Z77-6358

TECHNICAL INFORMATION EXCHANGE

IBM®

Sept. 30, 1966

OPERATING SYSTEM/360 CONVERSION AND INSTALLATION INFORMATION FOR PROGRAMMERS

Miss Sylvia S. Murphey
IBM Corporation
1439 Peachtree Street, N. E.
Atlanta, Georgia 30309

This paper centers around an example program intended to demonstrate what can be included in a program for the Operating System. Covered in detail is a description of register conventions, including SAVE and RETURN macros; description of data sets, including Data Control Blocks and Data Definition statements; operator communication, including Write to Operator with Reply (WTOR), Write to Operator (WTO), and usage of the PARAM entry in the EXEC job control card; and a description of control characters. There is also a section on documentation covering program documentation and a scheme for naming jobs, job steps, Data Definition statements, Data Control Blocks, data sets and programs. A helpful list of topics is provided with reference to specific O/S manuals and page numbers. Although this paper is focused on the Queued Sequential Access Method (QSAM), most information can be applied to other access methods.

For IBM Internal Use Only

IBM Corporation, Technical Publications Dept., 112 E. Post Road, White Plains, N.Y. 10601

TABLE OF CONTENTS

INTRODUCTION

S/360 Operating System offers the user a myriad of facilities and options. The system creates a new environment for data processing in that one of its main objectives is to maintain a constant work flow through the computing system. Keywords, such as turn-around time and throughput, are given new significance and meaning by minimizing setup time and computer time lost on job-to-job transition. The concept of a JOB is expanded from being one program only to a broader idea of a unit of work to be done encompassing many programs and several job steps.

However, with enlarged capability comes complexity. Most novices in the Operating System find that the volume of information written about the system is quite large. The evergrowing and constantly changing storehouse of information can be quite overpowering. Systems Engineers and IBM customers must somehow sift out of all of the information available what they must know in order to begin the programming effort. It must be emphasized that there is NO substitute for reading the manuals. However, in an attempt to aid programmers get some idea of what is necessary to write a program for the Operating System, a skeleton example program may be very helpful. Often a concrete example helps the programmer to obtain a firm grasp on some of the things that he needs to know. An example may show many details which the programmer may otherwise overlook. If he sees something used, he may then return to his manuals to pursue the write-up of the macro or instruction in depth.

Documentation and program organization become even more important as the complexity of a program grows. Therefore, included in the example program is an illustration of one way a program may be organized. We have found that the important thing with documentation is that whatever may be agreed upon should be strictly enforced if it is to be effective.

The program was written for an installation which planned initially to use in most of its programs the Primary Control Program, the Queued Sequential Access Method (QSAM), the move mode of GET and PUT, and Assembler Language. These four limitations narrow the focus of this paper.

A. PREREQUISITES

1. A basic understanding of Assembler Language.

2. Familiarity with Introduction to O/S, C28-6534, and Concepts and Facilities, C28-6535.

B. SOURCE MANUALS

Introduction	C28-6534	(INTRO)
Concepts & Facilities	C28-6535	(CF)
Job Control Language	C28-6539	(JCL)
Control Program Services	C28-6541	(CPS)
Data Management	C28-6537	(DM)
Linkage Editor	C28-6538	(LE)

C. TOPIC REFERENCES

Topics are listed in order of presentation in paper.

Topic	Page	Manual
WTOR	112	(CPS)
Event Control Block	91	(CPS)
WTO	111	(CPS)
PARAM in EXEC Card	36	(CPS)
	15	(JCL)
Control Characters ASA	333	(CPS)
Machine Code	29	IBM 2821 Control Unit A24-3312
	12	S/360 Reference Data Card X20-1703-3

II. PROGRAM EXAMPLE

A. PROGRAM INTRODUCTION

The following program example contains:

