CONTENTS

TSO Commands

- **Panel-id show?** = PANELID

  Displays the name of the current panel.

- **Message-id** = MSGID

  Displays the Message-id of the last message.

- Automatic **REFLIST** can be accessed by
  - REFLISTD
  - REFLISTL
  - REFACTD
  - REFACTL

  These allow you to pull back datasets without seeing the intermediate lists.

  Commands to quickly add names to your personal lists. These are REFOPEND, REFOPENL, REFADDD, and REFADDL

- **RETP** - View the stack of your typed commands.
- **RETF** - Pull back commands in reverse order.
- **RETRIEVE** - Pop up the most recent typed commands from the stack.
- **CRETRIEV** - Pop up the most recent typed commands from the stack.
- **ISRDDN** - Shows the current data sets allocation for our ISPF session.
- **XMIT**

  The TSO/E Interactive Data Transmission Facility TRANSMIT command allows you to send data sets or messages to persons on other MVS systems via Network Job Entry or directly to persons on your own system.

  XMIT node.userid DA('your.pds') OUTDA('your.ps')

- **RECEIVE**

  The TSO/E Interactive Data Transmission Facility RECEIVE command allows you to obtain files transmitted to your userid. The RECEIVE command queries the Job Entry Subsystem

  RECEIVE INDS('your xmi.ps')

- **DSLIST**
- **SAVE /VDL**

  Save the data set list in log data set

- **QPRINT**

  Save the Quick Reference document in a PDS or PS

- **TSO PROFILE**

The profile command is used either to specify to the system certain user characteristics which are to be used to control the flow of information to and from the terminal or to list the user profile. The user profile is retained throughout each terminal session unless the user wishes to change it by issuing the profile command again and specifying the appropriate operands.

- **CUT DISPLAY**

  This command when given while a member is opened in view/edit mode will show the Clipboard manager. The Clipboard can then be browsed or deleted.

- **KEYLIST**

  Displays the Keylist utility. Invoking **KEYLIST OFF** can disable it.

- **KEYS**
  - If the **KEYLIST** is **ON**, it displays the keys of current Keylist.
  - If the **KEYLIST** is **OFF**, it displays the Primary keys definition.
- **PFSHOW**

  Displays the value of all the PFKEYS on the panel.
  They can be removed from the panel by invoking **PFSHOW OFF**

- **SCRNAME ON**

  Displays the screen name of the Top Left corner. We can give our own screen name to any screen by writing "**SCRNAME myscrname**"

- **SWAP LIST**

  Displays the Active ISPF logical Sessions.

LANGUAGE PROCESSING COMMANDS

- **ASM**

  Invoke assembler prompter and assembler f compiler.

- **CALC**

  Invoke itf:pl/1 processor for desk calculator mode.

- **COBOL**

  Invoke cobol prompter and ans cobol compiler.

- **FORT**

  Invoke fortran prompter and fortran iv g1 compiler.

PROGRAM CONTROL COMMANDS

- **CALL**

  Load and execute the specified load module.

- **LINK**

  Invoke link prompter and linkage editor.

- **LOADGO**

  Load and execute program.

- **RUN**

  Compile, load, and execute program.

- **TEST**

  Test user program.

- **TESTAUTH**

  Test apf authorized programs.

DATA MANAGEMENT COMMANDS

- **ALLOCATE**

  ALLOCATE A DATA SET WITH OR WITHOUT AN ATTRIBUTE LIST OF DCB PARAMETERS.

- **ALTLIB**

  DEFINE OPTIONAL, USER-LEVEL OR APPLICATION-LEVEL SETS OF LIBRARIES CONTAINING SAA/PL EXECS OR CLISTS. THESE LIBRARIES ARE SEARCHED WHEN IMPLICITLY INVOKING AN SAA/PL EXEC OR CLIST.

- **ATTRIB**

  ALLOW DCB PARAMETERS TO BE DYNAMICALLY INTRODUCED AND NAMED FOR USE WITH A SUBSEQUENT ALLOCATE COM.

- **CONVERT**

SIFT ITF/PL1 AND FORTRAN SOURCE.

- **COPY**

  COPY A DATA SET. (SEE NOTE BELOW.)

- **DELETE**

  DELETE A DATA SET.

- **EDIT**

  CREATE, EDIT, AND/OR EXECUTE A DATA SET.

- **FORMAT**

  FORMAT AND PRINT A TEXT DATA SET. (SEE NOTE BELOW.)

- **FREE**

  RELEASE A DATA SET AND/OR AN ATTRIBUTE LIST.

- **LIST**

  DISPLAY A DATA SET. (SEE NOTE BELOW.)

- **LISTALC**

  DISPLAY ACTIVE DATA SETS.

- **LISTBC**

  DISPLAY MESSAGES FROM OPERATOR/USER.

- **LISTCAT**

  DISPLAY USER CATALOGUED DATA SETS.

- **LISTDS**

  DISPLAY DATA SET ATTRIBUTES.

- **MERGE**

  COMBINE DATA SETS. (SEE NOTE BELOW).

- **PRINTDS**

PRINT A DATA SET.

- **PROTECT**

  PASSWORD PROTECT DATA SETS.

- **RENAME**

  RENAME A DATA SET.

- **TSOLIB**

  DEFINE OPTIONAL SEARCH LEVEL THAT TSO/E USES WHEN SEARCHING FOR COMMANDS AND PROGRAMS.

SYSTEM CONTROL COMMANDS

- **ACCOUNT**

  MODIFY/ADD/DELETE USER ATTRIBUTES.

- **CONSOLE**

  PLACE TERMINAL IN CONSOLE MODE.

- **OPERATOR**

  PLACE TERMINAL IN OPERATOR MODE.

- **PARMLIB**

  LIST/UPDATE TSO/E DEFAULTS.

- **RACONVRT**

  UADS TO RACF DATA BASE CONVERSION UTILITY.

- **SYNC**

  SYNCHRONIZE THE BROADCAST DATA SET WITH USER IDS FROM THE UADS AND RACF DATA BASE.

SESSION CONTROL

- **CONSPROF**

DEFINE USER CONSOLE CHARACTERISTICS.

- **EXEC**

  INVOKE COMMAND PROCEDURE.

- **EXECUTIL**

  ALTER REXX ENVIRONMENT.

- **HELP**

  INVOKE HELP PROCESSOR.

- **LOGOFF**

  END TERMINAL SESSION.

- **LOGON**

  START TERMINAL SESSION.

- **PROFILE**

  DEFINE USER CHARACTERISTICS.

- **SEND**

  SEND MESSAGE TO OPERATOR/USER.

- **TERMINAL**

  DEFINE TERMINAL CHARACTERISTICS.

- **TIME**

  LOG SESSION USAGE TIME.

- **TSOEXEC**

  EXECUTE AN AUTHORIZED OR UNAUTHORIZED COMMAND FROM WITHIN AN UNAUTHORIZED ENVIRONMENT.

- **WHEN**

  CONDITIONALLY EXECUTE NEXT COMMAND.

FOREGROUND INITIATED BACKGROUND COMMANDS

- **CANCEL**

CANCEL BACKGROUND JOB.

- **OUTPUT**

  DIRECT OUTPUT MEDIUM FOR BACKGROUND JOB.

- **STATUS**

  LIST STATUS OF BACKGROUND JOB.

- **SUBMIT**

  SUBMIT BACKGROUND JOB.

- **INTERACTIVE**

  DATA TRANSMISSION FACILITY.

- **RECEIVE**

  RECEIVE DATA SENT TO YOU BY A TRANSMIT COMMAND.

- **TRANSMIT**

  TRANSMIT MESSAGES AND/OR DATA SETS VIA NODES DEFINED TO JES

Computer Programming

A Program is a set of instructions that enables a computer to process data, there are two types of computer programs which are 1- Operating system programs, which control the overall operation of the system and 2- Applications programs, it is the actuall program tasks requried by the users. The term used to describe all type of programs is called software in general application program reads input, process it, and produce information or out put the user kneads. A set of computerized business procedures in an application area is called information systems.

Application program development process

1. Determine program specifications.
2. Design the program planning tools.
3. Code and enter the program.
4. Compile the program.
5. Test the program.
6. Document the program.

History of Cobol.

Developed by 1959 by a group called Conference on Data Systems Language (CODASYL). First COBOL compiler was released by December 1959.

First ANSI approved version	1968
Modified ANSI approved version	1974 (OS/VS COBOL)
Modified ANSI approved version	1985 (VS COBOL 2)

This book is written based on IBM COBOL for OS/390 V2R2.

Interdiction to Cobol

COBOL is a high-level programming language first developed by the CODASYL Committee (Conference on Data Systems Languages) in 1960. Since then, responsibility for developing new COBOL standards has been assumed by the American National Standards Institute (ANSI).

The word COBOL is an acronym that stands for Common Business Oriented Language. As the expanded acronym indicates, COBOL is designed for developing business, typically file-oriented, applications. It is not designed for writing systems programs. For instance you would not develop an operating system or a compiler using COBOL

Coding Sheet.

1-6	Page/line numbers - Optional (automatically assigned by compiler)
7	Continuity (-), Comment (*), Starting a new page (/), Debugging lines (D)
8-11	Column A -Division, Section, Paragraph, 01,77 declarations must begin here.
12-72	Column B -All the other declarations/statements begin here.
73-80	Identification field. It will be ignored by the compiler but visible in the source listing.

Language Structure.

Character Set	:-	Digits (0-9), Alphabets (A-Z), Space (b), Special Characters (+ - * / ( ) = $; " > < . ,)
Word	:-	One or more characters- User defined or Reserved.
Clause	:-	One or more words. It specifies an attribute for an entry.
Statement	:-	One or more valid words and clauses.
Sentence	:-	One or more statements terminated by a period.
Paragraph	:-	One or more sentences.
Section	:-	One or more paragraphs.
Division	:-	One or more sections or paragraphs.
Program	:-	Made up of four divisions.

Divisions in COBOL.

There are four divisions in a COBOL program and Data division is optional.

1. Identification Division.
2. Environment Division.
3. Data Division.
4. Procedure Division.

Identification Division.

This is the first division and the program is identified here. Paragraph PROGRAM-ID followed by user-defined name is mandatory. All other paragraphs are optional and used for documentation. The length of user-defined name for IBM COBOL is EIGHT.

