Converting MVS/JCL to REXX/TSO

Hobart Spitz
MTA New York City Transit

Converting MVS/JCL to REXX/TSO

Presented at REXX Symposium

May '94

Hobart Spitz

MTA New York City Transit

718-694-3112

5520808@MCIMail.com

Converting MVS/JCL to REXX/TSO

Abstract:

The speaker will discuss his experiences in using REXX/TSO in place of MVS/JCL. The advantages of REXX over JCL will be covered, as will a step-by-step methodology for converting existing JCL to REXX for batch and/or interactive use. JCL to REXX/TSO equivalents will be spelled out in detail. Guidelines and techniques for portability positioning to VM, OS/2, etc. will be reviewed.

Speaker:

Hobart Spitz (SBW)
MTA New York City Transit
130 Livingston St. 5041A
Brooklyn NY 11202

Phone: 718-694-3112
Alternate Voice Mail: 718-694-1719
Fax: 718-694-4309
E-mail (internet): 5520808@MCIMail.com

Application Backgrounds

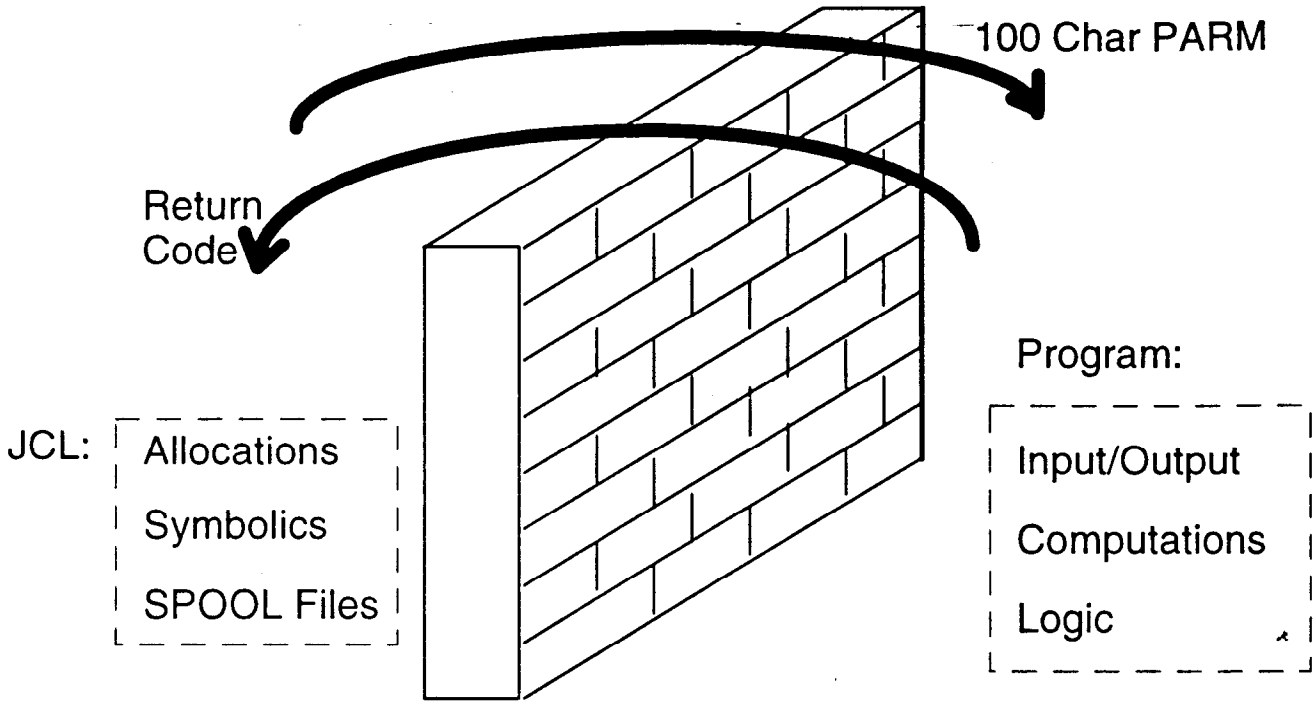
NYNEX Computer Services - Billing Service Bureau - Multiple Clients

- DB2 with 3rd party host command interface
- CICS Transaction Processing
- TSO Scheduler Access
- Limited TSO User Access
- Original Batch Design:
  JCL, COBOL II.
- Final Application:
  REXX, JCL, COBOL II.

New York City Transit Authority - Change Control Managment System

- Data stored in VSAM file and ISPF tables
- Entirely TSO Based user access
- Original Batch Design:
  ISPF Skels, JCL, COBOL.
- Final Application:
  REXX, ISPF Skels, JCL, COBOL.

