

Standardizing SAS Code for Quality Programs

Clarence Wm. Jackson, CQA, Change Manager

City of Dallas, Communication and Information Services, Change Management Group

Abstract and Introduction

SAS software is a powerful programming system that allows even casual users to write very complicated solutions for business problems. It is free form, meaning that it has no syntactical constraints for defining the program structure, as is found in such programming languages as COBOL and ALC. However, the free form of SAS code can be a mixed blessing, where code maintenance is concerned. Whenever changes in the SAS code are required, if someone other than the original programmer has to make the changes, this may result in errors, lost productivity and a reduction in the consistency and overall quality of the program. This situation may be avoided by the implementation of appropriate standards for the writing of SAS programs. This paper will review industry standards for other programming languages, and will discuss how SAS code could be standardized. The paper also will review the benefits and opportunities for quality improvement.

Standards Provide Basis for Quality

What is a "Standard"? The American Heritage dictionary defines "standard" as

"an acknowledged measure of comparison for quantitative or qualitative value; criterion; norm; a degree or level of requirement, excellence or attainment."

There are two categories of standards related to Information Technology, 'industry' and 'installation'. The industry standards are those set by a recognized standards organization meant to facilitate the efficient interchange of information between organizations and companies. The installation standards serve the purpose of providing guidelines for the efficient operation of a single installation or shop.

Why Use a Standard? There are many reasons why to use a standard. The main reasons are:

1. To create software products which are consistent with one another
2. To create quality software products;
3. To establish levels of acceptance for both system development and system maintenance activities

When to Use a Standard? Always if possible! When applied to manufacturing, industry standards ensure that one product is compatible with another. For instance, light bulbs are made to standards accepted by the manufacturers and the consumers. This benefits the manufacture by providing monetary savings through use of mass production to reduce development costs. This benefits the consumer by reducing the cost of goods, and enable adequate evaluation of the quality of light bulbs from competing vendors.

Standards also seek to ensure that the finished product meets a specified level of quality. Software standards should be developed specifying the technical qualities in terms of software metrics that software of product must meet. These standards should also specify technical standards of accuracy and readability that documentation must meet. Software development can be likened to the manufacturing of engineering products such as light bulbs. The steps to produce both software and hardware are the same from an engineering point of view. Each production cycle is expressed in the development life cycle of planning, designing, building, testing, implementing, and maintaining.

Programming Standards

Because of the statement syntax of some programming languages, there are certain coding constraints that guide the structure of programs that force compliance to some standards. For instance, COBOL, ALC and other languages require certain card image columns be used and other not used, comments to be coded in a specific column, and 'required' sections.

SAS on the other hand is free style, in that there is no coding syntax that contributes to statement structure. SAS does not require sections to be included. One could code a SAS program without the use of a 'DATA' or 'PROC' step (although one step name must be used). This is good and bad. One could code several SAS statements or even steps on the same line or spread the statement over several lines with one keyword per line. The options depend on what standards are in place, and the programmer's personal style.

There should be a standard for constructing SAS code. Those standards should define statement structure, naming structures for datasets, variables, programs, stored code, macro, libraries, etc, documentation of program code, and change and version control. Then these standards should be communicated to all SAS programmers within the installation or shop so that everyone is aware of them and use them in future programming tasks.

Program Maintenance Issues

Programs must be "maintained", and until the program is retired or replaced, this is the phase of the life cycle that will incur the highest cost. Some programming maintenance issues are:

- } Changes in source data
- } Programming requirements
- } Operating environment
- } Bug fixes
- } ABENDS
- } Upgrades
- } Organizational changes
- } Enhancements

SAS programs are like all others in that they must be maintained, and sometimes by someone other than the original programmer. The time and effort spent on 'reviewing' the code prior to making changes increases when someone 'new' must modify the program.

Not having a standard or even a common style of programming can put SAS programs in the 'hard to maintain' group in major IT shops. Why? Because the other programming languages have had industry standards in place for years, a COBOL or ALC programmer will understand the code faster in a new shop during job changes. Plus, these programming languages do not allow the programmer to deviate (much!) in programming style, making them easier to standardize. There are specific rules in writing COBOL and ALC programs that must be learned in order to use the language, and these rules are in alignment with industry standards. Installation standards take these rules one-step further, and require that these installation standards be adhered to in order for programs to be placed in production. This makes those programs easier to maintain. (Although it will take one longer to code a COBOL program than SAS!)