1. Comment cards used to organize the program into basic sections
   a. Housekeeping
   b. Main Line
   c. Sub-routines
   d. Special Routines
   e. Constants, Accumulators, and Working Storage
   f. Input and Output Areas
   g. Data Control Blocks
2. The example is designed to use the Queued Access Method with the Move Mode of GET and PUT.
3. There are three input files and four output files. Input from tape, card, and disk--Output to tape, card, disk, and printer.
4. Examples of WTOR, GET, PUT are given.

```
1  NAME  10 OP 16  OPERAND                                          72
PGM      TITLE 'GENERAL PROGRAM OUTLINE FOR O/S'
PROG1    START
*              *** HOUSEKEEPING ***
START    SAVE  (14,12),,PROG1       SAVE REGISTERS 14,15,0-12
         BALR  12,0
         USING *,12
         USING *+4096,11
         USING *+8192,10
ASTRSK   L     11,ADCON1
         L     10,ADCON2
         ST    13,SAVE+4            STORE ADDR OF CALLING PROG'S SAVE AREA
*                                   IN YOUR SAVE AREA
         LA    5,SAVE               STORE ADDR OF YOUR SAVE AREA IN
         ST    5,8(0,13)             CALLING PROG'S SAVE AREA
         B     BEGIN
ADCON1   DC    A(ASTRSK+4096)
ADCON2   DC    A(ASTRSK+8192)
         EJECT
*              *** MAIN LINE PROGRAM ***
BEGIN    OPEN  (TAPEIN,(INPUT))           OPEN ALL DCB'S
         OPEN  (DISKIN,(INPUT))
         OPEN  (CARDIN,(INPUT))
         OPEN  (TAPEOUT,(OUTPUT))
         OPEN  (DISKOUT,(OUTPUT))
         OPEN  (CARDOUT,(OUTPUT))
         OPEN  (PRINTER,(OUTPUT))
      INSTRUCTIONS
      INSTRUCTIONS
         BAL   3,GETCARD            BRANCH AND LINK TO COMMON GET ROUTINES
         BAL   3,GETDISK
         BAL   3,GETTP
      INSTRUCTIONS
      INSTRUCTIONS
*                                   ISSUE MESSAGE TO OPERATOR VIA CONSOLE
         NI    ECB1,X'BF'
MSG1     WTOR  'GIVE CURRENT PROCESSING DATE',CURDT,6,ECB1
         WAIT  ECB=ECB1
*                                   ANSWER IS PLACED IN STORAGE CURDT
MSG2A    NI    ECB1,X'BF'
MSG2     WTOR  'IS THIS A WEEKLY RUN',ANS,3,ECB1
         WAIT  ECB=ECB1
         CLC   ANS,NOT              DEPENDING ON REPLY A BRANCH
         BE    DAILY                   IS TAKEN
         CLC   ANS,YES
         BE    WEEKLY
         B     MSG2A                REISSUE MESSAGE IF WRONG REPLY
      INSTRUCTIONS
      INSTRUCTIONS
         BAL   3,HEADRTN
         BAL   3,PUTTP              BRANCH AND LINK TO COMMON PUT
         BAL   3,PUTDSK
         BAL   3,PUTCD
         BAL   3,WRITEPRT
         CLOSE (TAPEIN)             CLOSE ALL DCB'S
```



```
1  NAME  10 OP 16  OPERAND                                          72
         CLOSE (DISKIN)
         CLOSE (CARDIN)
         CLOSE (TAPEOUT)
         CLOSE (DISKOUT)
         CLOSE (CARDOUT)
         CLOSE (PRINTER)
         L     13,SAVE+4            LOAD ADDR OF YOUR SAVE AREA IN REG 13
         RETURN (14,12)             RESTORE SAVED REGISTERS
         EJECT                      END OF MAIN LINE OF PROGRAM
*              *** SUBROUTINES ***
         SPACE 2
GETCARD  GET   CARDIN,WRKREAD        GET MOVE PLACES A LOGICAL RECORD
         BR    3                     IN SPECIFIED WORK AREA
GETTP    GET   TAPEIN,WRKTPIN
         BR    3
GETDSK   GET   DISKIN,WRKDSKIN
         BR    3
PUTTP    PUT   TAPEOUT,WRKTPOUT      ALL PUT ROUTINES ARE PUT MOVE
         BR    3                     DATA MUST HAVE BEEN MOVED VIA
PUTDSK   PUT   DISKOUT,WRKDSKOU      PROGRAMMING TO THE OUTPUT WORK AREA
         BR    3
PUTCD    PUT   CARDOUT,WRKPUNCH
         BR    3
WRITEPRT CP    COUNT,MAX             TEST FOR NUMBER OF LINES PRINTED
         BE    SKIP
         MVI   CNTRL,X'40'           CARRIAGE CONTROL FOR SPACE ONE LINE
*                                     BEFORE PRINTING - SEE P.333 GPS
         AP    COUNT,ONE             INCREMENT LINE COUNTER BY ONE