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.     PROGRAM NAME.
 AUTHOR.         COMMENT ENTRY.
 INSTALLATION.   COMMENT ENTRY.
 DATE-WRITTEN.   COMMENT ENTRY.
 DATE-COMPILED.  COMMENT ENTRY.
 SECURITY.       COMMENT ENTRY.
```

Security does not pertain to the operating system security, but the information that is passed to the user of the program about the security features of the program.

Environment Division.

Only machine dependant division of COBOL program. It supplies information about the hardware or computer equipment to be used on the program. When your program moves from one computer to another computer, the only section that may need to be changed is ENVIRONMENT division.

Configuration Section.	It supplies information concerning the computer on which the program will be compiled (SOURCE-COMPUTER) and executed (OBJECT-COMPUTER). It consists of three paragraphs - SOURCE COMPUTER, OBJECT-COMPUTER and SPECIAL-NAMES. This is OPTIONAL section from COBOL 85.	
	SOURCE-COMPUTER.	IBM-4381 (Computer and model # supplied by manufacturer) WITH DEBUGGING MODE clause specifies that the debugging lines in the program (statements coded with 'D' in column 7) be compiled.
	OBJECT-COMPUTER.	IBM-4381 (Usually same as source computer)
	SPECIAL-NAMES.	This paragraph is used to relate hardware names to user-specified mnemonic names. 1. Substitute character for currency sign. (CURRENCY SIGN IS litearal-1) 2. Comma can be used as decimal point. (DECIMAL-POINT IS COMMA) 3. Default collating sequence can be changed. It will be explained later . 4. New class can be defined using CLASS keyword. (CLASS DIGIT is "0" thru "9")
Input-Output Section.	It contains information regarding the files to be used in the program and it consists of two paragraphs FILE-CONTROL & I-O CONTROL.	
	FILE CONTROL.	Files used in the program are identified in this paragraph.
	I-O CONTROL.	It specifies when check points to be taken and storage areas that are shared by different files.

Data Division.

Data division is used to define the data that need to be accessed by the program. It has three sections.

FILE SECTION	Describes the record structure of the files.
WORKING-STORAGE SECTION	Is used to for define intermediate variables.
LINKAGE SECTION	Is used to access the external data. Ex: Data passed from other programs or from PARM of JCL.

Literals, Constants, Identifier,

1. Literal is a constant and it can be numeric or non-numeric.
2. Numeric literal can hold 18 digits and non-numeric literal can hold 160 characters in it. (COBOL74 supports 120 characters only)
3. Literal stored in a named memory location is called as variable or identifier.
4. Figurative Constant is a COBOL reserved word representing frequently used constants. They are ZERO/ZEROS/ZEROES, QUOTE/QUOTES, SPACE/SPACES, ALL, HIGH-VALUE/HIGH-VALUES, LOW-VALUE/LOW-VALUES.

Example: 01 WS-VAR1 PIC X (04) VALUE 'MUSA'.
'MUSA ' is a non-numeric literal. WS-VAR1 is a identifier or variable.

Declaration of variable

Level# $ {Variable/Filler} $ Picture clause $ Value clause $ Usage Clause $ Sync clause.

Level#

It specifies the hierarchy of data within a record. It can take a value from the set of integers between 01-49 or from one of the special level-numbers 66 77 88

01 level	:-	Specifies the record itself. It may be either a group item or an Elementary item. It must begin in Area A.
02-49 levels	:-	Specify group or elementary items within a record. Group level items must not have picture clause.
66 level	:-	Identify the items that contain the RENAMES clause.
77 level	:-	Identify independent data item.
88 level	:-	Condition names.

Variable name and Qualifier

Variable name can have 1-30 characters with at least one alphabet in it. Hyphen is the only allowed special character but it cannot be first or last letter of the name. Name should be unique within the record. If two variables with same name are there, then use OF qualifier of high level grouping to refer a variable uniquely.
Ex: MOVE balance OF record-1 TO balance OF record-2.

FILLER

When the program is not intended to use selected fields in a record structure, define them as FILLER. FILLER items cannot be initialized or used in any operation of the procedure division.

PICTURE Clause

Describes the attributes of variable.

Numeric	9 (Digit), V (Implied decimal point), S (Sign)
Numeric Edited	+ (Plus Sign), - (Minus Sign), CR DB (Credit Debit Sign), '.' (Period), b (Blank), ','(comma), 0 (Zero), / (Slash) BLANK WHEN ZERO (Insert blank when data value is 0), Z (ZERO suppression), * (ASTERISK), $(Currency Sign)
Non Numeric	A (alphabet), B (Blank insertion Character), X(Alpha numeric), G(DBCS)
Exclusive sets	1. + - CR DB

2. V '.'
3. $ + - Z * (But $ Can appear as first place and * as floating. $***.**)

DBCS (Double Byte Character Set) is used in the applications that support large character sets. 16 bits are used for one character. Ex: Japanese language applications.

Refreshing Basics

Nibble	:- 4 Bits is one nibble. In packed decimal, each nibble stores one digit.
Byte	:- 8 Bits is one byte. By default, every character is stored in one byte.
Half word	:- 16 bits or 2 bytes is one half word. (MVS)
Full word	:- 32 bits or 4 bytes is one full word. (MVS)
Double word	:- 64 bits or 8 bytes is one double word. (MVS)

Usage Clause

DISPLAY	:-	Default. Number of bytes required equals to the size of the data item.
COMP	:-	Binary representation of data item. PIC clause can contain S and 9 only. S9(01) - S9(04) Half word. S9(05) - S9(09) Full word. S9(10) - S9(18) Double word. Most significant bit is ON if the number is negative.
COMP-1	:-	Single word floating point item. PIC Clause should not be specified.
COMP-2	:-	Double word floating-point item. PIC Clause should not be specified.
COMP-3	:-	Packed Decimal representation. Two digits are stored in each byte. Last nibble is for sign. (F for unsigned positive, C for signed positive and D for signed negative) Formula for Bytes: Integer ((n/2) + 1)) => n is number of 9s.
INDEX	:-	It is used for preserve the index value of an array. PIC Clause should not be specified.

VALUE Clause

It is used for initializing data items in the working storage section. Value of item must not exceed picture size. It cannot be specified for the items whose size is variable.

```
        VALUE IS literal.
Syntax: VALUES ARE literal-1 THRU | THROUGH literal-2
        VALUES ARE literal-1, literal-2
```

Literal can be numeric without quotes OR non-numeric within quotes OR figurative constant.

SIGN Clause

Syntax - SIGN IS (LEADING) SEPARATE CHARACTER (TRAILING).
It is applicable when the picture string contain 'S'. Default is TRAILING WITH NO SEPARATE

CHARACTER. So 'S' doesn't take any space. It is stored along with last digit.
+1=A, +2=B, +3=C, +4=D, +5=E, +6=F, +7=G, +8=H, +9=I, -0=}, -1= J, -2= K, -3=L, -4=M, -5=N, -6=O, -7=P, -8=Q, -9=R

Number	TRAILING SIGN (Default)	LEADING SIGN	LEADING SEPARATE
-125	12N	J25	-125
+125	12E	A25	+125

SYNC Clause and Slack Bytes

SYNC clause is specified with COMP, COMP-1 and COMP-2 items. These items are expected to start at half/full/double word boundaries for faster address resolution. SYNC clause does this but it may introduce slack bytes (unused bytes) before the binary item.

```
01 WS-TEST.
  10 WS-VAR1 PIC X(02).
  10 WS-VAR2 PIC S9(6) COMP SYNC.
```

Assumes WS-TEST starts at relative location 0 in the memory, WS-VAR1 occupies zero and first byte. WS-VAR2 is expected to start at second byte. As the comp item in the example needs one word and it is coded with SYNC clause, it will start only at the next word boundary that is 4th byte. So this introduces two slack bytes between WS-VAR1 and WS-VAR2.

REDEFINES

The REDEFINES clause allows you to use different data description entries to describe the same computer storage area. Redefining declaration should immediately follow the redefined item and should be done at the same level. Multiple redefinitions are possible. Size of redefined and redefining need not be the same.
Example:

```
01 WS-DATE          PIC9(06).
01 WS-REDEF-DATE WS-DATE.
   05                 WS-YEAR  PIC9(02).
   05                 WS-MON   PIC9(02).
   05                 WS-DAY   PIC9(02).
```

RENAMES

It is used for regrouping of elementary data items in a record. It should be declared at 66 levels. It need not immediately follows the data item, which is being renamed. But all RENAMES entries associated with one logical record must immediately follow that record's last data description entry. RENAMES cannot be done for a 01, 77, 88 or another 66 entry.

```
01   WS-REPSONSE.
   05 WS-CHAR143              PICX(03).
   05 WS-CHAR4                PICX(04).
   66 ADD-REPSONSE RENAMES WS-CHAR143.
```

CONDITION name

It is identified with special level '88'. A condition name specifies the value that a field can contain and used as abbreviation in condition checking.

```
01   SEX      PICX.
   88 MALE     VALUE '1'
   88 FEMALE VALUE '2' '3'.
```

IF SEX=1 can also be coded as IF MALE in Procedure division.
'SET FEMALE TO TRUE ' moves value 2 to SEX.
If multiple values are coded on VALUE clause, the first value will be moved when it is set to true.

JUSTIFIED RIGHT

This clause can be specified with alphanumeric and alphabetic items for right justification. It cannot be used with 66 and 88 level items.

OCCURS Clause

OCCURS Clause is used to allocate physically contiguous memory locations to store the table values and access them with subscript or index. Detail explanation is given in Table Handling section.

LINKAGE SECTION

It is used to access the data that are external to the program. JCL can send maximum 100 characters to a program thru PARM. Linkage section MUST be coded with a half word binary field, prior to actual field. If length field is not coded, the first two bytes of the field coded in the linkage section will be filled with length and so there are chances of 2 bytes data truncation in the actual field.

