEXT=(READY),STATUS=HOLD,
              SCAN=YES,THEN=CONT,ELSE=WAIT
      TEXT (SE ''End of TPNS test, bye ! '' )
        ENTER
        STOP
ENDSAS  ENDTXT

TSOLOGOF MSGTXT
*********************************************
* Model TPNS LOGOFF Message Deck.           *
*********************************************
*
* CHECK FOR IKJ56470I UIVPCT1 LOGGED OFF AT ...
*
0       IF LOC=RU+0,TEXT=(IKJ56470I),SCAN=YES,
           STATUS=HOLD,THEN=E-LOGOFF
*
1       IF LOC=RU+0,TEXT=('32'),THEN=E-UNBIND1,
             STATUS=HOLD
        WTO (ISSUING LOGOFF COMMAND)
*       DEACT IFS=(0,1,3)
        TEXT (logoff)
        ENTER

*
* CHECK FOR IKJ56470I UIVPCT1 LOGGED OFF AT ...
*
0       IF LOC=B+0,TEXT=(IKJ56470I),SCAN=YES,
           SCAN=YES,STATUS=HOLD,
           THEN=E-LOGOFF,ELSE=WAIT,DELAY=CANCEL
1       IF   LOC=RU+0,TEXT=('32'),THEN=CONT,
             STATUS=HOLD
        WAIT
UNBIND  LABEL
        DEACT IFS=ALL
        WTO  (UNBIND RECEIVED FOR USER $ID,7$)
        WTO  (END OF MESSAGE DECK $MSGTXTID$)
        WTO (UNBIND RECEIVED)
        WTO (QUIESCING)
        SET  NC1=-1
        IF  WHEN=IMMED,LOC=NC1,TEXT=0,THEN=CONT,
            ELSE=B-QUIESCE
        WTO (ALL TERMINAL SESSIONS HAVE QUIESCED )
        WTO ($CNTR,NC2,2$ SESSION((S)) STARTED )
        WTO ($CNTR,NC3,2$ SESSION((S)) LOGGED ON)
        WTO ($CNTR,NSEQ,2$  ..  LOGGED OFF)
QUIESCE QUIESCE
CLEAR   LABEL
        ENTER
        RETURN
```

```
LOGOFF   LABEL
         DEACT IFS=(0)
         DATASAVE AREA=U+0,LOC=*,LENG=80
         WTO (LOGOFF REPLY RECEIVED FROM HOST:)
         WTO ($RECALL,U+0,100$)
         WTO (WAITING FOR UNBIND)
         RETURN
UNBIND1  LABEL
         DATASAVE AREA=U+0,LOC=*,LENG=80
         WTO (UNEXPECTED UNBIND RECEIVED )
         WTO (IN MSGDECK $MSGTXTID$)
         WTO (LAST MSG: $RECALL,U+0,80$)
         QUIESCE
ENDLOGOF ENDTXT
```

Appendix C

```
         //**
         //****------   JOB # 1  ----------******
         //**
   (1)   //JOB&JID1.&JID2 JOB (XXXX,XXXX),
         //          'KANTHAN KL',CLASS=X
         /*JOBPARM S=ZZZ
         /*ROUTE   PRINT  LOCAL


         //**
         //***--------  JOB # 2  ----------*****
         //**
   (2)   //JOBN&JID2 JOB (YYYY,XXXX),
         //          'KANTHAN KL ',CLASS=X


      IF WEEKDAY(TODAY()) GT 2 THEN DO;
      //***-----------------------------------***
      //***    THIS  JOB IS ADDED for tuesday **
 (3)  //***        - saturday                  **
      //***-----------------------------------***
      //JOBNAME1 JOB (ZZZZ,ZZZZ),'KANTHAN KL ',CLASS=X
      //STEP11   EXEC SAS,OPTIONS='MPRINT',REGION=1024K
      //STEPLIB DD  DSN=XXXG.SAS.LIBRARY,DISP=SHR
      //DDN1     DD  DSN=DSNANE.JOB1(&MEMBER),DISP=SHR
      END;

         //**
         //**-------   JOB  # 3    ----------***
         //**
         //JOBNAME3 JOB (XXXX,YYYY),'KANTHAN  KL',CLASS=X
         /*JOBPARM S=ANY
         /*ROUTE PRINT SRSC
         //*************************************
   (4)   //SELSMF   PROC  SYS='A',SMF1='0990.',SMF2='DSK'
         //SEL     EXEC  PGM=XXXXXX
         //INFIL     DD  DSN=ZZZZZ.&SYS&SMF1&SYS&SMF2,DISP=SHR
         //OUTRMF    DD  DSN=&&RMF&SYS,DISP=(NEW,PASS),....
         //SYSIN     DD  DSN=DSNAME.CODE(MEMBER),DISP=SHR
         //        PEND