         BAL   5,PUTPRT
         BR    3
SKIP     MVI   CNTRL,X'40'
         ZAP   COUNT,CLEAR           CLEAR COUNTER
         BAL   5,PUTPRT
         B     HEADRTN
PUTPRT   PUT   PRINTER,WRKPRT
         MVC   WRKPRT,CLEARPRT       CLEAR PRINT AREA
         BR    5
HEADRTN  MVI   CNTRL,X'F1'           SKIP TO CHANNEL ONE
         MVC   WRKPRT+32(7),HEADING  MOVE HEADING TO WORK AREA
         BAL   5,PUTPRT
         BR    3
         EJECT
*              *** SPECIAL ROUTINES ***
ERROR    WTO   'AN INPUT OUTPUT ERROR HAS OCCURED - JOB ENDED'
         BR    14
*              *** CONSTANTS-ACCUMULATORS-WORKING STORAGE ***
         SPACE 2
*               ** HALF WORD ALIGNMENT **
         DS    0H
*               ** FULL WORD ALIGNMENT **
         DS    0F
SAVE     DS    18F
ECB1     DC    F'0'
*               ** DOUBLE WORD ALIGNMENT **
         DS    0D
         SPACE 2
```

```
*                ** NO ALIGNMENT **
*                 * CONSTANTS *
MAX       DC     X'050C'         MAXIMUM NUMBER OF LINES PER PAGE
CLEAR     DC     X'0C'
ANS       DS     CL3             WEEKLY OR DAILY RUN
CURDT     DS     CL6             CURRENT DATE
*                 * COUNTERS *
COUNT     DC     PL2'0'          LINE COUNTER
ONE       DC     P'1'
*                 * SWITCHES *
SWITCH    DC     X'00'               ONE BYTE CAN REPRESENT MANY SWITCHES
*                 * HEADINGS *
HEADING   DC     C'HEADING'
*                 * EDIT WORDS *
*                 * ACCUMULATORS *
*                   INVOICE       DESCRIPTIVE COMMENTS
          DC STATEMENTS             DEFINE ACCUMULATORS AS ZERO
          DC STATEMENTS              CONSTANTS WITH A GOOD SIGN
*                   FILE
          DC STATEMENTS
          DC STATEMENTS
*                 * WORKING STORAGE *
          DS STATEMENTS
          SPACE 2
*                ** LITERALS **
          LTORG
          EJECT
*                *** INPUT/OUTPUT WORK AREAS ***
          SPACE 2
*                 ** INPUT WORK AREAS **
WRKREAD   DS     0CL80
CDFLD1    DS     CL20
CDFLD2    DS     CL40
CDFLD3    DS     CL20
          SPACE 2
WRKTPIN   DS     0CL50
TPFLD1    DS     CL25
TPFLD2    DS     CL25
          SPACE 2
WRKDSKIN  DS     0CL50
DKFLD1    DS     CL20
DKFLD2    DS     CL20
DKFLD3    DS     CL10
          SPACE 2
*                 ** OUTPUT WORK AREAS **
WRKPUNCH  DS     0CL80
PUFLD1    DS     CL20
PUFLD2    DS     CL40
PUFLD3    DS     CL20
          SPACE 2
WRKTPOUT  DS     0CL50
TPFLD1A   DS     CL25
TPFLD2A   DS     CL25
          SPACE 2
WRKDSKOU  DS     0CL50
DKFLD1A   DS     CL25
```



```
DKFLD1A   DS     CL25
DKFLD2A   DS     CL25
          SPACE 2
CLEARPRT  DC     C' '
WRKPRT    DS     0CL133
CNTRL     DS     CL1
PRT       DS     CL132
          EJECT
*                 *** DATA CONTROL BLOCKS ***
          SPACE 2
*                 ** INPUT DCBS **
TAPEIN    DCB    DSORG=PS,MACRF=GM,DDNAME=TAPEIN,                      X
               EODAD=ENDTAPE,SYNAD=ERROR
          SPACE 2
DISKIN    DCB    DSORG=PS,MACRF=GM,DDNAME=DISKIN,                      X
               EODAD=ENDDISK,SYNAD=ERROR
          SPACE 2
CARDIN    DCB    DSORG=PS,MACRF=GM,DDNAME=CARDIN,                      X
               EODAD=ENDCARD,SYNAD=ERROR,EROPT=SKP,                  X
               RECFM=FBS,BLKSIZE=80,LRECL=80,BFTEK=S,                X
               BUFNO=2,BUFL=80,BFALN=F
          SPACE 2
*                 ** OUTPUT DCBS **
TAPEOUT   DCB    DSORG=PS,MACRF=PM,DDNAME=TAPEOUT,SYNAD=ERROR
          SPACE 2
DISKOUT   DCB    DSORG=PS,MACRF=PM,DDNAME=DISKOUT,SYNAD=ERROR
          SPACE 2
CARDOUT   DCB    DSORG=PS,MACRF=PM,DDNAME=CARDOUT,SYNAD=ERROR
          SPACE 2
PRINTER   DCB    DSORG=PS,MACRF=PM,DDNAME=PRINTER,BFTEK=S,             X
               BUFNO=1,BUFL=133,BFALN=F,EROPT=ACC
          END    START
```