```
 01      LK-DATA.
    05 LK-LENGTH    PIC S9(04) COMP.
    05 LK-VARIABLE PIC X(08).
```

Procedure Division.

This is the last division and business logic is coded here. It has user-defined sections and paragraphs. Section name should be unique within the program and paragraph name should be unique within the section.
Procedure division statements are broadly classified into following categories.

Statement Type	Meaning
Imperative	Direct the program to take a specific action. Ex: MOVE ADD EXIT GO TO
Conditional	Decide the truth or false of relational condition and based on it, execute different paths . Ex: IF, EVALUATE
Compiler Directive	Directs the compiler to take specific action during compilation. Ex: COPY SKIP EJECT
Explicit Scope terminator	Terminate the scope of conditional and imperative statements. Ex: END-ADD END-IF END-EVALUATE
Implicit Scope terminator	The period at the end of any sentence, terminates the scope of all previous statements not yet terminated.

MOVE Statement

It is used to transfer data between internal storage areas defined in either file section or working storage section.

Syntax:

MOVE identifier1/literal1/figurative-constant TO identifier2 (identifier3)
Multiple move statements can be separated using comma, semicolons, blanks or the keyword THEN.

Numeric move rules:

A numeric or numeric-edited item receives data in such a way that the decimal point is aligned first and then filling of the receiving field takes place. Unfilled positions are filled with zero. Zero suppression or insertion of editing symbols takes places according to the rules of editing pictures. If the receiving field width is smaller than sending field then excess digits, to the left and/or to the right of the decimal point are truncated.

Alphanumeric Move Rules:

Alphabetic, alphanumeric or alphanumeric-edited data field receives the data from left to right. Any unfilled field of the receiving filed is filled with spaces. When the length of receiving field is

shorter than that of sending field, then receiving field accepts characters from left to right until it is filled. The unaccomodated characters on the right of the sending field are truncated. When an alphanumeric field is moved to a numeric or numeric-edited field, the item is moved as if it were in an unsigned numeric integer mode. CORRESPONDING can be used to transfer data between items of the same names belonging to different group-items by specifying the names of group-items to which they belong.
MOVE CORRESPONDING group-1 TO group-2

Group Move rule

When MOVE statement is used to move information at group level, the movement of data takes place as if both sending and receiving fields are specified as alphanumeric items. This is regardless of the description of the elementary items constituting the group item.
Samples for understanding MOVE statement (MOVE A TO B)

Picture of A	Value of A	Picture of B	Value of B after Move
PIC 99V99	12.35	PIC 999V99	012.35
PIC 99V99	12.35	PIC 9999V9999	0012.3500
PIC99V999	12.345	PIC9V99	2.34
PIC9(05)V9(03)	54321.543	PIC 9(03)V9(03)	321.543
PIC9(04)V9(02)	23.24	PICZZZ99.9	23.2
PIC99V99	00.34	PIC$$$.99	$.34
PICX(04)	MUSA	XBXBXB	M U S

ARITHMETIC VERBS

All the possible arithmetic operations in COBOL using ADD, SUBTRACT, MULTIPLY and DIVIDE are given below:

Arithmetic Operation	A	B	C	D
ADD A TO B	A	A+B		
ADD A B C TO D	A	B	C	A+B+C+D
ADD A B C GIVING D	A	B	C	A+B+C
ADD A TO B C	A	A+B	A+C	
SUBTRACT A FROM B	A	B-A		
SUBTRACT A B FROM C	A	B	C-(A+B)	
SUBTRACT A B FROM C GIVING D	A	B	C	C-(A+B)
MULTIPLY A BY B	A	A*B		
MULTIPLY A BY B GIVING C	A	B	A*B	
DIVIDE A INTO B	A	B/A		
DIVIDE A INTO B GIVING C	A	B	B/A	
DIVIDE A BY B GIVING C	A	B	A/B	
DIVIDE A INTO B GIVING C REMAINDER D	A	B	Integer (B/A)	Integer remainder

GIVING is used in the following cases:
1.To retain the values of operands participating in the operation.
2.The resultant value of operation exceeds any of the operand size.

ROUNDED option

With ROUNDED option, the computer will always round the result to the PICTURE clause specification of the receiving field. It is usually coded after the field to be rounded. It is prefixed with REMAINDER keyword ONLY in DIVIDE operation.
ADD A B GIVING C ROUNDED.
Caution: Don't use for intermediate computation.

ON SIZE ERROR

If A=20 (PIC 9(02)) and B=90 (PIC 9(02)), ADD A TO B will result 10 in B where the expected value in B is 110. ON SIZE ERROR clause is coded to trap such size errors in arithmetic operation.
If this is coded with arithmetic statement, any operation that ended with SIZE error will not be carried out but the statement follows ON SIZE ERROR will be executed.
ADD A TO B ON SIZE ERROR DISPLAY 'ERROR!'.

COMPUTE

Complex arithmetic operations can be carried out using COMPUTE statement. We can use arithmetic symbols than keywords and so it is simple and easy to code.
+ For ADD, - for SUBTRACT, * for MULTIPLY, / for DIVIDE and ** for exponentiation.
Rule: Left to right -
1.Parentheses
2.Exponentiation
3.Multiplication and Division
4.Addition and Subtraction
Caution: When ROUNDED is coded with COMPUTE, some compiler will do rounding for every arithmetic operation and so the final result would not be precise. 77 A PIC 999 VALUE 10
COMPUTE A ROUNDED = (A+2.95) *10.99
Result: (ROUNDED(ROUNDED(12.95) * ROUNDED(10.99)) =120 or ROUNDED(142.3205) = 142 So the result can be 120 or 142.
Be cautious when using ROUNDED keyword with COMPUTE statement.
All arithmetic operators have their own explicit scope terminators. (END-ADD, END-SUBTRACT, END-MULTIPLY, END-DIVIDE, END-COMPUTE). It is suggested to use them.
CORRESPONDING is available for ADD and SUBTRACT only.

INITIALIZE

VALUE clause is used to initialize the data items in the working storage section whereas INITIALIZE is used to initialize the data items in the procedure division. INITIALIZE sets the alphabetic, alphanumeric and alphanumeric-edited items to SPACES and numeric and numeric-edited items to ZERO. This can be overridden by REPLACING option of INITIALIZE. FILLER, OCCURS DEPENDING ON items are not affected.

Syntax: INITIALIZE identifier-1 REPLACING (ALPHABETIC/ALPHANUMERIC/ALPHA-NUMERIC-EDITED NUMERIC/NUMERIC-EDITED) DATA BY (identifier-2 /Literal-2)

ACCEPT

ACCEPT can transfer data from input device or system information contain in the reserved data items like DATE, TIME, DAY.
ACCEPT WS-VAR1 (FROM DATE/TIME/DAY/OTHER SYSTEM VARS).
If FROM Clause is not coded, then the data is read from terminal. At the time of execution, batch program will ABEND if there is no in-stream data from JCL and there is no FROM clause in the ACCEPT clause. DATE option returns six digit current date in YYYYMMDD
DAY returns 5 digit current date in YYDDD
TIME returns 8 digit RUN TIME in HHMMSSTT
DAY-OF-WEEK returns single digit whose value can be 1-7 (Monday-Sunday respectively)

DISPLAY

It is used to display data. By default display messages are routed to SYSOUT.
Syntax: DISPLAY identifier1| literal1 (UPON mnemonic name)

STOP RUN, EXIT PROGRAM & GO BACK

STOP RUN is the last executable statement of the main program. It returns control back to OS.
EXIT PROGRAM is the last executable statement of sub-program. It returns control back to main program.
GOBACK can be coded in main program as well as sub-program as the last statement. It just gives the control back from where it received the control.

Collating Sequence

There are two famous Collating Sequence available in computers. IBM and IBM Compatible machine use EBCDIC collating sequence whereas most micro and many mainframe systems use ASCII collating sequence. The result of arithmetic and alphabetic comparison would be same in both collating sequences whereas the same is not true for alphanumeric comparison

EBCDIC (Ascending Order)	ASCII (Ascending Order)
Special Characters	Special Characters
a-z	0-9
A-Z	A-Z
0-9	a-z

Default collating sequence can be overridden by an entry in OBJECT-COMPUTER and SPECIAL NAMES paragraphs.

1. Code the PROGRAM COLLATING SEQUENCE Clause in the Object computer paragraph.
PROGRAM COLLATING SEQUENCE IS alphabet-name
2. Map the alphabet-name in the SPECIAL-NAMES paragraph as follows:
ALPHABET alphabet-name is STANDARD-1 | NATIVE
NATIVE stands for computer's own collating sequence whereas STANDARD-1 stands for ASCII collating sequence.

IF/THEN/ELSE/END-IF

The most famous decision making statement in all language is 'IF'. The syntax of IF statement is given below: IF can be coded without any ELSE statement. THEN is a noise word and it is optional.
If ORs & ANDs are used in the same sentence, ANDs are evaluated first from left to right, followed by ORs.
This rule can be overridden by using parentheses. The permitted relation conditions are =, <, >, <=, >=, <>
CONTINUE is no operation statement. The control is just passed to next STATEMENT.
NEXT SENTENCE passes the control to the next SENTENCE.
If you forgot the difference between statement and sentence, refer the first page. It is advised to use END-IF, explicit scope terminator for the IF statements than period, implicit scope terminator.

```
IF condition1 AND condition2 THEN
Statement-Block-1
ELSE
IF condition3 THEN
CONTINUE
ELSE
IF condition4 THEN
Statement-Block-2
ELSE
NEXT SENTENCE
END-IF
END-IF
END-IF
```

Statement-Block-2 will be executed only when condition 1, 2 and 4 are TRUE and condition 3 is FALSE.
Implied operand: In compound conditions, it is not always necessary to specify both operands for each condition.
IF TOTAL=7 or 8 is acceptable. Here TOTAL=8 is implied operation.

SIGN test and CLASS test

SIGN test is used to check the sign of a data item. It can be done as follows -
IF identifier is POSITIVE/NEGATIVE/ZERO
CLASS test is used to check the content of data item against pre-defined range of values. It can be done as follows -
IF identifier is NUMERIC/ALPHABETIC/ALPHABETIC-HIGHER/ALPHABETIC-LOWER

You can define your own classes in the special names paragraph. We have defined a class DIGIT in our special names paragraph. It can be used in the following way. IF identifier is DIGIT

Negated conditions.

Any simple, relational, class, sign test can be negated using NOT.
But it is not always true that NOT NEGATIVE is equal to POSITIVE. (Example ZERO)

EVALUATE

With COBOL85, we use the EVALUATE verb to implement the case structure of other languages. Multiple IF statements can be efficiently and effectively replaced with EVALUATE statement. After the execution of one of the when clauses, the control is automatically come to the next statement after the END-EVALUATE. Any complex condition can be given in the WHEN clause. Break statement is not needed, as it is so in other languages.
General Syntax

```
EVALUATE         subject-1 (ALSO subject2..)
                 WHEN object-1 (ALSO object2..)
                 WHEN object-3 (ALSO object4..)
                 WHEN OTHER imperative statement
END-EVALUATE.
```

1.Number of Subjects in EVALUATE clause should be equal to number of objects in every WHEN clause.
2.Subject can be variable, expression or the keyword TRUE/ FLASE and respectively objects can be values, TRUE/FALSE or any condition.
3.If none of the WHEN condition is satisfied, then WHEN OTHER path will be executed.
Sample

```
EVALUATE         SQLCODE ALSO TRUE
                 WHEN 100 ALSO A=B imperative statement
                 WHEN -305 ALSO (A/C=4) imperative statement
                 WHEN OTHER imperative statement
END-EVALUATE.
```

PERFORM STATEMENTS

PERFORM will be useful when you want to execute a set of statements in multiple places of the program. Write all the statements in one paragraph and invoke it using PERFORM wherever needed. Once the paragraph is executed, the control comes back to next statement following the

PERFORM.

1.SIMPLE PERFORM.

```
 PERFORM PARA-1.
 DISPLAY 'PARA-1 executed'
 STOP RUN.
 PARA-1.
                          Statement1.
                          Statement2.
```

It executes all the instructions coded in PARA-1 and then transfers the control to the next instruction in sequence.

2.INLINE PERFORM.

When sets of statements are used only in one place then we can group all of them within PERFORM END-PERFORM structure. This is called INLINE PERFORM. This is equal to DO..END structure of other languages.

```
 PERFORM
                ADD A TO B
                MULTIPLE B BY C
                DISPLAY 'VALUE OF A+B*C ' C
 END-PERFORM
```

3. PERFORM PARA-1 THRU PARA-N.

All the paragraphs between PARA-1 and PARA-N are executed once.

4. PERFORM PARA-1 THRU PARA-N UNTIL condition(s).

The identifiers used in the UNTIL condition(s) must be altered within the paragraph(s) being performed; otherwise the paragraphs will be performed indefinitely. If the condition in the UNTIL clause is met at first time of execution, then named paragraph(s) will not be executed at all.

5. PERFORM PARA-1 THRU PARA-N N TIMES.

N can be literal defined as numeric item in working storage or hard coded constant.

6. PERFORM PARA-1 THRU PARA-N VARYING identifier1

FROM identifier 2 BY identifier3 UNTIL condition(s) Initialize identifier1 with identifier2 and test the condition(s). If the condition is false execute the statements in PARA-1 thru PARA-N and increment identifier1 BY identifier3 and check the condition(s) again. If the condition is again false, repeat this process till the condition is satisfied.

7.PERFORM PARA-1 WITH TEST BEFORE/AFTER UNTIL condition(s).

With TEST BEFORE, Condition is checked first and if it found false, then PARA-1 is executed and this is the default. (Functions like DO- WHILE) With TEST AFTER, PARA-1 is executed once and then the condition is checked. (Functions like DO-UNTIL)

EXIT statement.

COBOL reserved word that performs NOTHING. It is used as a single statement in a paragraph that indicate the end of paragraph(s) execution. EXIT must be the only statement in a paragraph in COBOL74 whereas it can be used with other statements in COBOL85.

GO TO Usage:

In a structured top-down programming GO TO is not preferable. It offers permanent control transfer to another paragraph and the chances of logic errors is much greater with GO TO than PERFORM. The readability of the program will also be badly affected.
But still GO TO can be used within the paragraphs being performed. i.e. When using the THRU option of PERFORM statement, branches or GO TO statements, are permitted as long as they are within the range of named paragraphs.

```
PERFORM    100-STEP1 THRU STEP-4
100-STEP-1.
           ADD A TO B GIVING C.
           IF D = ZERO DISPLAY 'MULTIPLICATION NOT DONE'
           GO TO 300-STEP3
           END-IF.
200-STEP-2.
           MULTIPLY C BY D.
300-STEP-3.
           DISPLAY 'VALUE OF C:' C.
```

Here GO TO used within the range of PERFORM. This kind of Controlled GO TO is fine with structured programming also!

TABLES

An OCCURS clause is used to indicate the repeated occurrences of items of the same format in a structure. OCCURS clause is not valid for 01, 77, 88 levels.
It can be defined as elementary or group item. Initialization of large table occurrences with specific values are usually done using perform loops in procedure division. Simple tables can be initialized in the following way.

```
 01      WEEK-ARRAY VALUE 'MONTUEWEDTHUFRISATSUN'.
     05 WS-WEEK-DAYS OCCURS 7 TIMES PIC X(03).
```

Dynamic array is the array whose size is decided during runtime just before the access of first element of the array.