Industry Standards in Software Engineering

Standards can be used for all phases of the development life cycle. There are several standards that are used industry wide. The ANSI and ISO standards are the most influential to software coding, but there are others that define standards for software development process (or the life cycle). The software life cycle comes in many forms (Waterfall, RAD, JAD, etc), but are the same steps taken in building most anything (right). The many life cycles all share these common objectives:

- } Define the business problem to be solved and
- } Select a set of solutions tools
- } Design the solution tool
- } Build and test the solution tool
- } Approve and Implement the solution tool
- } Maintain and improve on the solution tool

Standards such as SEI/CMM, ISO, ANSI, and others, seek to provide the following benefits:

- } Improve documentation
- } Reduce maintenance
- } Improve reliability
- } Improve quality
- } Optimize operations

SAS Programming Standards

Some simple standards that can be applied to SAS programming are:

- } Statement structure
 - a. Use indents and white space
 - b. No more than one statement per line
 - c. Define 'INPUT' and 'PUT' variables one per line using '@' pointer control
- } Name structures
 - a. Datasets and filenames should be meaningful

- b. Variable names should be clearly defined
 - c. Macros, stored code, and called items should follow a standard
 - d. Program name
- } Documentation
- a. Program documentation should be required
 - b. Each step should be preceded by documentation regarding purpose
 - c. Change control
 - i. Should be defined within the installation (shop)
 - ii. Should be recoverable in case there's a problem
 - iii. Should cover all changes made to program and include dates

Conclusions

Improving the quality of SAS programs by using a standard for writing SAS programs is needed as SAS programs become more mainstreamed in IT shops. By installing standards, the maintenance of SAS programs is reduced and reliability increased.

Contact Information

Clarence Wm. Jackson, CQA

Change Manager

City of Dallas Communication and Information Services

Change Management Group

1500 Marilla St 4DS

Dallas, TX 75201

Cljacks@ci.dallas.tx.us (work)

CJac@compuserve.com (home)

<http://ourworld.compserve.com/homepages/CJac>

SAMPLE SAS Program Documentation

PAGE;

TITLE 'CITY OF DALLAS, CIS CHANGE MANAGEMENT GROUP';
TITLE2 "PROGRAM NAME IS &SYSJOBID, ROSCOE MEMBER QA6.XXXXXX";

*** SAS PROGRAM DOCUMENTATION ***

201.XX PROGRAM PURPOSE

201.01 PROGRAM IDENTIFICATION

PROGRAM DEVELOPED ON ROSCOE MEMBER QA6._____

BY CLARENCE JACKSON DATE _____

REQUEST NUMBER	DATE	PROGRAM CODED BY (AUTHOR, LAST CHANGED)	REQUEST DESCRIPTION
ORIGINAL	MMDDYY	CLARENCE JACKSON	

201.02 PROGRAM DESCRIPTION

201.05 DATA SET REFERENCE

FILE
DATA SET NAME

201.06 REPORT TITLES

REPORT TITLE	REPORT NUMBER	DISTRIBUTED REPORT TO

*** END OF DOCUMENTATION SECTION ***

;;

PAGE;

SAMPLE ALC Program

```
SAMPLE  TITLE 'SAMPLE ALC PROGRAM'
        INIT
        OPEN  (FILEIN, INPUT, PRINTER, OUTPUT)
*-----*
*          PROCESSING RECORDS          *
*-----*
GETRCD  EQU  *
        GET   FILEIN, CARD_AREA
        MVC   DATA_AREA (L'CARD_AREA) , CARD_AREA
        BAS   RA, PRINT
        B     GETRCD
PRINT   EQU  *
        PUT   PRINTER, LINE
        BR    RA
TOCLOSE EQU  *
        CLOSE (FILEIN, , PRINTER)
        EOS
LINE    DS   0CL133
LINEASA DC   CL01' '
DATA_AREA DC CL132' '
CARD_AREA DS   CL80
PRINTER DCB   DSORG=PS, MACRF=PM, DDNAME=PRINTER
FILEIN   DCB   DSORG=PS, MACRF=GM, DDNAME=FILEIN, EODAD=TOCLOSE
        END
```