III. DESCRIPTION OF DATA SETS

The two sources of information used to describe the data sets in the example program are the Data Definition (DD) statement and the Data Control Block (DCB). The following are provided.

Data Definition Statements

A. The DD cards as they might be coded.

B. An example of the DD parameters chosen.

Data Control Blocks

A. The DCB's themselves are shown in the example program.

B. A chart is given of all of the DCB parameters showing

1. What each entry is in the example program.

2. Which entries were actually put in the DCB and which ones were put in the DD card.

A. DATA DEFINITION STATEMENTS

1. INTRODUCTION

Of the various parameters available in the DD statement, some fall into the category of being "necessary to make the job run." As an introduction to Operating System coding, a good approach was to concern the programmer in depth, at least initially, with only those options which he must include. This by no means indicates that the many other options are not useful, or, as the programmers progress, necessary to obtain optimum efficiency. However, as a basic introduction, in keeping with the effort to give the programmer a feel for what can be included in a DD statement, the following DD statements were given as examples of what is needed to complete the description of the data sets used in the example program.

2. DATA DEFINITION CONTROL CARDS

1. Output Data Sets

A. Disk

1. Data set is catalogued

```
//DISKOUT DD  DSNAME=MASTER(+1),DCB=(,EROPT=SKP,RECFM=FBS,          X
//           BLKSIZE=250,LRECL=50,BFTEK=S,BUFNO=2,BUFL=250,        X
//           BFALN=F),SPACE=(TRK,(50,10),RLSE),                    X
//           VOLUME=REF=*.DISKIN,DISP=(,CATLG)
```

2. Data set is passed

```
//DISKOUT DD  DSNAME=MASTER,DCB=(,EROPT=SKP,RECFM=FBS,             X
//           BLKSIZE=250,LRECL=50,BFTEK=S,BUFNO=2,BUFL=250,        X
//           BFALN=F),SPACE=(TRK,(50,10),RLSE),                    X
//           VOLUME=REF=*.DISKIN,DISP=(,PASS)
```

B. Tape

```
//TAPEOUT DD  DSNAME=DETAIL(+1),DCB=(,EROPT=SKP,DEN=2,             X
//           RECFM=FBS,BLKSIZE=250,LRECL=50,BFTEK=S,               X
//           BUFNO=2,BUFL=250,BFALN=F),UNIT=TAPE,                  X
//           LABEL=(,SL,RETPD 0004),DISP=(,CATLG)
```

C. Card

```
//CARDOUT DD  DSNAME=CARD,DCB=(,EROPT=SKP,RECFM=FBS,               X
//           BLKSIZE=80,LRECL=80,BFTEK=S,BUFNO=2,BUFL=80           X
//           BFALN=F),UNIT=PUNCH
```

D. Printer

```
//PRINTER DD   DSNAME=REPORT,DCB=(,RECFM=FSA,BLKSIZE=133,          X
//            LRECL=133),SYSOUT=A
```

II. Input Data Sets

A. Disk

```
//DISKIN DD    DSNAME=MASTER(0),DCB=(,EROPT=SKP,RECFM=FBS,       X
//            BLKSIZE=250,LRECL=50,BFTEK=S,BUFNO=2,BUFL=250,    X
//            BFALN=F),UNIT=DISK,DISP=(OLD,CATLG)
```

B. Tape

1. Data set is catalogued

```
//TAPEIN DD    DSNAME=DETAIL(0),DCB=(EROPT=SKP,DEN=2,            X
//            RECFM=FBS,BLKSIZE=250,LRECL=50,BFTEK=S,BUFNO=2,   X
//            BUFL=250,BFALN=F),UNIT=TAPE,DISP=(OLD,CATLG)
```

2. Data set is not catalogued

```
//TAPEIN DD    DSNAME=DETAIL,DCB=(,EROPT=SKP,DEN=2,RECFM=FBS,    X
//            BLKSIZE=250,LRECL=50,BFTEK=S,BUFNO=2,BUFL=250,    X
//            BFALN=F),UNIT=TAPE,DISP=(OLD,CATLG),              X
//            VOLUME=SER=123456
```

C. Card

```
//CARDIN DD    *
                CARD DATA HERE
/*
```


3. EXPLANATION

I. Output Data Sets

A. DISKOUT

When writing a new data set on disk, as done in the example program, the SPACE parameter is included. In our example, we reserved 50 tracks initially, specified that if there was insufficient space for the data set on these 50 tracks, space was to be allocated in increments of 10 tracks each. At the end of this step, if all of the space allocated was not used, the unused tracks were to be released (RLSE) for use by other data sets.