```
 01      WS-MONTH-DAY-CAL.
     05 WS-DAYS OCCURS 31 TIMES DEPENDING ON WS-OCCURENCE.
```

IF MONTH = 'FEB' MOVE '28' to WS-OCCURRENCE.
Array Items can be accessed using INDEX or subscript and the difference between them are listed in the table. Relative subscripts and relative indexes are supported only in COBOL85. Literals used in relative subscripting/indexing must be an unsigned integer.
ADD WS-SAL(SUB) WS-SAL(SUB + 1) TO WS-SAL(SUB + 2).

Sl #	Subscript	Index
1	Working Storage item	Internal Item-No need to declare it.
2	It means occurrence	It means displacement
3	Occurrence, in turn translated to displacement to access elements and so slower than INDEX access.	Faster and efficient.
4	It can be used in any arithmetic operations or for display.	It cannot be used for arithmetic operation or for display purpose.
5	Subscripts can be modified by any arithmetic statement.	INDEX can only be modified with SET, SEARCH and PERFORM statements.

Sometimes, you may face a question like how to randomly access the information in the sequential file of 50 records that contains all the designation and the respective lower and higher salary information.
Obviously, OS does not allow you to randomly access the sequence file. You have to do by yourself and the best way is, load the file into a working storage table in the first section of the program and then access as you wish.
The table look-up can be done in two ways.
-Sequential search.
-Binary search.

Sequential SEARCH

During SERIAL SEARCH, the first entry of the table is searched. If the condition is met, the table look-up is completed. If the condition is not met, then index or subscript is incremented by one and the next entry is searched and the process continues until a match is found or the table has been completely searched.
SET indexname-1 TO 1.
SEARCH identifier-1 AT END display 'match not found:'

WHEN condition-1 imperative statement-1 /NEXT SENTENCE
WHEN condition-2 imperative statement-2 /NEXT SENTENCE
END-SEARCH
Identifier-1 should be OCCURS item and not 01 item.
Condition-1, Condition-2 compares an input field or search argument with a table argument.
Though AT END Clause is optional, it is highly recommended to code that. Because if it is not coded and element looking for is not found, then the control simply comes to the next statement after SEARCH where an invalid table item can be referred and that may lead to incorrect results / abnormal ends.
SET statement Syntax:
SET index-name-1 TO/UP BY/DOWN BY integer-1.

Binary SEARCH

When the size of the table is large and it is arranged in some sequence -either ascending or descending on search field, then BINARY SEARCH would be the efficient method.
SEARCH ALL identifier-1 AT END imperative-statement-1
WHEN dataname-1 = identifier-2/literal-1/arithmetic expression-1
AND dataname-2 = identifier-3/literal-2/arithmetic expression-2
END-SEARCH.
Identifier-2 and identifier-3 are subscripted items and dataname-1 and dataname-2 are working storage items that are not subscripted.
Compare the item to be searched with the item at the center. If it matches fine, else repeat the process with the left or right half depending on where the item lies.

Sl #	Sequential SEARCH	Binary SEARCH
1	SEARCH	SEARCH ALL
2	Table should have INDEX.	Table should have INDEX.
3	Table need not be in SORTED order.	Table should be in sorted order of the searching argument. There should be ASCENDING/DESCENDING Clause.
4	Multiple WHEN conditions can be coded.	Only one WHEN condition can be coded.
5	Any logical comparison is possible.	Only = is possible. Only AND is possible in compound conditions.
6	Index should be set to 1 before using SEARCH	Index need not be set to 1 before SEARCH ALL.
7	Prefer when the table size is small	Prefer when the table size is significantly large.

Multi Dimensional Arrays

COBOL74 supports array of maximum of three dimensions whereas COBOL85 supports up to seven dimensions. The lowest- level OCCURS data-name or an item subordinate to it is used to access an entry in the array or the table.
If we use SEARCH for accessing multi-dimension table, then INDEXED BY must be used on all OCCURS levels. Expanded nested perform is available for processing multi level tables. The syntax of this perform is given below:

PERFORM para-1 thru para-n VARYING index-1 from 1 BY 1 UNTIL index-1 > size- of- outer-occurs AFTER VARYING index-2 from 1 by 1 until index-2 > size of inner occurs.
SEARCH example for multi level tables:

```
01          EMP-TABLE.
   05       DEPTNUMBER OCCURS 10 TIMES INDEXED BY I1.
      10    EMP-DETAIL OCCURS 50 TIMES INDEXED BY I2.
         15 EMP-NUMBER PIC9(04).
         15 EMP-SALARY PIC9(05).
77          EMPNUMBER-IN PIC 9(04) VALUE '2052'.
```

```
PERFORM 100-SEARCH-EMP-SAL VARYING I1 FROM 1 BY 1
UNTIL I1 > 10 OR WS-FOUND
100-SEARCH-EMP-SAL.
SET I2 TO I.
SEARCH EMP-DETAIL AT END DISPLAY 'NOT FOUND'. == > Lowest Occurs
WHEN EMPNUMBER-IN = EMP-NUMBER(I1,I2).
DISPLAY 'SALARY IS:' EMP-SALARY(I1,I2).
SET WS-FOUND TO TRUE. == > Search ends
END-SEARCH.
```

NESTED PROGRAMS, GLOBAL, EXTERNAL

One program may contain other program(s). The contained program(s) may themselves contain yet other program(s). All the contained and containing programs should end with END PROGRAM statement. PGMB is nested a program in the example below:

```
          IDENTIFICATION DIVISION.
Example:  PROGRAM-ID. PGMA
          .....
          IDENTIFICATION DIVISION.
          PROGRAM-ID. PGMB
          .....
          END PROGRAM PGMB.
          ....
          END PROGRAM PGMA.
```

If you want access any working storage variable of PGMA in PGMB, then declare them with the clause 'IS GLOBAL' in PGMA. If you want to access any working storage variable of PGMB in PGMA, declare them with the clause 'IS EXTERNAL' in PGMB. Nested Programs are supported only in COBOL85.
If there is a program PGMC inside PGMB, it cannot be called from PGMA unless it's program id is qualified with keyword COMMON.

SORT and MERGE

The programming SORT is called as internal sort whereas the sort in JCL is called external sort. If you want to manipulate the data before feeding to sort, prefer internal sort. In all other cases, external sort is the good choice. Internal sort, in turn invokes the SORT product of your installation. (DFSORT). In the run JCL, allocate at least three sort work files. (SORT-WKnn => nn can be 00-99).
FASTSRT compiler option makes the DFSORT to do all file I-O operation than your COBOL program. It would significantly improve the performance. The result of the SORT can be checked in SORT-RETURN register. If the sort is successful, the value will be 0 else 16.

Syntax:

SORT SORTFILE ON ASCENDING /DESCENDING KEY sd-key-1 sd-key2
USING file1 file2 / INPUT PROCEDURE IS section-1
GIVING file3 / OUTPUT PROCEDURE is section-2
END-SORT
File1, File2 are to-be-sorted input files and File3 is sorted-output file and all of them are defined in FD.SORTFILE is Disk SORT Work file that is defined at SD. It should not be explicitly opened or closed.
INPUT PROCEDURE and USING are mutually exclusive. If USING is used, then file1 and files should not be opened or READ explicitly. If INPUT PROCEDURE is used then File1 and file2 need to be OPENed and READ the records one by one until end of the file and pass the required records to sort-work-file using the command RELEASE.

Syntax:

RELEASE sort-work-record from input-file-record.
OUTPUT Procedure and GIVING are mutually exclusive. If GIVING is used, then file3 should not be opened or WRITE explicitly. If OUTPUT procedure is used, then File3 should be OPENed and the required records from sort work file should be RETURNed to it. Once AT END is reached for sort-work-file, close the output file.

Syntax:

RETURN sort-work-file-name AT END imperative statement.

INPUT PROCEDURE Vs OUTPUT PROCEDURE

Sometimes it would be more efficient to process data before it is sorted, whereas other times it is more efficient to process after it is sorted. If we intend to eliminate more records, then it would be better preprocess them before feeding to SORT. If we want to eliminate all the records having spaces in the key field then it would be efficient if we eliminate them after sorting. Because the records with blank key comes first after sorting.

MERGE

It is same as sort. USING is mandatory. There should be minimum two files in USING.
MERGE Sort-work-file ON ASCENDING KEY dataname1 dataname2
USING file1 file2
GIVING file3 / OUTPUT PROCEDURE is section-1
END-MERGE

Program sort registers (and its equivalent DFSORT parameter/meaning)

SORT-FILE-SIZE (FILSZ), SORT-CORE-SIZE (RESINV), SORT-MSG(MSGDDN)
SORT-MODE-SIZE (SMS=nnnnn)
SORT-RETURN(return-code of sort) and
SORT-CONTROL (Names the file of control card - default is IGZSRTCD)

STRING MANIPULATION

A string refers to a sequence of characters. String manipulation operations include finding a particular character/sub-string in a string, replacing particular character/sub-string in a string, concatenating strings and segmenting strings. All these functions are handled by three verbs INSPECT, STRING and UNSTRING in COBOL. EXAMINE is the obsolete version of INSPECT supported in COBOL74.

INSPECT- FOR COUNTING

It is used to tally the occurrence of a single character or groups of characters in a data field.
INSPECT identifier-1 TALLYING identifier-2 FOR ALL/LEADING literal-1|identifier-3 [BEFORE|AFTER INITIAL identifier-4|literal-2] - Optional.
INSPECT identifier-1 TALLYING identifier-2 FOR CHARACTERS [BEFORE|AFTER INITIAL identifier-4|literal-2] - Optional.
Main String is identifier-1 and count is stored in identifier-2. Literal-1 or Identifier-3 is a character or group-of-characters you are looking in the main-string. INSPECT further qualifies the search with BEFORE and AFTER of the initial occurrence of identifier-4 or literal-2.
Example:
WS-NAME - 'MUTHU SARAVANA SURYA CHANDRA DEVI'
INSPECT WS-NAME TALLYING WS-COUNT ALL 'S'
BEFORE INITIAL 'SARAVANA' AFTER INITIAL 'CHANDRA'
END-INSPECT
Result:
WS-COUNT contains - 1

INSPECT- FOR REPLACING

It is used to replace the occurrence of a single character or groups of characters in a data field.
INSPECT identifier-1 REPLACING ALL|LEADING literal-1|identifier-2 BY identifier-3|literal-2 [BEFORE|AFTER INITIAL identifier-4|literal-2] - Optional.

INSPECT identifier-1 REPLCING CHARACTERS BY identifier-2 BEFORE|AFTER INITIAL identifier-3|literal-1
INSPECT-FOR COUNTING AND REPLACING It is a combination of the above two methods.
INSPECT identifier-1 TALLYING (tallying part) REPLACING (replacing part)

STRING

STRING command is used to concatenate one or more strings.

Syntax:

STRING identifier-1 / literal-1, identifier-2/ literal-2 DELIMITED BY (identifier-3/literal-3/SIZE) INTO identifier-4 END-STRING.
01 VAR1 PIC X(10) VALUE 'MUTHU '
01 VAR2 PIC X(10) VALUE 'SARA '
01 VAR2 PIC X(20).
To get display 'MUTHU,SARA'
STRING VAR1 DELIMITED BY ' ' ',' DELIMITED BY SIZE VAR2 DELIMITED BY ' ' INTO VAR3 END-STRING.
The receiving field must be an elementary data item with no editing symbols and JUST RIGHT clause.
With STRING statement, specific characters of a string can be replaced whereas MOVE replaces the full string.
01 AGE-OUT PIC X(12) VALUE '12 YEARS OLD'.
STRING '18' DELIMITED BY SIZE INTO AGE-OUT. => 18 YEARS OLD.

Reference Modification - equivalent of SUBSTR

'Reference modification' is used to retrieve or overwrite a sub-string of a string. ':' is known as reference modification operator.

Syntax:

String(Starting-Position:Length)
MOVE '18' TO AGE-OUT(1:2) does the same as what we did with STRING command.
When it is used in array elements, the syntax is Array-element (occurrence) (Starting-Position:Length)

UNSTRING

UNSTRING command is used to split one string to many strings.

Syntax:

UNSTRING identifier-1 [DELIMITED BY (ALL/) identifier2/literal1 [,OR (ALL/) (identifier-3/literal-2),..]] INTO identifier-4 [,DELIMITER IN identifier-5, COUNT IN identifier-6] [,identifier-7 [,DELIMITER IN identifier-8, COUNT IN identifier-9]
01 WS-DATA PIC X(12) VALUE '10/200/300/1'.
UNSTRING WS-DATA DELIMITED BY '/'
INTO WS-FLD1 DELIMITER IN WS-D1 COUNT IN WS-C1
WS-FLD2 DELIMITER IN WS-D2 COUNT IN WS-C2
WS-FLD3 DELIMITER IN WS-D3 COUNT IN WS-C3
END-UNSTRING.

Result:

WS-FLD1 = 10 WS-FLD2 =200 WS-FLD3=300 WS-C1 = 2 WS-C2=3 WS-C3=3 WS-D1 = '/' WS-D2='/' WS-D3 ='/'
ON OVERFLOW can be coded with STRING and UNSTRING. If there is STRING truncation then the imperative statements followed ON OVERFLOW will be executed.

COPY Statement

A COPY statement is used to bring a series of prewritten COBOL entries that have been stored in library, into a program.
1.Common routines like error routine, date validation routine are coded in a library and bring into the program by COPY.
2. Master files are used in multiple programs. Their layout can be placed in one copybook and be placed wherever the files are used. It promotes program standardization since all the programs share the same layout and the same data names.
This reduces coding and debugging time. Change in layout needs change in copybook only. It is enough if we just recompile the program for making the new copy effective.

Syntax:

COPY copybook-name [(OF/IN) library name]
[REPLACING string-to-be-replaced BY replacing-string]
Copybooks are stored as members in PDS library and during compilation time, they are included into the program. By default, the copybook library is SYSLIB and it can be changed using IN or OF of COPY statement.
Copybooks can be used in the following paragraphs.
SOURCE-COMPUTER, OBJECT-COMPUTER, SPECIAL-NAMES, FILE-CONTROL, IO-CONTROL, FD SECTION, PARAGRAPHS IN PROCEDURE DIVISION.
If the same copybook is used more than once in the program, then there will be "duplicate data declaration" error during compilation, as all the fields are declared twice. In this case, one copybook can be used with REPLACING verb to replace high-level qualifier of the all the variables with another qualifier.

Example:

COPY CUSTOMER REPLACING 'CUST1-' BY 'CUST2-'.
Delimiter '= =' should be used for replacing pseudo texts. The replacing option does not alter the prewritten entries in the library; the changes are made to the user's source program only.

CALL statement (Sub-Programs)

When a specific functionality need to be performed in more than one program, it is best to write them separately and call them into each program. Sub Programs can be written in any programming language. They are typically written in a language best suited to the specific task required and thus provide greater flexibility.

Main Program Changes:

CALL statement is used for executing the sub-program from the main program. A sample of CALL statement is given below:
CALL 'PGM2' USING BY REFERENCE WS-VAR1, BY CONTENT WS-VAR2.
PGM2 is called here. WS-VAR1 and WS-VAR2 are working storage items.
WS-VAR1 is passed by reference. WS-VAR2 is passed by Content. BY REFERENCE is default in COBOL and need not be coded. BY CONTENT LENGTH phrase permits the length of data item to be passed to a called program.

Sub-Program Changes:

WS-VAR1 and WS-VAR2 are working storage items of main program.
As we have already mentioned, the linkage section is used for accessing external elements. As these working storage items are owned by main program, to access them in the sub-program, we need to define them in the linkage section.
LINKAGE SECTION.