"The Great Wall of MVS Batch"



Limitations of MVS/JCL

- Rigid isolation between JCL level (allocation, SPOOL datasets, symbolics, and return codes) and program level (I/O, computation, and logic).
- No interaction between application data and application control.
- Limit of 100 characters in PARM=.
- Minimal logical operators, even with new MVS features.
- Heavy manual intervention requirements in most cases.
- Single level PROC invocation, until recently.

Advantages and Benefits of REXX/TSO Over JCL

- Automation
- Simplification
- Readability, Write-ability, and Maintainability
- Modularity
- Environment dependant code can be isolated. Portability and Reusability is feasible between Foreground TSO, Batch, VM, MS-DOS, OS/2, Windows, and, maybe someday, CICS.
- Flexibility/Control Structures - Looping, General Conditionals, Expressions, PARSEing
- Controlled Recovery and Restart
- Up-front Validation and Handling of Clerical Errors
- Addresses Batch Window Criticality
- Reduced Programming Requirements
- Almost Unlimited Procedure Invocations Levels, Including Recursion
- Application data and control can interact
- Avoids MVS Steps per JOB limit.

In short, every // costs you time and your installation money.

Conversion Steps

- Extract each JCL step to REXX EXEC by PGM= and PROC name.
- Change DDs to equivalent "ALLOCATE . . .":
  - KEYWORD=VALUE becomes KEYWORD(VALUE).
  - Subparameter, KEYWORD=(SUBPARM=VALUE), becomes parameter, SUBPARM(VALUE).
  - Add quotes around permanent dataset names.
- Move each // EXEC PGM= to end of its step.
- After EXEC PGM= effect normal disposition:
  IF RC = 0 THEN "FREE DD(SYSUT2. . .) CATALOG"
- Change EXECs to equivalents.
- Drop IEFBR14 DELETE/ALLOCATE; use SYSDSN().
- Create JOB stream to invoke converted REXX module.

EXEC Equivalents

JCL		REXX/TSO Allocate Parameters
// EXEC PGM=ppp,[PARM=xxx]		ADDRESS ISPEXEC "SELECT PGM(ppp) [PARM(xxx)]"
// EXEC [PROC=]mmm,kwd=val		ADDRESS ISPEXEC "SELECT CMD(%mmm kwd=val . . .)"
	or	ADDRESS TSO "%mmm kwd=val . . . "
	or	CALL mmm kwd=val . . . (mmm has also been converted to REXX/TSO.)
// PROC kwd=val		ARG "kwd="kwd . if kwd = "" then kwd = "val"
	or	ARG kwd1 kwd2 kwd3
// EXEC . . . ,COND=(0,NE)		IF RC = 0 THEN . . . after commands. IF RESULT = 0 THEN . . . after REXX CALL. Save RC/RESULT for complex or deferred tests.

DD Dataset Parameter Equivalents

JCL	REXX/TSO Allocate Parameters
DSNAME=q1.q2.q3 DSN=q1.q2.q3	DSNAME('q1.q2.q3')
DISP=(OLD,KEEP,DELETE)	OLD DELETE (no wait, see SOMVSE93039) . . . IF RC = 0 THEN "FREE DDNAME(...) KEEP"
DCB=(model.dsn,BLKSIZE=bbb, LRECL=lll,RECFM=abc)	LIKE('model.dsn') BLKSIZE(bbb) LRECL(lll) RECFM(a b c)
VOL=SER=(vvvvvv,volcount)	VOLUME(vvvvvv) MAXVOL(volcount)
LABEL=(n,ll,EXPDT=yyddd)	POSITION(n) LABEL(ll) EXPTD(yyddd) (ddd = 0 not valid; IBM future direction.)
UNIT=(uuuu,n)	UNIT(uuuu) UCOUNT(n)
SPACE=(CYL,(pp,ss,dd),RLSE) SPACE=(800,(pp,ss),,ROUND)	CYL SPACE(pp ss) DIR(dd) RELEASE BLOCK(800) SPACE(pp ss) ROUND

Other DD Equivalents

JCL	REXX/TSO Allocate Parameters
Label on DD	DDNAME(...)
Concatenated DD	DSN('q1.q2.q3' 'q4.q5.q6')
Repeated DD across steps	REUSE required in absense of FREE.
DD * or DD DATA	"ALLOC DD(dddd) UNIT(VIO)", "TRACK SPACE(1 1)", "RECFM(F B) LRECL(80) BLKSIZE(4000)" QUEUE "information" QUEUE "more information" QUEUE "" "EXECIO * DISKW dddd (FINI"
SYSOUT=c SYSOUT=*	SYSOUT(c), c ^= *. DSN(*) (output goes to SYSTSPRT)
DEST=rmt	DEST(rmt)
HOLD=YES/NO	HOLD/NOHOLD
COPIES=n	COPIES(n)
FORMS=ltrh	FORMS(ltrh)
OUTPUT=opnam	OUTDES(opnam)
DUMMY	DUMMY
JOBLIB, STEPLIB, ISPLLIB // OUTPUT, SUBSYS=	retain - no direct eqvuivalents.