Also specified was the request that the output MASTER(+1) data set be placed on the same physical unit as MASTER(0) defined in the DD statement DISKIN in this same job step. VOLUME=REF=*.DISKIN

The disk output data set is to be made a new member of its data set. Therefore, the disk data set name is MASTER(+1). At the completion of this job step, it is to be catalogued. Since it is new, the first parameter of the DISP (disposition) does not have to be specified since NEW is assumed by default. When this data set is catalogued, it is automatically made the most current generation or 'son'. Accordingly, its element is changed from (+1) to (0) at disposition time and will be the input data set the next time the job step is run. At this time its serial number is recorded in the catalogue along with its element.

An additional DD statement is included for DISKOUT showing the parameters required if this data set is not to be catalogued but instead passed to the next job step where more processing can be done and then the disposition specified. An example of where this method might be used is a job in which the first job step creates a file on disk, passes it to the next job step, and then this file is printed, possibly with some additional processing.

B. TAPEOUT

The tape output data set is also catalogued as described under the explanation of DISKOUT. DETAIL(+1) is to have standard labels and a retention period of 4 days.

C. CARDOUT

The parameters required to complete the DCB are shown. No disposition, DISP, is necessary because the data set is new, and it is to be deleted at the end of the job step.

D. PRINTER

For a file that is to be put on the printer, the DD parameter, SYSOUT, is used to specify the standard output class. At the sequential scheduler level, the UNIT parameter must be omitted if SYSOUT is specified.

II. Input Data Sets

A. DISKIN

DISKIN is an OLD data set, for it was previously created. MASTER(0) is the data set name under which the disk data set is now catalogued. MASTER(0) indicates that the input should be the most current generation of the data set. When the disposition is executed at the end of the step, the generation number or 'element' of this data set will become (-1), indicating that it is now the 'father' version of the MASTER data set, the 'son' being the most current version.

At System Generation time the addresses of the disk units were all equated to 'DISK'. Therefore, the DD statement does not have to specify a particular device address, but may specify UNIT=DISK. In this way 'drive independence' is obtained.

We complete the information needed for the DCB by specifying DCB= (parameter list). Note that in neither the DCB nor the DD statement is the DEVD or type of device parameter specified. This omission is made for an important reason. When the DCB is expanded, its length depends upon the type of device specified. If no DEVD parameter is given, the DCB is expanded to a maximum length. This is important when the data set being defined is a printer because a printer DCB expands into a shorter length than does a disk or tape device. Therefore, if a printer happened to be 'down', the printer data set could be temporarily written on disk or tape only if the DCB expansion assembled in the program were large enough to describe a disk or tape data set. Consequently, if you always allow the maximum length of the DCB by omitting the DEVD parameter, you may change the UNIT on the DD statement and be sure that the DCB is large enough to handle the file description.

B. TAPEIN

DETAIL is the data set name of the input tape. This data set is also catalogued. DETAIL is a generation data set and therefore the input data set name is DETAIL(0). It is important to note that the volume serial number does not have to be specified if the data set is catalogued because the serial number is kept by generation number in the catalogue with the data set name. In addition to the DD example of TAPEIN as a catalogued data set, another example is given of DETAIL as an uncatalogued data set. In this case, the programmer would have to call for the data set by serial number.

As with the disk data set, we do not call for a specific unit but say UNIT=TAPE.

The DCB is completed in the DD statement, again omitting the DEVD parameter.

C. CARDIN

This data set is to come in via the input stream. Therefore, an asterisk, *, is the only parameter allowed in the DD statement. The data should immediately follow with a /* card at the end of the card data.

DATA SET CHARACTERISTICS

* Indicates parameters left for DD statement

Name of Data Control Block	I/O Macro Form	Data Set Description: Organization	Name DD	Record Form	Block Size	Length Record	Type of Device	Name of End of Data	I/O Errors: Error Routine	Error Option	Buffers: Tech.	No.	Length	Alignment
DCB	MACRF	DSORG	DDNAME	RECFM	BLKSIZE	LRECL	DEVD	EODAD	SYNAD	EROPT	BFTEK	BUFNO	BUFL	BUFALN
TAPEIN	GM	PS	TAPEIN	FBS *	250 *	50 *	DEN=2 *	ENDTAPE	ERROR	SKP *	S *	2 *	250 *	F *
DISKIN	GM	PS	DISKIN	FBS *	250 *	50 *	omit	ENDDISK	ERROR	SKP *	S *	2 *	250 *	F *
CARDIN	GM	PS	CARDIN	FBS	80	80	omit	ENDCARD	ERROR	SKP	S	2	80	F
TAPEOUT	PM	PS	TAPEOUT	FBS *	250 *	50 *	DEN=2 *	omit	ERROR	SKP *	S *	2 *	250 *	F *
DISKOUT	PM	PS	DISKOUT	FBS *	250 *	50 *	omit	omit	ERROR	SKP *	S *	2 *	250 *	F *
CARDOUT	PM	PS	CARDOUT	FBS *	80 *	80 *	omit	omit	ERROR	SKP *	S *	2 *	80 *	F *
PRINTER	PM	PS	PRINTER	FSA *	133 *	133 *	omit	omit	omit	ACC	S	1	133	F