```
 01      LINKAGE SECTION.
     05 LK-VAR1 PIC9(04).
     05 LK-VAR2 PIC9(04).
```

In addition to define them in linkage section, the procedure division should be coded with these data items for address-ability.

PROCEDURE DIVISION USING LK-VAR1,LK-VAR2
There is a one-one correspondence between passed elements and received elements (Call using, linkage and procedure division using) BY POSITION. This implies that the name of the identifiers in the called and calling program need not be the same (WS-VAR1 & LK-VAR1) but the number of elements and picture clause should be same.
The last statement of your sub-program should be EXIT PROGRAM. This returns the control back to main program. GOBACK can also be coded instead of EXIT PROGRAM but not STOP RUN. EXIT PROGRAM should be the only statement in a paragraph in COBOL74 whereas it can be coded along with other statements in a paragraph in COBOL85.
PROGRAM-ID. IS INITIAL PROGRAM.

If IS INITIAL PROGRAM is coded along with program-id of sub program, then the program will be in initial stage every time it is called (COBOL85 feature). Alternatively CANCEL issued after CALL, will set the sub-program to initial state.
If the sub program is modified then it needs to be recompiled. The need for main program recompilation is decided by the compiler option used for the main program. If the DYNAM compiler is used, then there is no need to recompile the main program. The modified subroutine will be in effect during the run. NODYNAM is default that expects the main program recompilation.
Difference between Pass-by-reference and Pass-by-content

Sl #	Passl By Reference	Pass By Content
1	CALL 'sub1' USING BY REFERENCE WS-VAR1	CALL 'sub1' USING BY CONTENT WS-VAR1 (BY CONTENT keyword is needed)
2	It is default in COBOL. BY REFERENCE is not needed.	BY CONTENT key word is mandatory to pass an element by value.
3	Address of WS-VAR1 is passed	Value of WS-VAR1 is passed
4	The sub-program modifications on the passed elements are visible in the main program.	The sub-program modifications on the passed elements are local to that sub-program and not visible in the main program.

Difference between Static Call and Dynamic Call

Sl #	STATIC Call	DYNAMIC Call
1	Identified by Call literal. Ex: CALL 'PGM1'.	Identified by Call variable and the variable should be populated at run time. 01 WS-PGM PIC X(08). Move 'PGM1' to WS-PGM CALL WS-PGM
2	Default Compiler option is NODYNAM and so all the literal calls are considered as static calls.	If you want convert the literal calls into DYNAMIC, the program should be compiled with DYNAM option. By default, call variables and any un-resolved calls are considered as dynamic.
3	If the subprogram undergoes change, sub program and main program need to be recompiled.	If the subprogram undergoes change, recompilation of subprogram is enough.
4	Sub modules are link edited with main module.	Sub modules are picked up during run time from the load library.
5	Size of load module will be large	Size of load module will be less.
6	Fast	Slow compared to Static call.
7	Less flexible.	More flexible.
8	Sub-program will not be in initial stage the next time it is called unless you explicitly use INITIAL or you do a CANCEL after each call.	Program will be in initial state every time it is called.

INTRINSIC FUNCTIONS:

LENGTH	:-	Returns the length of the PIC clause. Used for finding length of group item that spanned across multiple levels.
MAX	:-	Returns the content of the argument that contains the maximum value
MIN	:-	Returns the content of the argument that contains the minimum value
NUMVAL	:-	Returns the numeric value represented by an alphanumeric character string specified in the argument.
NUMVAL-C	:-	Same as NUMVAL but currency and decimal points are ignored during conversion.
CURRENT DATE	:-	Returns 21 Chars alphanumeric value - YYYYMMDDHHMMSSnnnnnn
INTEGER OF DATE	:-	Returns INTEGER equivalent of Gregorian date passed.
INTEGER OF DAY	:-	Returns INTEGER equivalent of Julian date passed.
DATE OF INTEGER	:-	Returns Gregorian date for the integer passed.
DAY OF INTEGER	:-	Returns Julian date for the integer passed.

Note: FUNCTION INTEGER OF DATE (01-01-1601) returns 1.

FILE HANDLING

A data file is collection of relevant records and a record is collection of relevant fields. The file handling in COBOL program involves five steps.

Steps in file-handing

1.Allocation: The files used in the program should be declared in FILE-CONTROL paragraph of environment division. The mapping with JCL DDNAME is done here. The file is allocated to your program by this statement.
2.Definition. The layout of the file and its attributes are defined in the FILE SECTION of DATA DIVISION.
3.Open: Dataset is connected/readied to your program using OPEN statement. The mode of OPEN decides the operation allowed and the initial pointer in the dataset. For example, EXTEND mode allows only write access and the pointer is kept on the end of file to append.
4.Process: Process the file as per requirement, using the I-O statements provided by COBOL. (READ, WRITE, REWRITE and DELETE)
5. Close: After the processing, close the file to disconnect it from the program.

Allocation of file - SELECT Statement

(ENVIRONMENT-> INPUT-OUTPUT-> FILE-CONTROL)

SELECT [OPTIONAL] FILENAME ASSIGN to DDNAME	:- ALL Files
ORGANIZATION IS SEQUENTIAL/INDEXED/RELATIVE	:- ALL Files
ACCESS IS SEQUNETIAL/RANDOM/DYNAMIC	:- ALL Files

RECORD KEY IS FILE-KEY1	:- KSDS
RELATIVE KEY IS WS-RRN	:- RRDS
ALTERNARE RECORD KEY IS FILE-KEY2 WITH DUPLICATES	:- KSDS
ALTERNARE RECORD KEY IS FILE-KEY3 WITHOUT DUPLICATES	:- AIX
FILE STATUS IS WS-FILE-STAT1	:- ALL Files
[,WS-FILE-STAT2]	:- VSAM Files

SELECT Statement- OPTIONAL Clause

This can be coded only for input files. If OPTIONAL is not coded, then the input file is expected to present in JCL. If not, an execution error will occur. If OPTIONAL is coded, then if the file is not mapped in JCL, it is considered as empty file and the first read results end of file. The file can also be dynamically allocated instead of static allocation in JCL.

SELECT Statement- ASSIGN TO

FILENAME is the logical name used inside the program and DDNAME is the logical name in the JCL, mapped with physical dataset. DDNAME can be prefixed with 'S-' to indicate QSAM file, '-AS' to indicate ESDS file and with no prefix to indicate KSDS/RRDS file.
JCL Step executing the program should have a dataset with DDNAME as label
//DDNAME DD DSN=BPMAIN.EMPLOYEE.DATA,DISP=SHR

SELECT Statement-ORGANIZATION

It can be SEQUENTIAL (PS or VSAM ESDS), INDEXED (VSAM KSDS), RELATIVE (VSAM RRDS). Default is Sequential.

SELECT Statement-ACCESS MODE

SEQUENTIAL.

It is default access mode and it is used to access the records ONLY in sequential order. To read 100th record, first 99 records need to be read and skipped.

RANDOM.

Records can be randomly accessed in the program using the primary/alternate key of indexed file organization or relative record number of relative organization.100th record can directly be read after getting the address of the record from the INDEX part for INDEXED files.100th record can directly be read for RELATIVE files even without any index.

DYNAMIC.

It is mixed access mode where the file can be accessed in random as well as sequential mode in the program. Example: Reading the details of all the employees between 1000-2000. First randomly access 1000th employee record, then read sequentially till 2000th employee record. START and READ NEXT commands are used for this purpose in the procedure division.

SELECT Statement-RECORD KEY IS

It is primary key of VSAM KSDS file. It should be unique and part of indexed record structure.

SELECT Statement-ALTERNATE RECORD KEY IS

This phrase is used for KSDS files defined with AIX. Add the clause WITH DUPLICATES if the AIX is defined with duplicates. Referring to VSAM basics, every alternate index record has an associated PATH and the path should be allocated in the JCL that invokes this program. The DDNAME of the path should be DDNAME of the base cluster suffixed with 1 for the first alternate record clause, suffixed with n for nth ALTERNATE RECORD KEY clause in SELECT clause.

SELECT Statement-FILE STATUS IS WS-FILE-STAT1,WS-FILE-STAT2

WS-FILE-STAT1 should be defined as PIC X(02) in working storage section. After every file operation, the file status should be checked for allowable values. WS-FILE-STAT2 can be coded for VSAM files to get the VSAM return code (2 bytes), VSAM function-code (1 byte) and VSAM feedback code (3 bytes). This is a 6- byte field in working storage.

RESERVE Clause.

RESERVE clause [RESERVE integer AREA] can be coded in the SELECT statement. The number of buffers to be allocated for the file is coded here. By default two buffers will be allocated if the clause is not coded. Since similar option is available in JCL, this is not coded in program. RESERVE 1 AREA allocates one buffer, for the file in the SELECT statement.

Defining the file in FILE SECTION - FD

```
FD FILENAME
RECORDING MODE IS V/VB/F/FB
RECORD CONTAINS M CHARACTERS (TO N CHARACTERS)
BLOCK CONTAINS X CHARACTERS/RECORDS (TO Y CHARACTERS/RECORDS)
LABEL RECORDS ARE OMITTED/STANDARD
DATA RECORD IS FILE-RECORD.
01 FILE-RECORD PIC X(nnn).
```

FD-RECORD CONTAINS

It specifies the length of the record in terms of bytes. (It will be RECORD contains m to n CHARACTERS for variable format files)

FD-BLOCK CONTAINS

It specifies the physical record size. It can be mentioned as number of logical records OR number of characters, that is multiple of logical record length. It is suggested to code BLOCK CONTAINS 0 RECORDS so that system will decide the optimum size for the file based on the device used for storing the file. BLOCK CONTAINS clause is treated as comments for VSAM files.

Advantage of Blocking:

1.I-O time is reduced as n numbers of records are read into main memory buffer during an I-O.
2.Inter record gap is removed and the gap exist only between blocks. So memory wastage due to IRG is avoided.

FD-RECORDING MODE IS

It can be F (FIXED) V(VARIABLE) FB(FIXED BLOCK) VB(VARIABLE BLOCKED)

Variable record file identification:

If there is no recording mode/record contains clause, it is still possible to identify variable length records. If there is an OCCURS depending on clause or there are multiple 01 levels and every 01 level is of different size, then the file would be of variable length. Multiple 01 level in File section is an example for implicit redefinition.

FD-LABEL RECORDS Clause

As a general rule, LABEL RECORDS are STANDARD is coded for Disk and Tape files, LABEL RECORDS ARE OMITTED is coded for printer files. In COBOL74, this clause is a mandatory clause whereas COBOL85 made this as optional.

FD-DATA RECORD IS Clause

It is used to name the data record(s) of the file. More than one record can be coded here.

OPEN STATEMENT

Syntax:

OPEN OPENMODE FILENAME
OPENMODE can be INPUT OUTPUT I-O EXTEND

INPUT	:- File can be used ONLY-FOR-READ purpose.
OUTPUT	:- File can be used ONLY-FOR-WRITE purpose.
I-O	:- File can be used FOR READ, WRITE and REWRITE purpose.
EXTEND	:- File can be used FOR appending records using WRITE.

CLOSE statement.

The used files are closed using CLOSE statement. If you don't close the files, the completion of the program closes all the files used in the program.

Syntax:

CLOSE FILENAME

OPEN and CLOSE for TAPE files - Advanced

If more than one file is stored in a reel of tape, it is called as multi-file volume. When one file is stored in more than one reel of tape, it is called as multi-volume label. One reel is known as one volume. When the end of one volume is reached, automatically the next volume opens. So there is no special control is needed for multi volume files.