Example - Original JCL

```
//SHARE81C   JOB     (...),...
//*
//TAP2DSK    PROC    MEM=
//      EXEC  PGM=IEFBR14
//XXX     DD  DISP=(MOD,DELETE),
//            UNIT=SYSDA,
//            SPACE=(TRK,1),
//            DSN=NCSCB40.OUTPUT.TEXT
//*
//      EXEC  PGM=IEBGENER
//SYSUT1        DD    DISP=OLD,
//            VOL=SER=C12345,UNIT=TAPE,
//            LABEL=(3,BLP,EXPTD=98000),
//            DCB=(RECFM=FB,LRECL=82,BLKSIZE=8200,
//            OPTCD=Q),
//            DSN=TAPE.INPUT
//SYSUT2        DD    DISP=(,CATLG,DELETE),UNIT=SYSDA,
//            SPACE=(CYL,(2,5,10),RLSE),
//            DCB=(RECFM=VB,LRECL=100,BLKSIZE=10000),
//            DSN=NCSCB40.OUTPUT.TEXT(&MEM)
//SYSPRINT DD SYSOUT=*
//SYSIN   DD  DUMMY
//      PEND
//*
//      EXEC TAP2DSK,MEM=D930722
```

Example - REXX Equivalent

```
//SHARE81C  JOB (...)...
//         EXEC TDCMUTR1,CMD='%TAPTODSK'     (batch "logon" proc)
```

TAPTODSK

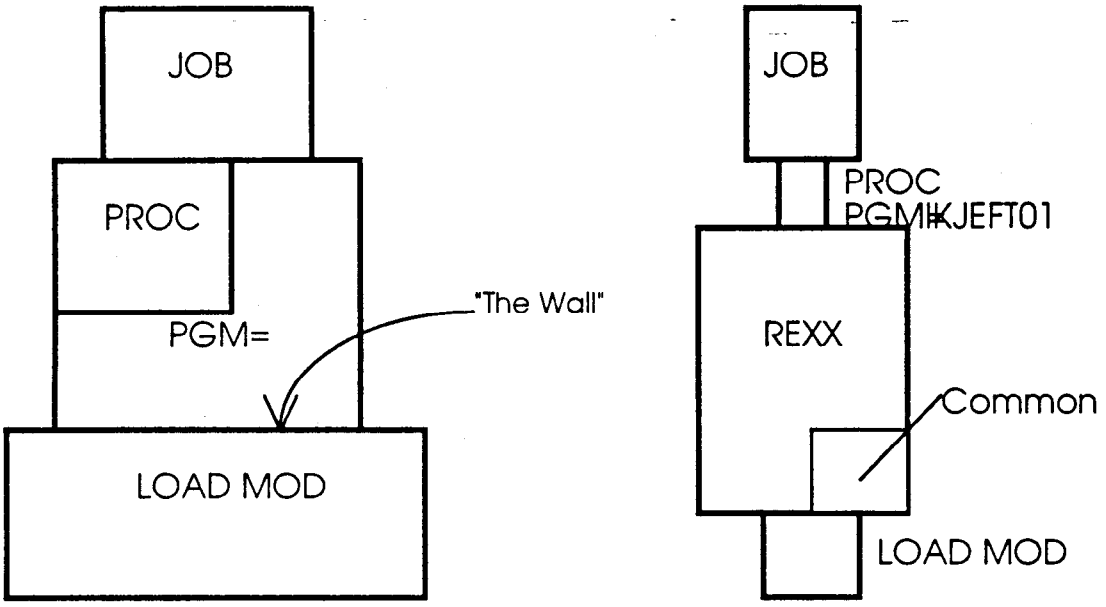
```
/* REXX */
IF SYSDSN("'NCSCB40.OUTPUT.TEXT'") = "OK" THEN DO
    "DELETE OUTPUT.TEXT"
    END

SIGNAL ON ERROR
"ALLOCATE REUSE DDNAME(SYSUT1) OLD UNIT(TAPE)",
  "VOL(C12345) POSITION(3) LABEL(BLP) EXPDT(98001)",
  "RECFM(F B) LRECL(82) BLKSIZE(8200) OPTCD(Q)",
  "DSN('TAPE.INPUT')"
"ALLOCATE REUSE DDNAME(SYSUT2) NEW DELETE",
  "UNIT(SYSDA) CYL SPACE(2,5) DIR(10) RELEASE",
  "RECFM(V B) LRECL(100) BLKSIZE(10000)",
  "DSN('NCSCB40.OUTPUT.TEXT(D"RIGHT(DATE("S"),6)"'))"
SIGNAL OFF ERROR
"GENER"
RETURN RC
```