SAMPLE COBOL (1/7)

```
* MEMBER: STUBCOBA (COBOL/AMIGOS PROGRAM STUB) * 00000020
*-----* 00000030
***** 00000040
* IDENTIFICATION DIVISION * 00000050
***** 00000060
IDENTIFICATION DIVISION. 00000070
00000080
00000090
PROGRAM-ID. PGMNAME. 00000100
AUTHOR. FIRSTNAME LASTNAME. 00000110
DATE-WRITTEN. MM/DD/YY. 00000120
DATE-COMPILED. 00000130
00000140
*-----* 00000150
* SYSTEM NAME: _____ 00000160
* SYSTEM ID: _____ 00000170
* PROGRAM ID: _____ 00000180
* ACCOUNT CODE: _____ 00000190
*-----* 00000200
* ACCESS (DELETE, FILE (AMIGOS, 00000210
* FILES ACCESSED READ, ADD, UPD) VSAM, DB, QSAM) 00000220
* _____ 00000230
* _____ 00000240
* _____ 00000250
* _____ 00000260
* _____ 00000270
*-----* 00000280
* PROGRAM PURPOSE: 00000290
* 00000300
* DESCRIBE HERE THE PURPOSE/OBJECTIVE OF THE PROGRAM. 00000310
* USUALLY THIS INFORMATION CAN BE TAKEN FROM THE PROGRAM 00000320
* SPECIFICATION, PROGRAM PURPOSE. REPEATED HERE, IT COULD 00000330
* BE OF IMMEDIATE AND GREAT VALUE TO THE PROGRAMMER. 00000340
* 00000350
*-----* 00000360
* SUBROUTINES: 00000370
* LIST ANY SUBROUTINES HERE. 00000380
* 00000390
* 00000400
* 00000410
*-----* 00000420
EJECT 00000430
*-----* 00000440
* PROGRAM RETURN CODES AND MEANINGS: 00000450
* 00000460
* 0000 - SUCCESSFUL RUN. 00000470
* 0001 - DESCRIBE THE MEANING OF THIS RC IF APPLICABLE. 00000480
* 0002 - DESCRIBE THE MEANING OF THIS RC IF APPLICABLE. 00000490
* 0003 - DESCRIBE THE MEANING OF THIS RC IF APPLICABLE. 00000500
* 0004 - FILE ACCESS ERROR. 00000510
```

SAMPLE COBOL (2/7)

```

*
*-----*
      EJECT
00000520
00000530
00000540
00000550
00000560
*****
*          E N V I R O N M E N T      D I V I S I O N          *
*****
      ENVIRONMENT DIVISION.
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
*****
*          D A T A      D I V I S I O N          *
*****
      DATA DIVISION.
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900
*-----*
*          WORKING STORAGE STARTS HERE          *
*-----*
      WORKING-STORAGE SECTION.
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
*-----*
*  LIBRARIAN AUDIT TRAIL          DATE: 09/27/91  *
*-----*
01  LIBRARIAN-AUDIT-STAMP.
00001000
00001010
00001020
00001030
00001040
      05  FILLER          PIC X(16) VALUE 'AUDIT-STAMP****'.