IV. REGISTER CONVENTIONS

A. EXPANSION OF SAVE AND RETURN MACROS

+ Indicates Expansion

(PROG 1)

```
     SAVE   (14,12), ID
+    DS     OH
+    STM    14,12,12(13)
```

The calling program must load register 13 with the address of its save area. Therefore, when your program (the called program) issues the SAVE macro, you are storing the calling program's registers in the calling programs save area. Note that the store multiple instruction uses register 13 as a base register with a displacement of 12. One register needs 4 bytes of storage.

Calling program's save area: 1 word = 4 bytes

SAVEAREA DS 18F SAVEAREA is 18 full words

WORD 1	WORD 2	WORD 3 Addr.Called Prog's Savearea	WORD 4 Register 14	WORD 5 Register 15	WORD 6 Register 0
WORD 7 Register 1	WORD 8 Register 2	WORD 9 Register 3	WORD 10 Register 4	WORD 11 Register 5	WORD 12 Register 6
WORD 13 Register 7	WORD 14 Register 8	WORD 15 Register 9	WORD 16 Register 10	WORD 17 Register 11	WORD 18 Register 12

The called program issues the following instructions:

```
     LA     5,SAVE
     ST     5,8(13)
```

Save is the address of the called program's save area. This address is placed in the third word of the calling program's save area. Note that register 13 still has the address of the calling program's save area and that register 13 is used as a base register in the store instruction above.

The called program issues the following instruction:

```
     ST     13,SAVE+4
```


This instruction places the contents of register 13 in the second word of its (the called program's) save area. This is necessary because the called program must reload register 13 with this address before it issues a return.

```
     L      13,SAVE+4
     RETURN (14,12)
+    LM     14,12,12(13) restore the registers
+    BR     14
```

The expansion of the RETURN macro indicates clearly why the address of register 13, containing the address of the calling program's save area, must be stored and then reloaded. As with the SAVE macro, register 13 is used as a base register. Note that the return branch is on the address in register 14 which had been loaded by the calling program before the called program was given control initially.

Note that if the calling program is not a program as we may normally think of one, but is instead the control program, these conventions of linkage must still be observed.

B. REGISTER RESTRICTIONS

There are five registers which should not be used by the problem program. They are registers 0, 1, 13, 14, 15.

V. COMMUNICATION WITH THE OPERATOR

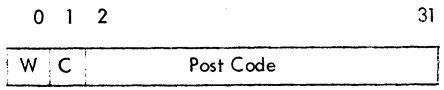
The three ways selected for communication with the operator are via the two macros, WTOR (Write to Operator with Reply) and WTO (Write to Operator), and the PARAM entry in the EXEC job control card.

A. WRITE TO OPERATOR WITH REPLY

```
         NI      ECB1,X'BF'
MSG1     WTOR    'GIVE CURRENT PROCESSING DATE',CURDT,6,ECB1
         WAIT    ECB=ECB1
CURDT    DS      CL6
ECB1     DC      F'0'
```

The WTOR macro must specify:

1. The message to be written.
2. The storage location reserved by the program where the control program is to place the answer.
3. The length of the answer.
4. The name of the Event Control Block (ECB) which the supervisor may use. This ECB must be defined as a full word zero constant. Format of the Event Control Block:



After the WTOR is issued, the programmer must issue a WAIT, if his program logic depends upon the reply. When the WAIT is issued, the supervisor sets bit zero of the ECB specified to 1. When the action has occurred, the supervisor issues a POST which turns bit zero of the ECB, the completion code to 1. The problem program is then given control. It is the programmer's responsibility to be sure that the completion flag is zero before the WTOR is issued again. An "And Immediate',NI , instruction before the WTOR will always insure that the completion flag is zero.