```
OPEN INPUT file-1     [WITH NO REWIND | REVERSED]
OPEN OUTPUT file-2 [WITH NO REWIND]
CLOSE file-3             [{REEL|UNIT} [WITH NO REWIND| FOR REMOVAL]
CLOSE file-3             [WITH NO REWIND|LOCK]
```

UNIT and REEL are synonyms.

After opening a TAPE file, the file is positioned at its beginning. When opening the file if the clause REVERSED is coded, then the file can be read in the REVERESE direction. (Provided hardware supports this feature)
When you close the file, the tape is normally rewound. The NO REWIND clause specifies that the TAPE should be left in its current position.
CLOSE statement with REEL option closes the current reel alone. So the next READ will get the first record of next REEL. This will be useful when you want skip all the records in the first reel after n number of records processing.
Since TAPE is sequential device, if you create multiple files in the same TAPE, then before opening the second file, first file should be closed. At any point of time, you can have only one file is active in the program. In addition to this, you have to code MULTIPLE FILE clause in the I-O control paragraph of environment division.
MULTIPLE FILE TAPE CONTAINS OUT-FILE1 POSITION 1
OUT-FILE3 POSITION 3.
The files OUT-FILE1 and OUT-FILE3 used in the program are part of a same TAPE and they exist in first and third position in the tape. Alternatively, this information can be passed from JCL using LABEL parameter.

READ statement

READ statement is used to read the record from the file.

Syntax:

```
READ FILENAME [INTO ws-record] [KEY IS FILE-KEY1]
              [AT END/INVALID KEY imperative statement1]
              [NOT AT END/NOT INVALID KEY imperative statement2]
END-READ
```

If INTO clause is coded, then the file is directly read into working storage section record. It is preferred as it avoids another move of file-section-record to working-storage-record followed by simple READ. READ-INTO is not preferred for variable size records where the length of the record being read is not known.
KEY IS clause is used while accessing a record randomly using primary/alternate record key.
AT END and NOT AT END are used during sequential READ of the file.
INVALID KEY and NOT INVALID KEY are used during random read of the file. Before accessing the file randomly, the key field should have a value before READ.

WRITE Statement

Write statement is used to write a new record in the file. If the file is opened in EXTEND mode, the record will be appended. If the file is opened in OUTPUT mode, the record will be added at the current position.

Syntax:

```
WRITE FILE-RECORD [FROM ws-record]
                  [INVALID KEY imperative statement1]
END-WRITE
```

FROM clause avoids the explicit move of working storage record to file section record before WRITE.

REWRITE Statement

REWRITE is used to update an already read record. To update a record in a file, the file should be opened in I-O mode.

Syntax:

```
REWRITE FILE-RECORD [FROM ws-record]
                    [INVALID KEY imperative statement1]
END-REWRITE
```

START Statement

START is used with dynamic access mode of indexed files. It establishes the current location in the cluster for READ NEXT statement. START itself does not retrieve any record.

Syntax:

```
START FILENAME KEY is EQUAL TO/NOT LESS THAN/GREATER THAN key-name
                        [INVALID KEY imperative statement1]
END-START.
```

DELETE Statement

DELETE is used to delete the most recently read record in the file. To delete a record, the file should be opened in I-O mode.

Syntax:

```
DELETE FILENAME RECORD
                         [INVALID KEY imperative statement1]
END-DELETE.
```

File Error - Handling

There are chances for failure of any file I-O processing. The failure of an I-O operation can be accepted or cannot be tolerated. The severity of failure has to be defined in the program design stage.
Let us assume that we don't have any error handling in our program. In this case, for example, if you don't have a specific record in the file, the random read of that record would immediately terminate the program with error 'record not found'.

Error Handling Clauses Provided by COBOL.

The sudden termination can be avoided by handling this error, with INVALID KEY clause of READ. Based on the importance of the record and business rule, we can continue our program with next record or terminate the program properly.
AT END is another error handling clause provided by COBOL. But there is no way to handle all such errors in this way.

Assign file-status and take the responsibility.

The second method is, assigning file-status to the file in the SELECT clause and checks the file status after each and every I-O and ensures that the value of status code is one of the allowable values. If it is not an allowable return code, then abnormally end the program with error statements that would be easier to debug.
But we have to do this checking after each and every I-O operation.
This is MOST PREFERRED ERROR HANDLING METHOD in structured programming.

Declaratives - USE statement

COBOL provides an option to group all the possible errors of specific operation(s) in a place and that will be automatically invoked during the respective operation(s) of any file. This avoids redundant code.
This is done in DECLARATIVE section of the procedure division. DECLARATIVE should be the

first section in the procedure division if coded.