         //STEP1  EXEC SAS,.....
         //DDNAMXX  DD DSN=XXXXX
         //         DISP=&DISP,SPACE=(TRK,(0)),UNIT=SYSDA
         IF "&SID" NE 'X' THEN
   (5)   //DDN1     DD  DSN=DDNAME.FOR.NOT.X,DISP=SHR
         ELSE
         //DDNAM2   DD  DSN=DDN.YYYYYY,DISP=&DISP1
         //DNAMX  DD DSN=DSNAM.&SID.SUFX.SUFX2,DISP=(,CATLG,DELETE),
         //         UNIT=SYSDA .....
         //SRCLIB   DD  DSN=SOURCE.LIB.YYY,DISP=SHR
         //SYSIN    DD  DSN=SAS.TEST(&MEMBER),DISP=SHR


         IF WEEKDAY(TODAY()) GT 2 THEN DO;
         //*****************************
         //SELC    EXEC  SELSMF,SYS='C'
         //*********************************
         END;
         ELSE DO;
         //*****************************
         //SELA    EXEC  SELSMF
         END;
   (6)   IF WEEKDAY(TODAY()) EQ 3 THEN DO;
         //STEP4    EXEC  SAS
         //DDNAME   DD  DSN=.............,DISP=SHR
         //SYSIN    DD  DSN=.............,DISP=SHR
         //*****************************
         END;
```

```
         //****
         D = TODAY(); %_WK;
         IF WEEK = 50 THEN DO;
         //***
   (7)   //***----   RUN ON xxTH WEEK ONLY  ---****
         //***
         //STEP4    EXEC  SAS
         //DDNAME   DD  DSN=.............,DISP=OLD
         //SYSIN    DD  DSN=.............,DISP=SHR
         //*****************************
         END;


         //**
         //****---------      JOB # 1  -------------------***
         //**
   (1)   //JOBJ1J2 JOBY(XXXX,XXXX),'KANTHAN KL',CLASS=X
         /*ROUTE   PRINT  LOCAL
         //***-------------------------------------------------
         //***    THIS  JOB IS ADDED IN THE BASED ON THE  DAT
         //***-------------------------------------------------
   (3)   //JOBNAME1 JOB (ZZZZ,ZZZZ),'KANTHAN KL ',CLASS=X
         //STEP11   EXEC SAS,OPTIONS='MPRINT',REGION=1024K
         //STEPLIB DD  DSN=XXXG.SAS.LIBRARY,DISP=SHR
         //DDN1     DD  DSN=DSNANE.JOB1(MEMBER01),DISP=SHR
         //**
         //***----------      JOB # 2    ---------------*****
         //**
   (2)   //JOBNJ2 JOB (YYYY,XXXX),'KANTHAN KL ',CLASS=X
         //STEP1  EXEC SAS,.....
         //DDNAMXX  DD DSN=XXXXX
         //         DISP=NEW,SPACE=(TRK,(0)),UNIT=SYSDA
   (5)   //DDN1     DD  DSN=DDNAME.FOR.NOT.X,DISP=SHR
         //DNAMX  DD DSN=DSNAM.TSUFX.SUFX2,DISP=(,CATLG,DELET
         //         UNIT=SYSDA .....
         //SRCLIB   DD  DSN=SOURCE.LIB.YYY,DISP=SHR
         //SYSIN    DD  DSN=SAS.TEST(MEMBER01),DISP=SHR
         //**
         //**-------       JOB  # 3 -------------------------
         //**
         //JOBNAME3 JOB (XXXX,YYYY),'KANTHAN  KL',CLASS=X
   (4)   /*JOBPARM S=ANY
         /*ROUTE PRINT SRSC
         //*************************************************
         //SELSMF   PROC  SYS='A',SMF1='0990.',SMF2='DSK'
         //SEL     EXEC  PGM=XXXXXX
         //INFIL     DD  DSN=ZZZZZ.&SYS&SMF1&SYS&SMF2,DISP=SH
         //OUTRMF    DD  DSN=&RMF&SYS,DISP=(NEW,PASS),....
         //SYSIN     DD  DSN=DSNAME.CODE(MEMBER),DISP=SHR
         //        PEND
         //*****************************
   (6)   //SELC    EXEC  SELSMF,SYS='C'
         //*********************************
         //****
         //***
   (7)   //***----   RUN ON xxTH WEEK ONLY  ---****
         //***
         //STEP4    EXEC  SAS
         //DDNAME   DD  DSN=.............,DISP=OLD
         //SYSIN    DD  DSN=.............,DISP=SHR
         //*****************************
```