```

SAMPLE COBOL (3/7)

```

05  AUDIT-MODNAME      PIC X(08) VALUE '[MODNAME'.          00001050
05  AUDIT-DATEUPD      PIC X(08) VALUE '[DATEUPD'.          00001060
05  AUDIT-TIMEUPD      PIC X(08) VALUE '[TIMEUPD'.          00001070
05  AUDIT-LVNO         PIC X(05) VALUE '[LVNO'.             00001080
05  AUDIT-UPNO         PIC X(05) VALUE '[UPNO'.             00001090
05  AUDIT-DATEACS      PIC X(08) VALUE '[DATEACS'.          00001100
05  AUDIT-TEMPDAT      PIC X(08) VALUE '[TEMPDAT'.          00001110
05  AUDIT-LIB-DSN      PIC X(44) VALUE                      00001120
    '[DATA-SET-NAME-FOR-THE-LIBRARIAN-MASTER-FILE'.        00001130
05  FILLER              PIC X(16) VALUE '****AUDIT-STAMP'.   00001140
                                                                00001150
*-----* 00001160
*              PROGRAM FLAGS                                * 00001170
*-----* 00001180
01  PROGRAM-FLAGS.                                          00001190
    02  AMIGOS-RETURN-CODE      PIC S99  VALUE +00 COMP.    00001200
        88  RETURN-CODE-IS-GOOD  VALUE +00.                00001210
        88  RETURN-CODE-IS-BAD   VALUES +01, +02,          00001220
                                +04 THRU +09.                00001230
        88  END-OF-AMIGOS        VALUE +03.                00001240
                                                                00001250
    EJECT                                                    00001260
                                                                00001270
*-----* 00001280
*              PROGRAM WORK AREAS                          * 00001290
*-----* 00001300
01  PROGRAM-WORK-AREAS.                                    00001310
                                                                00001320
                                                                00001330
    02  PROGRAM-COUNTERS.                                    00001340
        03  AMIGOS-RECORD-COUNTER  PIC 9(7) COMP-3  VALUE 0. 00001350
        03  PRINT-PAGE-COUNTER     PIC 9(5) COMP-3  VALUE 0. 00001360
        03  PRINT-LINE-COUNTER     PIC 9(3) COMP-3  VALUE 0. 00001370
                                                                00001380
    02  PROGRAM-VARIABLES.                                    00001390
        03  AMIGOS-COMMAND          PIC X(5) VALUE SPACES.  00001400
                                                                00001410
        03  AMIGOS-TAG.              00001420
            04  AMIGOS-FILE          PIC X(8) VALUE 'ERICH'. 00001430
            04  AMIGOS-ERROR-CODE    PIC S99  VALUE +0 COMP. 00001440
                                                                00001450
*    02  PROGRAM-CONSTANTS.                                    00001460
                                                                00001470
    EJECT                                                    00001480
                                                                00001490
*-----* 00001500
*              PROGRAM COPY BOOKS                          * 00001510
*-----* 00001520
01  PROGRAM-COPY-AREAS.                                    00001530
*  PLACE RECORD LAYOUTS AND COPY STATEMENTS HERE          00001540
                                                                00001550
    02  WS-AMIGOS-RECORD          PIC X(55).                00001560
                                                                00001570

```

SAMPLE COBOL (4/7)

01	WORK-STORAGE-END	PIC X(38) USAGE IS DISPLAY	00001580
	VALUE 'WORKING STORAGE FOR [MODNAME ENDS HERE'].		00001590
*			00001600
			00001610
	EJECT		00001620
			00001630
			00001640
-----			00001650
*	LINKAGE SECTION		00001660
-----			00001670
			00001680
*	DELETE THIS SECTION IF NOT APPLICABLE		00001690
	LINKAGE SECTION.		00001700
01	EXEC-PARAMETER.		00001710
05	PARM-LENGTH	PIC S9(4) COMP.	00001720
05	PARM-FORMAT.		00001730
10	PARM-DATE.		00001740
15	LK-PARM-CC	PIC X(2).	00001750
15	LK-PARM-YY	PIC X(2).	00001760
15	LK-PARM-MM	PIC X(2).	00001770
15	LK-PARM-DD	PIC X(2).	00001780
10	PARM-FILLER	PIC X(92).	00001790
	EJECT		00001800
*****			00001810
*	P R O C E D U R E D I V I S I O N		00001820
*****			00001830
			00001840
	PROCEDURE DIVISION.		00001850
*	IF THE LINKAGE SECTION IS NEEDED, USE THE FOLLOWING STATEMENT:		00001860
*	PROCEDURE DIVISION USING EXEC-PARAMETER.		00001870
			00001880
	0000-MAIN-DRIVER.		00001890
			00001900
	PERFORM 1000-INIT THRU 1000-EXIT.		00001910
	PERFORM 2000-PROCESS THRU 2000-EXIT		00001920
	UNTIL RETURN-CODE-IS-BAD OR END-OF-AMIGOS.		00001930
	PERFORM 3000-EOF THRU 3000-EXIT.		00001940
	STOP RUN.		00001950
			00001960
	EJECT		00001970
			00001980
-----			00001990
*	INITIALIZATION ROUTINE		00002000
-----			00002010
			00002020
	1000-INIT.		00002030
			00002040
*	PERFORM ANY INITIAL ROUTINES FOR TABLES, DATA ITEMS, ETC...		00002050
	PERFORM 1100-OPEN-FILES THRU 1100-EXIT.		00002060
	PERFORM 3400-AMIGOS-VERIFY THRU 3400-EXIT.		00002070
	PERFORM 1200-POSB-AMIGOS THRU 1200-EXIT.		00002080
	PERFORM 3400-AMIGOS-VERIFY THRU 3400-EXIT.		00002090
	PERFORM 2300-GET-AMIGOS THRU 2300-EXIT.		00002100