```
PROCEDURE DIVISION.
DECLARATIVES.
USE-PROCEDURE SECTION.
  USE AFTER EXCEPTION PROCEDURE ON INPUT.
ERROR-PROCEDURE.
  Check the file-status code for validity.
END-DECLARATIVES.
```

Whenever there is an error in the processing of ANY FILE opened in INPUT mode, then the control comes to ERROR-PROCEDURE. The validity of error should be checked in this paragraph and allow or restrict the process down, based on severity of error code.

Syntax:

USE AFTER STANDARD ERROR|EXCEPTION PROCEDURE ON INPUT|OUTPUT|I-O|EXTEND| file-1

If INPUT is coded, the following procedure will be executed for every operation involved in any file that is opened in INPUT mode. OUTPUT, I-O and EXTEND have the same meaning but the mode is different.

If file name (file-1) is coded in the USE statement, then all the input-output operation of that specific file will be checked.

ERROR and EXCEPTION are synonyms.

The Procedure written in a DECLARATIVE section should not refer to any non-declarative procedure written after the end procedure and vice-versa.

I-O-CONTROL - SAME AREA AND SAME RECORD AREA

RESERVE clause of SELECT statement specifies the number of buffers to be allocated for a file. SAME AREA allows more than one file to use the same buffer area. This will be very useful when the program must work with a limited memory space. But the problem is only one file should be open at a time if SAME AREA is coded.

Syntax:

SAME AREA FOR file-1 file-2 file-3.

If SAME RECORD AREA is coded, then the buffer is not shared but only the record area is shared. So more than one file can be in open state. We should be careful while filling in the record area of the output file. This may destroy the record read most recently.

Syntax:

SAME RECORD AREA FOR file-1 file-2 file-3.

SAME SORT AREA allows more than one sort/merge work files to use the same area. The sort work files are automatically allocated when file is opened and de-allocated when file is closed. As the sort file is automatically opened and closed during a SORT and two sort files cannot be opened at a time, this clause may not be useful.

Syntax:

SAME SORT|SORT-MERGE AREA for file-1 file-2. File-1 or file-2 should be a SD file.

I-O CONTROL- RERUN Clause

RERUN ON rescue FOR EVERY integer RECORDS on file-1
This will cause checkpoint to be taken for every interger-1 records processing of file-1. If the program ABENDED before the complete processing of the file-1, then the program will restart from integer+1ST record instead of first record. The rescue file details should be mentioned outside the program and it varies from installation to installation.

ENTRY statement

ENTRY statement establishes an alternate ENTRY point in a COBOL called sub-program. When a CALL statement naming the alternate entry point is executed in a calling program, control is transferred to the next executable statement following the entry statement. Except when a CALL statement refers to an entry name, the ENTRY statements are ignored at run-time.

Matching Logic

If you have been given two files of similar type, say master and transaction file and you are requested to update the master file with transaction file information for existing records and prepare a report of new transactions and deleted transactions, then you should go for what is called Matching logic. This is also known as co-sequential processing.
Sort both the files on key and compare the keys. If the keys are matching then update the file. If you find any record that is found in transaction but not in master file, then that is new addition and the reverse is deletion. If the master key is greater than transaction key, then that corresponds to the first case and reverse is the second case.
This can be easily done in JCL using ICETOOL.

FILE STATUS CODES

File Status	Result	Explanation
2	VALID DUPLICATE ALTERNATE KEY DETECTED.	N/A
5	ATTEMPT TO OPEN A FILE THAT IS NOT AVAILABLE.	N/A
7	INCONSISTENCY IN STORAGE DEVICE.	N/A
10	END OF FILE REACHED	A sequential READ statement was attempted an no next logical record existed in the file because end of the file had been reached, or the first REA was attempted on an optional input file that was present.
14	INVALID READ ATTEMPT ON RELATIVE FILE. THE GIVEN RRN NUMBER IS LARGER THAN THE RELATIVE KEY.	A sequential READ statement was attempted for relative file and the number of significant digits ir the relative record number was larger than the s of the relative key data item described for the file
16	A READ WAS ATTEMPTED WHEN AT THE END CONDITION IS TRUE.	N/A
20	INVALID KEY.	N/A
21	RECORD OUT OF SEQUENCE	A sequence error exists for a sequentially acces indexed file. The prime record key value has bee changed by the program between the successfu execution of a READ statement and the executic of the next REWRITE statement for that file, or t ascending requirements for successive record k values were violated.
22	DUPLICATE KEY	An attempt was made to write a record that woul create a duplicate key in a relative file; or an attempt was made to write or rewrite a record th would create a duplicate prime record key or a duplicate alternate record key without the DUPLICATES phrase in an indexed file. This key value applies to an indexed file in which the alternate key has been declared 'UNIQUE'.
23	RECORD NOT FOUND	An attempt was made to randomly access a reco that does not exist in the file, or a START or random READ statement was attempted on an optional input file that was not present.
24	NO MORE SPACE ALLOCATED TO FILE. WRITING BEYOND THE ALLOCATION OF THE FILE.	An attempt was made to write beyond the extern defined boundaries of a relative or indexed file. (a sequential WRITE statement was attempted fo relative file and the number of significant digits ir the relative record number was larger than the s of the relative key data item described for the file
30	UNCORRECTABLE I/O ERROR.	No further information
34	AN ATTEMPT WAS MADE TO WRITE BEYOND THE BOUNDARY OF THE FILE.	A permanent error exists because of a boundary violation; an attempt was made to write beyond t externally-defined boundaries of a sequential file

35	ATEEMPTING TO OPEN AN EMPTY FILE IN INPUT OR I/O MODE. DD NAME IS MISSING OR WRONGLY GIVEN.	An OPEN statement with the INPUT, I-O, or EXTEND phrase was attempted on a non-optional file that was not present.
37	OPEN WAS ATTEMPTED ON A FILE THAT WILL NOT SUPPORT THE OPEN MODE SPECIFIED.	An OPEN statement was attempted on a file that would not support the open mode specified in the OPEN statement. Possible violations are: 1. The EXTEND or OUTPUT phrase was specified but the file would not support write operations. 2. The I-O phrase was specified but the file would not support the input and output operations permitted. 3. The INPUT phrase was specified but the file would not support read operations.
37	ATTEMPT TO OPEN A FILE THAT HAS BEEN CLOSED WITH LOCK	An OPEN statement was attempted on a file previously closed with lock.
39	ERROR DURING OPENING.INCONSISTENCY BETWEEN FILE DESC AND ACTUAL FILE	The OPEN statement was unsuccessful because a conflict was detected between the fixed file attributes and the attributes specified for that file in the program. These attributes include the organization of the file (sequential, relative, or indexed), the prime record key, the alternate record keys, the code set, the maximum record size, and the record type (fixed or variable).
40	RECORD IS INCONSISTENT WITH THE DESCRIPTION OF THE RECORD THAT HAS BEEN READ	N/A
41	ATTEMPT TO OPEN A FILE WHICH IS ALREADY OPENED	An OPEN statement was attempted for a file in the open mode.
42	ATTEMPT TO CLOSE A FILE THAT IS NOT OPEN.	A CLOSE statement was attempted for a file not in the open mode.
43	A READ DID NOT PRECEDE EXECUTION OF CURRENT REWRITE COMMAND.	For a mass storage file in the sequential access mode, the last input-output statement executed for the associated file prior to the execution of a REWRITE statement was not a successfully executed READ statement. For relative and indexed files in the sequential access mode, the last input-output statement executed for the file prior to the execution of a DELETE or REWRITE statement was not a successfully executed READ statement.
44	A BOUNADRY VIOLATION DUE TO ATTEMPT TO WRITE A RECORD OF IMPROPER LENGTH.	A boundary violation exists because an attempt was made to rewrite a record to a file and the record was not the same size as the record being replaced, or an attempt was made to write or rewrite a record that was larger than the largest or smaller than the smallest record allowed by the RECORD IS VARYING clause of the associated file-name.
46	ATTEMPT TO READ THE NEXT NONEXISTENT RECORD.	A sequential READ statement was attempted on a file open in the input or I-O mode and no valid next record had been established because: 1. The preceding START statement was

		unsuccessful, or 2.The preceding READ statement was unsuccessful but did not cause an at end condition, or 3. The preceding READ statement caused an at end condition.
47	ATTEMPT TO READ FROM A FILE WHICH IS NOT OPEN IN INPUT OR I-O MODE.	The execution of a READ statement was attempted on a file not open in the input or I-O mode.
48	ATTEMPT TO WRITE ON A FILE NOT OPEN IN OUTPUT OR EXTEND MODE.	The execution of a WRITE statement was attempted on a file not open in the I-O, output, or extend mode.
49	ATTEMPT TO WRITE ON A FILE NOT OPEN IN I-O MODE.	The execution of a DELETE or REWRITE statement was attempted on a file not open in the I-O mode.
88	VSAM START FAILURE.	N/A
90	SOME TYPE OF VSAM LOGIC ERROR.	No further information.
91	PASSWORD FAILURE.	For VSAM only: Password failure.
92	DOING IN WRONG MODE.	Logic error.
93	NOT ENOUGH VIRTUAL STORAGE FOR VSAM TASK.	Resource not available.
94	NO CURRENT POINTER FOR SEQUENTIAL PROCESSING.	For VSAM with CMPR2 compiler-option only:No file position indicator for sequential request.
95	CONFLICTING ATTRIBUTES.	For VSAM only: Invalid or incomplete file information.
96	NO CORRESPONDING DD STATEMENT FOR THE FILE.	For VSAM only: No DD statement specified for this file.
97	FILE NOT CLOSED BY PREVIOUS JOB.	For VSAM only: OPEN statement execution successful: File integrity verified.

Cobol Coding

	Good COBOL Coding practices for better Modularity and Performance.
1.	Do not use 66 level. Planning of data definition should make this redundant
2.	Do not use 77 level. Collect all these together and group under a 01 level.
3.	Use 88 level appropriately. For conditional names.
4.	All level numbers increment by an odd number in order to allow for the insertion of extra le during modifications; this reduces the need for wholesale renumbering of the levels with all problems that this entails. 01 LEVEL-1. 05 LEVEL-2. 10 LEVEL-3 PIC 9(02) COMP-3 VALUE ZEROES.
5.	PARAGRAPHs or SECTIONs should be used; they should not be mixed except when the PARAGRAPH names are used to subdivide the SECTION. If this is used then only the SE should be performed.
6.	It corresponds to the sequence/logical position in the structure of the code. This enables th location to be easily ascertained in a listing and also allows the user to see where the prog has progressed to in the event of any problems.
7.	The SORT and MERGE statements should be avoided if at all possible, it is more efficient an external sort, e.g. SYNCSORT/DFSORT, rather than calling the internal sort which also one of these sorts. This is because with an internal sort, control is passed back to the prog after each record reducing the speed of the sort. It is also easier to debug a COBOL progra it is to debug the dump from an internal sort abend. If SORT routines are used then the ma key length is 4092 bytes with no maximum number of keys. To increase the efficiency of th keep the key fields at the start of the record and as contiguous as possible.
8.	GOTO should never be used; a structured program should not need to use this command. however because of coding or processing constraints, then the only possible use is to tran control to the end of the SECTION or PARAGRAPH being processed. It MUST NOT be us any other case.
9.	Nested IF statements should be avoided and consideration should be given to the use of EVALUATE as an alternative. The other option is to restructure the logic of the program so the if statement is either eliminated or simplified. The WHEN OTHER must always be code this is not present then if none of the prior conditions are satisfied then the program will fall through with no processing occurring. The WHEN OTHER statement should always be the option; if it is not then all conditions following are irrelevant as they will not be tested.

10.	Care should be taken when using STRING and UNSTRING as this may hide data structures. When using these operations then the ON OVERFLOW clause should always be used; if this is not used then you are not notified of the incomplete operation and control passes to the next sequential statement.
11.	All structures are terminating with END-XXX.
12.	The READ and WRITE PARAGRAPH/SECTIONS should be grouped together
13.	For each file there should be a separate READ PARAGRAPH/SECTION with an 'AT END' clause; this should be performed in all cases and the AT-END condition tested for sequential files or the return code. For each file there should be a separate READ section. This is to reduce the size of the load module as the machine instructions generated for the execution of a perform are less than that for a READ; it also means that the file is only read in the one place rather than it scattered throughout the program, easing maintenance.
14.	An initial value of spaces or zero can be given to an item either in the DATA DIVISION or the PROCEDURE DIVISION. In the latter division then there are two options for doing this, to move an initial value to the individual items or to use in COBOL II onwards, the INITIALIZE statement. The use of the VALUE field of the data division should be used to give an initial value to constants.
15.	For variables that are used in working storage the INITIALIZE statement should be used. The INITIALIZE statement is limited in that it will only move spaces (to alphanumeric) or zeros (to numeric) depending on the item definition. All numeric types are supported for this statement, zoned, packed, binary. If any other value is required as an initial value then this must be moved separately, it is still the case that this should be done in the initialization PARAGRAPH or SECTION that is performed prior to processing and at any other time as required.
16.	How the table has been defined in the DATA DIVISION controls the access method used to retrieve the data. If an index has been defined then the method used is 'SEARCH ALL' if the data is in key sequence (and BATCH), else use SEARCH; if there are more than 10 entries, and the table will be heavily accessed consider sorting the data; if no index is present then use PERFORM VARYING to retrieve the data.
17.	An index is more efficient than a subscript; this is because a factor of the element size for the table is built into the index, ensuring that the displacement from the start of the table is known for the element and does not have to be calculated at run time as it does with subscripts. This increases the speed at which the elements can be accessed when an index is used. If a table is to be searched sequentially, then if possible put the occurrences that are more likely to satisfy the requirements when a table is checked at the start. If the table is to be searched using a binary search, then the items must be stored in ASCENDING/DESCENDING key sequence.
18.	Static call should be used when size is not an issue but speed is, or the program is called often. Or else use Dynamic call.

19.	The following keywords must not be used: NEXT SENTENCE ADD CORR, MOVE CORR MULTIPLY GO TO

JCL(Job Control Language)

1. WHAT IS JCL?

1.1 JOB CONTROL LANGUAGE consists of control statements that:

- introduce a computer job to the operating system
- request hardware devices
- direct the operating system on what is to be done in terms of running applications and scheduling resources

JCL is not used to write computer programs. Instead it is most concerned with input/output---telling the operating system everything it needs to know about the input/output requirements. It provides the means of communicating between an application program and the operating system and computer hardware.

1.2 IS JCL DIFFICULT? ... NOT NECESSARILY!

The role of JCL sounds complex and it is---JCL can be downright difficult.

JCL can be difficult because of the way it is used. A normal programming language, however difficult, soon becomes familiar through constant usage. This contrasts with JCL in which language features are used so infrequently that many never become familiar.

JCL can be difficult because of its design - JCL:

- consists of individual parameters, each of which has an effect that may take pages to describe
- has few defaults--must be told exactly what to do
- requires specific placement of commas and blanks
- is very unforgiving--one error may prevent execution

JCL is not necessarily difficult because most users only use a small set of similar JCL that never changes from job to job.

1.3 HOW DO YOU SEND INFORMATION TO THE COMPUTER?

BATCH PROCESSING VS. INTERACTIVE PROCESSING

Interactive Processing means that you give the computer a command and the computer responds to your command request. It is more like a conversation.

Batch Processing means that you give the computer a whole group of commands, usually in the form of some sort of program you have written, and have the computer process this group of commands. It is more like writing a letter.

2. BASIC SYNTAX OF JCL STATEMENTS

```
//NAME    OPERATION    OPERAND,OPERAND,OPERAND    COMMENTS

   |          |                  |                    |
  name    operation         operand field          comment
  field     field                                   field
```

name field - identifies the statement so that other statements or the system can refer to it. The name field must begin immediately after the second slash. It can range from 1 to 8 characters in length, and can contain any alphanumeric (A to Z) or national (@ $ #) characters.
operation field - specifies the type of statement: JOB, EXEC, DD, or an operand command.
operand field - contains parameters separated by commas. Parameters are composites of prescribed words (keywords) and variables for which information must be substituted.
comments field - optional. Comments can be extended through column 80, and can only be coded if there is an operand field.

General JCL Rules:

- Must begin with // (except for the /* statement) in columns 1 and 2
- Is case-sensitive (lower-case is just not permitted)
- NAME field is optional
- must begin in column 3 if used
- must code one or more blanks if omitted
- OPERATION field must begin on or before column 16
- OPERATION field stands alone
- OPERANDS must end before column 72
- OPERANDS are separated by commas
- All fields, except for the operands, must be separated by one blank.

2.1 CONTINUATION OF JCL STATEMENTS

```
//LABEL  OPERATION  OPERAND, OPERAND,
//   OPERAND,OPERAND,
//  OPERAND,
//     OPERAND
```

When the total length of the fields on a control statement exceeds 71 columns, continue the fields onto one or more following statements.

- Interrupt the field after a complete operand (including the comma that follows it) at or before column 71
- Code // in columns 1 and 2 of the following line
- Continue the interrupted statement beginning anywhere in columns 4 to 16.

2.2 COMMENTING JCL

//* THIS IS A COMMENT LINE

JCL should be commented as you would any programming language. The comments statement contains //* in columns 1 to 3, with the remaining columns containing any desired comments. They can be placed before or after any JCL statements following the JOB statement to help document the JCL. Comments can also be coded on any JCL statement by leaving a blank field after the operand field.

3. THREE TYPES OF JCL STATEMENTS

JOB Identifies the beginning of a job

EXEC Indicates what work is to be done

DD Data Definition, i.e., Identifies what resources are needed and where to find them

4. THE JOB STATEMENT

The JOB statement informs the operating system of the start of a job, gives the necessary accounting information, and supplies run parameters. Each job must begin with a single JOB statement.//jobname JOB USER=userid

jobname - a descriptive name assigned to the job by the user which is the banner on your printout
- any name from 1 to 8 alphanumeric (A-Z,0-9) or national ($,@,#) characters
- first character must be alphabetic or national

JOB - indicates the beginning of a job

Userid - a 1 to 7 character user identification assigned to access the system

4.1 ADDITIONAL OPERANDS OF THE JOB STATEMENT

//jobname JOB USER=userid, TIME=m, MSGCLASS=class, NOTIFY=userid

USER=userid Identifies to the system the user executing the job
TIME=m Total machine (m)inutes allowed for a job to execute
MSGCLASS=class Output class for the job log
NOTIFY=userid User to receive a TSO message upon completion of a job

4.1.1 MSGCLASS

The MSGCLASS parameter allows you to specify the output class to which the operating system MVS is to write the job log or job entry subsystem (JES) messages. If you do not code the MSGCLASS parameter, MSGCLASS=J is the default and will be used. MSGCLASS=J indicates the output will be printed on 8 1/2" by 11" hole paper.

This includes the following:

1. Job Entry Subsystem (JES) Messages
2. Error Messages
3. JCL Statements
4. Dataset Dispositions
5. Accounting information

Here is a list of available output classes:

- A greenbar paper
- 5-9 TSO held output
- C-Z IBM page printer output classes. See chart in section "Sys-out Classes for IBM Print" on page 28.

5. THE EXEC STATEMENT

Use the EXEC (execute) statement to identify the application program or cataloged or in-stream procedure that this job is to execute and to tell the system how to process the job.