GENER

```
/* REXX */
/* IEBGENER DRIVER */
SIGNAL ON ERROR
"ALLOCATE REUSE DDNAME(SYSPRINT) DSN(*)"
"ALLOCATE REUSE DDNAME(SYSIN)    DUMMY"
SIGNAL OFF ERROR
ADDRESS ISPEXEC "SELECT PGM(IEBGENER)"
CONDCODE = RC
IF RC = 0 THEN "FREE DDNAME(SYSUT2) CATALOG"
RETURN CONDCODE
```

Code Volume Perspective



Classic REXX

Converted System:

- Fewer JOB Streams
- Near Elimination of JCL
- Reduction of Compiled Language Application Code
- Increased Modularity
- Facilitates isolation of host dependant code and creates portability.

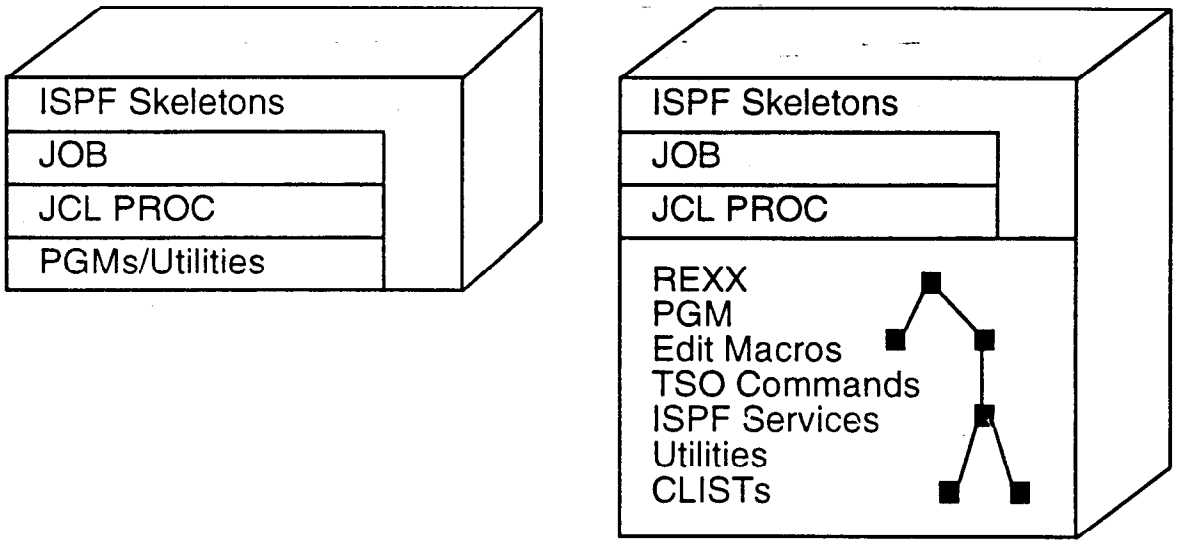
Analysis of Actual Batch System

NYCTA's CCM Release 2.1 had one PROCLIB consisting of 51 members containing 191 steps, calling 19 programs. Using 1:1 as an approximate ratio for JCL statement to REXX host command ratio, these 191 steps should be replaceable by 70 (51+19) REXX EXECs of approximately the same length, one for each // EXEC. In practice, the results were much better as most of the 51 JCL PROCs were replaced by a few REXX main modules plus about 10 driver modules to call language processors (LKED, COBOL, COBOL II, CICS, etc.).

Conversion is Ideal for Initial Usage of REXX

- Nearly all JCL has direct REXX or TSO equivalents.
- Low exposure.
- Low cost.
- Process can be automated.
- High benefit.
- Probability of success is high.
- Required software is already in-house in most shops.

Batch Comparison



Batch REXX:

- Allows Integration of Software: Allocation, Utilities, Application.
- Supports Multiple Levels of Invocation and Common Modules.
- Removes Most Requirements for Manual Intervention: Overrides, Control Cards, etc.

Summary

- Brings Batch into '90s - Portably and Productively
- Provides Real Programming Constructs
- Enables Application Based Scheduling, Recovery, etc.
- Keeps Pace with PCs and Minis
- Breaks Down "Wall" Between Control and Software Functions
- Opens Exciting New Possibilities for Batch Processing