SAMPLE COBOL (5/7)

PERFORM 3400-AMIGOS-VERIFY THRU 3400-EXIT.	00002110
	00002120
1000-EXIT. EXIT.	00002130
	00002140
	00002150
-----	00002160
* OPEN QSAM FILES *	00002170
-----	00002180
	00002190
1100-OPEN-FILES.	00002200
* OPEN ALL FILES NEEDED HERE. IF VARIED ENOUGH, YOU COULD	00002210
* SEPARATE OUT QSAM OPENS FROM AMIGOS OPENS. AMIGOS-FILE	00002220
* NEEDED ONLY IF MULTIPLE AMIGOS FILES BEING ACCESSED.	00002230
OPEN OUTPUT PRINT-FILE.	00002240
MOVE 'OPENA' TO AMIGOS-COMMAND.	00002250
MOVE 'ERICH' TO AMIGOS-FILE.	00002260
CALL 'OPENA' USING AMIGOS-TAG, 1100-EXIT.	00002270
	00002280
1100-EXIT. EXIT.	00002290
	00002300
	00002310
-----	00002320
* POSITION AMIGOS FILE TO BEGINING *	00002330
-----	00002340
	00002350
1200-POSB-AMIGOS.	00002360
	00002370
MOVE 'POSB ' TO AMIGOS-COMMAND	00002380
CALL 'POSB' USING AMIGOS-TAG, 1200-EXIT.	00002390
	00002400
1200-EXIT. EXIT.	00002410
EJECT	00002420
	00002430
-----	00002440
* PROCESS ROUTINE *	00002450
-----	00002460
	00002470
2000-PROCESS.	00002480
* CODE PROCESSING ROUTINES HERE EXAMPLES LISTED BELOW. *	00002490
* PERFORM 2100-PREP-REC THRU 2100-EXIT. *	00002500
* PERFORM 2200-PRINT-REC THRU 2200-EXIT. *	00002510
	00002520
PERFORM 2300-GET-AMIGOS THRU 2300-EXIT.	00002530
PERFORM 3400-AMIGOS-VERIFY THRU 3400-EXIT.	00002540
	00002550
2000-EXIT. EXIT.	00002560
	00002570
	00002580
-----	00002590
* GET AMIGOS RECORD *	00002600
-----	00002610
	00002620
2300-GET-AMIGOS.	00002630

SAMPLE COBOL (6/7)

MOVE 'GET ' TO AMIGOS-COMMAND	00002640
CALL 'GET' USING AMIGOS-TAG, WS-AMIGOS-RECORD, 2300-EXIT.	00002650
	00002660
	00002670
 2300-EXIT. EXIT.	 00002680
	00002690
EJECT	00002700
	00002710
-----	00002720
* END OF FILE PROCESSING ROUTINE *	00002730
-----	00002740
	00002750
3000-EOF.	00002760
* PERFORM DEINITIALIZE ROUTINES HERE EXAMPLES LISTED BELOW	00002770
* PERFORM 3100-TOTAL-RTN THRU 3100-EXIT.	00002780
* PERFORM 3200-AVERAGE-RTN THRU 3200-EXIT.	00002790
	00002800
PERFORM 3300-CLOSE-FILES THRU 3300-EXIT.	00002810
PERFORM 3400-AMIGOS-VERIFY THRU 3400-EXIT.	00002820
	00002830
3000-EXIT. EXIT.	00002840
	00002850
-----	00002860
* CLOSE FILES *	00002870
-----	00002880
	00002890
3300-CLOSE-FILES.	00002900
	00002910
CLOSE PRINT-FILE.	00002920
IF AMIGOS-COMMAND = 'OPEN ' AND	00002930
AMIGOS-RETURN-CODE NOT = 0	00002940
NEXT SENTENCE	00002950
ELSE,	00002960
MOVE 'CLOSE' TO AMIGOS-COMMAND	00002970
CALL 'CLOSE' USING AMIGOS-TAG, 3300-EXIT.	00002980
3300-EXIT. EXIT.	00002990
	00003000
	00003010
-----	00003020
* CHECK/VERIFY THE AMIGOS RETURN CODES *	00003030
-----	00003040
	00003050
3400-AMIGOS-VERIFY.	00003060
* YOU COULD SET UP THE RETURN CODE EDITS IN THE MODULE AFTER THE	00003070
* PERFORM OF THE AMIGOS CALL STATEMENT (IF NOT VALID GO TO AMIGOS	00003080
* VERIFY ROUTINES).	00003090
	00003100
MOVE AMIGOS-ERROR-CODE TO AMIGOS-RETURN-CODE.	00003110
IF RETURN-CODE-IS-GOOD	00003120
IF AMIGOS-COMMAND = 'GET '	00003130
ADD 1 TO AMIGOS-RECORD-COUNTER	00003140
ELSE,	00003150