B. WRITE TO OPERATOR

```
MSG2     WTO     'JULY 25 IS THE DATE'
```

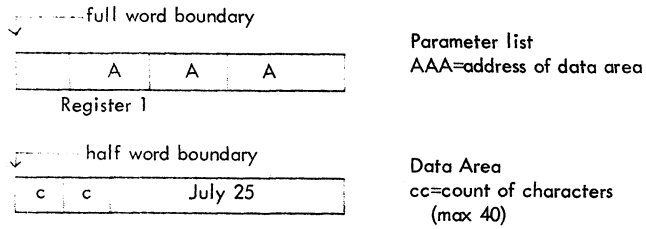
In the expansion of this macro, the message in quotes is found at the address MSG2+8. Therefore, if the programmer wanted to alter the message, he could move into this address the new information. For instance, suppose you had the current date in CURDT. To place this information in the message, you would write as follows:

```
         MVC     MSG2+8(7),CURDT
MSG2     WTO     '       IS THE DATE'
```

C. PARAM

```
PARAM='JULY 25'
```

The above would be written in the EXEC card. The PARAM entry may be up to 40 characters. When the program is given initial control, Register 1 points to the address of the parameter list. The parameter list has the address of the data area where the control program has placed 'JULY 25'.



The programmer would write as follows:

```
         L       5,0(1)
         MVC     CURDT,2(5)
```

Load contents of Reg. 1 into Reg. 5
Reg. 5 now has address of data area

These instructions would place the information put in the PARAM entry into the reserved core location CURDT.

VI. CONTROL CHARACTERS

Instead of using the CNTRL macro and PRTOV macro for direct printer control, control characters may be used. One advantage of this is that it enables the print file to be stored temporarily on disks and then later written on the printer as a SPOOL operation. There are two choices available for control characters.

A. ASA

The example program used ASA control characters as follows:

blank	Space one line before printing
0	Space two lines before printing
-	Space three lines before printing
+	Suppress space before printing
1	Skip to channel 1
2-C	Skip to channel N

When these characters are used the RECFM specified in either the DCB or DD statement must be 'FSA'.

Note that the ASA codes present one drawback in that they do not allow spacing or skipping after print.

B. Machine Codes

To obtain space or skip after print, machine codes should be used.

The RECFM would then be 'FM'.

Hex.	Operation
01	Write, no space
09	Write, space 1 after print
11	Write, space 2 after print
19	Write, space 3 after print
89	Write, skip to channel 1 after print
91	Write, skip to channel 2 after print
99	Write, skip to channel 3 after print
A1	Write, skip to channel 4 after print
A9	Write, skip to channel 5 after print
B1	Write, skip to channel 6 after print
B9	Write, skip to channel 7 after print
C1	Write, skip to channel 8 after print
C9	Write, skip to channel 9 after print
D1	Write, skip to channel 10 after print
D9	Write, skip to channel 11 after print
E1	Write, skip to channel 12 after print

VII. DOCUMENTATION

A. PROGRAM ORGANIZATION

The program example illustrates one suggested way to organize the various sections of a program.

1. A TITLE card is used to
   a. Identify the assembly listing.
   b. Provide identity for the object deck from the name field of the TITLE card.
2. HOUSEKEEPING should contain all of the necessary register set-up including SAVE and base register allocation.
3. The MAIN LINE PROGRAM contains the basic logic flow ending with RETURN.
4. SUBROUTINES may be either closed or open. A closed routine branches on a register. An open routine branches to a specific address. GET and PUT macro instructions are put under subroutines. The GET and PUT are placed here so that, no matter how many different places in the program a given file may be read or written, the macro is expanded only once.
5. SPECIAL ROUTINES may consist of SYNAD routines to handle I/O errors. In the example program ERROR is the SYNAD routine. It is suggested that a common error routine be written for the installation as a whole which can be inserted in each program with little, if any, modification.
6. CONSTANTS, ACCUMULATORS AND WORKING STORAGE is a general division which can be further subdivided to suit the needs of the program. It is suggested that all areas which need special alignment such as full or half word be grouped together and labeled as such.

   Accumulators, as a general rule, should be defined as zero with a good sign. They may be grouped according to the level, be it minor, intermediate or major (invoice, client, file). Each accumulator should be followed by a comment which clearly explains what it is used for even though the name of the accumulator may be neumonic.

   Switches may be grouped together. It is suggested that bit switches be used instead of byte switches in order to conserve core. However, this means that comment cards should explain specifically what each bit represents.

7. INPUT AND OUTPUT WORK AREAS need to be grouped together. Each work area should clearly indicate which DCB it applies to.

8. DCB's are the last division. They are organized by input and output.

The use of EJECT and SPACE instructions to the assembler help to organize the source listing into a more readable format.