SAMPLE COBOL (7/7)

NEXT SENTENCE	00003160
ELSE,	00003170
IF RETURN-CODE-IS-BAD	00003180
DISPLAY 'AMIGOS COMMAND: ' AMIGOS-COMMAND UPON SYSOUT	00003190
DISPLAY 'AMIGOS RETURN CODE: ' AMIGOS-RETURN-CODE UPON	00003200
SYSOUT	00003210
MOVE 4 TO RETURN-CODE	00003220
ELSE,	00003230
NEXT SENTENCE.	00003240
	00003250
3400-EXIT. EXIT.	00003260
EJECT	00003270
	00003280

SAMPLE JCL

```

//FCS8L143 JOB (PNB9,XXXX), $FINDESK, CLASS=P, REGION=4M          00000001
//*                                                                00000002
//JOB LIB DD DISP=(SHR,PASS), DSN=SYS9.OBJECT      NEED CLASS=P  00000003
// DD DISP=(SHR,PASS), DSN=SYS1.COB2LIB           00000004
// DD DISP=(SHR,PASS), DSN=SYS5.CAI.CAILIB        00000005
// DD DISP=(SHR,PASS), DSN=SYS5.CAI.CAILPA        00000006
// DD DISP=(SHR,PASS), DSN=SYNC.LINKLIB           00000007
// DD DISP=(SHR,PASS), DSN=SYS9.SUBRTN.LIBRARY     00000008
//*                                                                00000009
//GETCASM EXEC SYS0X13, PARM=A, HOLD=YES          PARM=P TO PRINT 00000010
//*                                                                00000011
//*****                                                                00000012
//***** GET MEMBERSHIP RECORDS FROM CAS BCOFSB ***** 00000013
//*****                                                                00000014
//FILEIN DD DISP=(SHR,PASS), DSN=PNB9.VSAM.BCOFSB 00000015
//FILEOUT DD DSN=&&CASMEMB, DISP=(NEW,PASS),        00000016
// DCB=(RECFM=FB, LRECL=810, BLKSIZE=8100),      00000017
// UNIT=SYSDA, SPACE=(CYL, (18,3))                00000018
//SYSIN DD DISP=(SHR,PASS), DSN=SYS9.CONTROL(FCS8L143) 00000019
//OUTREC DD DISP=(SHR,PASS), DSN=SYS9.CONTROL(CASMEM02) 00000020
//*                                                                00000021
//*                                                                00000022
//FCS8L143 EXEC FCS8L143 R007879=UN00?? UNLABELED CARTRIDGE # AVAIL. 00000023
//*                                                                00000024
//*----- 00000025
//* THE FOLLOWING IS TO CHECK WHETHER CERTAIN MEMBER IS ON THE TAPE 00000026
//* AND IT WILL ALSO GIVE THE RECORD COUNT. 00000027
//*----- 00000028
//SYS0X13 EXEC SYS0X13, PARM=P, HOLD=YES          00000029
//FILEIN DD DISP=OLD, DSN=T.PNB9.DALLAS.MEDCO     00000030
//*ILEIN DD DISP=OLD, DSN=&&MEMBCS                  00000031
//SYSIN DD *                                        00000032
0008 09 454580696 OK TO CHECK 9 DIGITS EQC 00000033
0008 09 455660593 OK TO CHECK 9 DIGITS EQC 00000034
0008 09 466778631 OK TO CHECK 9 DIGITS EQC 00000035
//                                                                00000036

```