B. PROGRAM FOLDER

1. TABLE OF CONTENTS

A program folder should contain all of the information which is needed to describe the program. Suggested contents are:

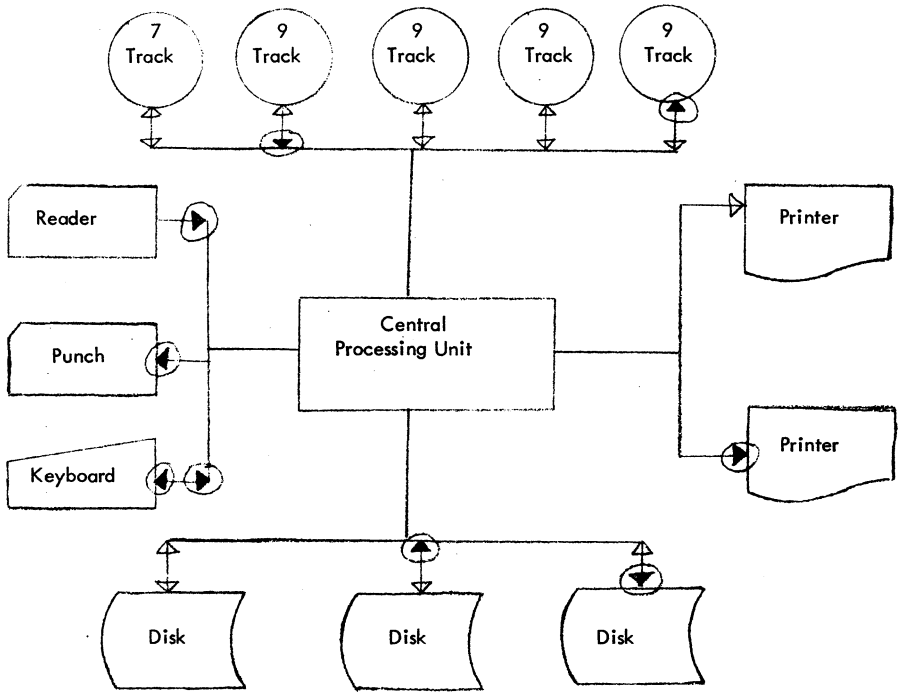
1. Brief program summary
2. Configurator
3. Layouts or formats of input and output records
4. A sample printout
5. A general block diagram
6. The source or assembler listing
7. A copy of the operator's instruction sheet
8. Samples of all job control cards
9. History of changes

The history of changes is a running documentary beginning with the original programmer and date. As a change is made to the program, the name of the programmer who made the change, the date, and a brief synopsis of the change made is entered.

The configurator is a handy way of giving a picture of the input and output units used by the program. A configurator of the sample program is included in this paper.

2. CONFIGURATOR

Circle the appropriate direction and darken the arrow.



C. NAMING CONVENTIONS

The method of naming jobs, programs and data sets varies greatly from installation to installation. With O/S 360 some way of relating jobs to job steps, and programs and of relating DD statements to DCB's is not only helpful, but almost mandatory. Within one installation there are many types of applications programmed for the computer, i.e., PAYROLL, DEMAND DEPOSIT ACCOUNTING, SAVINGS. These applications offer a natural way of organizing programs. For instance, a program written for PAYROLL would begin with the key letters 'PAY'.

There are three types of control cards needed--the JOB card, the EXEC card or job step card, and the DD cards. In the card you need a name with a maximum of 8 characters, which becomes the job name or step name. Since a step is part of some job, it easily follows that the stepname should relate to the jobname. The following is one suggested naming convention for JOB and EXEC cards:

JOB card

Application	Job Number	Step Number
{PAY, DDA, SAV}	1-F	0

Example:	1st job in payroll	PAY10
	2nd job in savings	SAV20
	11th job in demand deposits	DDAB0

EXEC card

Application	Job Number	Step Number
{PAY, DDA, SAV}	1-F	1-F

Example:	1st step in 1st job in payroll	PAY11
	3rd step in 2nd job in savings	SAV23
	15th step in 11th job in demand deposit	DDABF

The name of the DD card must be specified in the DCB parameter, DDNAME. Therefore, to simplify the naming process, it is suggested that the name of the DD statement be the same as the
name of the corresponding DCB in the program.

Programs relate in most cases to one job step. Therefore, their names can relate to the step name as follows:

Application	Job Number	Step Number	neumonic
{PAY, DDA, SAV}	1-F	1-F	

Example: the posting program which is used by the 1st step in the 1st job in payroll

PAY11PST

the dividend program which is used by the 3rd step in the 2nd job in savings

SAV23DIV

the statement program which is used by the 15th step in the 11th job in demand deposit accounting

DDABFSTM

There is no attempt to relate these data set names to a specific job or program since one data set may be used by many different jobs.

For temporary data sets a T prefix on the data set name helps to separate these data sets from those which are permanent.