```
//stepname EXEC procedure,REGION=####K
```

or

```
//stepname EXEC PGM=program,REGION=####K
```

stepname - an optional 1 to 8 character word used to identify the step
EXEC - indicates that you want to invoke a program or cataloged procedure
procedure - name of the cataloged procedure to be executed
program - name of the program to be executed
REGION=####K - amount of storage to allocate to the job

5.1 PROGRAMS AND CATALOGED PROCEDURES

```
// EXEC PGM=pgmname
```

A program referred to on the EXEC PGM= statement is a compiled and linked version of a set of source language statements that are ready to be executed to perform a designed task. It is also known as an executable load module. It must reside in a partitioned dataset.

```
// EXEC cataloged-procedure-name
```

Because the same set of JCL statements are often used repeatedly with little or no change they can be stored in cataloged procedures. JCL provides programmers with the option of coding these statements only once, recording and cataloging the statements under an appropriate name in a procedure library, and then invoking these statements through an EXEC statement. Such a previously established set of JCL statements is known as a "cataloged procedure." The effect of using a cataloged procedure is the same as if the JCL statements in the procedure appeared directly in the input stream in place of the EXEC statement calling the procedure. This saves the user from writing lengthy error-prone JCL statements. In short, this is JCL that does not have to be included in batch jobs.

* Note that within cataloged procedures a program will be executed.

5.1.1 Modifying Cataloged Procedures

Additional statements may be added to the JCL comprising a cataloged procedure at the time of invocation. Also, operand values on existing JCL statements may be altered or parameters defined in the procedure may be substituted at invocation time.

The value in the override statement replaces the value for the same parameter in the cataloged procedure. Cataloged procedure statements must be overridden in the same order as they appear in the procedure.

```
//procstepname.ddname DD parameter=value
```

You can modify cataloged procedures by:

- Overriding parameters in an existing DD statement
- Adding DD statements to the procedure. To add new DD statements let them follow any changed DD cards for that step.

To modify an existing DD statement, only those operands to be changed need be coded on the modifying DD statement. The remaining operands of the DD statement within the procedure will be unchanged. If more than one DD statement in a procedure is to be modified, the modifying DD statements must be placed in the same order as the original DD statements occur in the procedure.

To add a DD statement to an existing procedure, place the DD statement after the procdure invocation EXEC statement and any modified DD statements within the job step.

```
// EXEC SAS
//NEWDD DD DSN=user.file,DISP=SHR
```

When looking at your output, the following symbols will determine what kind of statement is indicated when your job runs:

// Indicates JCL statements

XX Indicates cataloged procedure statements

X/ Indicates a modified cataloged procedures statement

6. DATA DEFINITION (DD) STATEMENT

A DD (Data Definition) statement must be included after the EXEC statement for each data set used in the step. The DD statement gives the data set name, I/O unit, perhaps a specific volume to use, and the data set disposition. The system ensures that requested I/O devices can be allocated to the job before execution is allowed to begin.

The DD statement may also give the system various information about the data set: its organization, record length, blocking, and so on.

//ddname DD operand,operand,etc.

ddname - a 1 to 8 character name given to the DD statement

DD - DD statement identifier

operand - parameters used to define the input or output dataset

The DD Statement

- appears after an EXEC statement
- gives the system information on many things, including the dataset attributes, the disposition of the dataset when the job completes, and which input/output device(s) to use

6.1 DD STATEMENT FOR INSTREAM DATA

Instream data is perhaps the most common form of input. To include data in the input stream, code:

//ddname DD *
.
.
.
/* (to specify end of data)

SYSIN is often used as a ddname for instream data. The /* marks the end of the data.

6.2 EXAMPLE PROGRAMS WITH INSTREAM DD STATEMENTS

6.2.1 Fortran Example

In this example, the userid UGUSER is executing a Fortran procedure with the Fortran program inserted after the FORT.SYSIN line and the data inserted after the GO.SYSIN line.

```
//FORTRUN JOB USER=UGUSER
// EXEC FORTVCLG,REGION=1500K
//FORT.SYSIN DD *
```

```
FORTRAN statements
/*
//GO.SYSIN DD *
data lines
/*
//
```

6.2.2 SAS Example

In this example, the userid UGIBM is executing a SAS procedure with the SAS program inserted after the SYSIN line.

```
//SASRUN JOB USER=UGIBM
// EXEC SAS,REGION=1500K
//SYSIN DD *
SAS statements
/*
//
```

6.3 DATA DEFINITION (DD) STATEMENT FOR DISK DATASETS

```
//ddname DD UNIT=unittype,
//              DSN=userid.name,
//              DISP=(beginning,normal-end,abnormal-end),
//              SPACE=(TRK,(primary,secondary,directory)),
//              RECFM=xx,LRECL=yy,MGMTCLAS=retainx
```

ddname - data definition name; a 1-8 character word of your choice, must begin with a letter or $, @, #

DD - DD statement identifier

UNIT = unittype - an I/O unit is a particular type of I/O device: a disk, tape, etc. UNIT=SYSDA refers to the next available disk storage device.

DSN=userid.name DSN parameter names the data set. Data sets can be temporary or nontemporary. A temporary data set is created and deleted within the job, whereas nontemporary data sets can be retained after the job completes. A data set name can contain up to 44 characters including periods.

Ex. UGIBM.DATA

mgmtclas - MGMTCLAS specifies the name of the Management Class which is a set of specifications for the way the storage occupied by the data set should be treated by SMS. Generally, this deals with how long you want to keep this data set around. UCNS has set up the following management classes:

MGMTCLAS	Days Retention	MGMTCLAS	Days Retention
RETAIN0	0 (DEFAULT)	RETAIN8	8
RETAIN1	1	RETAIN9	9
RETAIN2	2	RETAIN10	10
RETAIN3	3	RETAIN14	14
RETAIN4	4	RETAIN28	28
RETAIN5	5	RETAIN56	56
RETAIN6	6	RETAIN95	95
RETAIN7	7	STANDARD	(18 months past last use)

6.4 DATA DEFINITION (DD) STATEMENT FOR TAPE DATASETS

```
//ddname DD UNIT=unittype,VOL=SER=unitname,
//            DSN=filename,
//            DISP=(beginning,normal-end,abnormal-end),
//            DCB=(RECFM=xx,LRECL=yy,BLKSIZE=zz,DEN=density),
//            LABEL=(file#,labeltype,,mode)
```

ddname - data definition name; a 1-8 character word of your choice, must begin with a letter or $, @, #

DD - DD statement identifier

UNIT=unittype - an I/O unit is a particular type of I/O device: disk, tape, etc.
TAPE16 refers to a 1600 bpi tape drive
TAPE62 refers to a 6250 bpi tape drive
TAPECA refers to a 38K or XF catridge tape drive

VOL=SER=unitname - VOL=SER parameter is needed if the data set is to be placed on a specific tape volume. This refers to the volume serial number on an internal tape label.

DSN=filename - DSN parameter names the file on the tape. The filename can be from 1 to 17 characters in length.
Ex. COWDATA

6.4.1 Disposition (DISP) Parameters

The DISP parameter describes the current status of the data set (old, new, or to be modified) and directs the system on the disposition of the dataset (pass, keep, catalog, uncatalog, or delete) either at the end of the step or if the step abnormally terminates. DISP is always required unless the data set is created and deleted in the same step.

```
//    DISP = (beginning, normal-termination ,abnormal-termination)
              _________  _________________  ___________________
                 |           |                  |
               NEW         CATLG              DELETE
               OLD          KEEP               KEEP
               SHR          PASS              CATLG
               MOD         DELETE            UNCATLG
                          UNCATLG
```

6.4.1.1 Beginning Dispositions

This is the status of the data set at the beginning of the step. If the data set is new, the system creates a data set label; if it is old, the system locates it and reads its label

NEW creates a new data set

OLD designates an existing data set; it can be an input data set or an output data set to rewrite

SHR identical to OLD except that several jobs may read from the data set at the same time.

MOD modifies a sequential data set - positions the pointer at the end of the data set in order to add new data to the data set.

6.4.1.2 Normal Termination and Abnormal Termination Dispositions

Normal disposition, the second term in the DISP parameter, indicates the disposition of the data set when the data set is closed or when the job terminates normally. The abnormal dispositions, effective only if the step abnormally terminates, are the same as normal dispositions except that PASS is not allowed.

PASS passes the data set on to subsequent job steps, and each step can use the data set once.

KEEP keeps nontemporary data sets.

DELETE deletes data sets.

CATLG catalogs a nontemporary data set. CATLG is similar to KEEP except that the unit and volume of the data set are recorded in the catalog along with the data set name.

UNCATLG uncatalogs a data set. UNCATLG is the same as KEEP except that the data set name is removed from the catalog.

6.4.1.3 Examples of DISP parameters

// DISP=SHR	read from
// DISP=OLD	write to

// DISP=(NEW,CATLG,DELETE)	create and catalog; delete if there is a system abend
// DISP=(OLD,PASS)	dataset already exists; pass it to the next step
// DISP=MOD	write to the bottom of an existing dataset

6.4.2 SPACE Parameter

All new data sets on disk volumes must be allocated space. Storage on disk volumes can be allocated in units of blocks, cylinders, tracks, kilobytes, and bytes.

The space may be requested as a primary and a secondary amount. The primary amount is allocated when the data set is opened with a disposition of NEW. The secondary amount is allocated if the primary amount is exceeded.

The primary amount can be conservative, with the secondary amount providing a reserve. The secondary amount provides for data set growth over time.

// SPACE=(TRK,(primary,secondary,directory))

primary receive this amount of space initially

secondary receive this amount of space each time more is needed (up to 15 times)

directory reserve this amount of blocks to keep the directory of a partitioned dataset (NOT USED for a sequential dataset)
1 directory block allows for 5 members in a partitioned dataset

Total Space = (1 * primary) + (15 * secondary)

6.4.2.1 Partitioned Dataset vs. Sequential Dataset

Any named collection of data is called a data set. A partitioned dataset consists of multiple files within one data structure. A sequential dataset consists of one file within a data structure.

Partitioned Dataset

- individual members are read/manipulated without disturbing other members
- it is advisable to never write directly to a partitioned dataset in your program
- on DD statememt: DSN=userid.file(member)
- call from editor: file(member)

Sequential Dataset

- dataset must be read from top to bottom
- on DD statement: DSN=userid.file

- call from editor: file

```
PARTITIONED DATASET          SEQUENTIAL DATASET
 DSN='userid.data-set-     DSN='userid.data-set-name'
    name(member)'

    ________________           ________________
   | DIRECTORY  |             |                |
    |-----------------|        |                |
   |   MEMBER1   |            |                |
    |-----------------|        |    DATASET     |
   |   MEMBER2   |            |                |
    |-----------------|        |                |
   |      .......      |       |                |
    |-----------------|        |                |
   |   MEMBERn   |            |                |
  |--------------------------| |________________|
  |________________|
```

A partitioned dataset differs from a sequential dataset in that it has a directory of its members. Whenever you refer to a member of a partitioned dataset, you include the member name in parentheses.

6.4.2.2 DFSMS (System Managed Datasets)

On the TSO service, data sets are typically created and reside on disk volumes. A volume is a standard unit of storage. These disk volumes are referred to as DASD, which stands for Direct Access Storage Device. Each block of data on a DASD volume has a distinct location and a unique address, making it possible to find any record without extensive searching. One DASD volume can be used for many different data sets, and space on it can be reallocated and reused.

At the University of Georgia, you are now required to utilize the Data Facilities Storage Management Subsystem (DFSMS) to establish permanent data sets. The Storage Management Subsystem (SMS) is an operating environment that automates the management of storage. With SMS, users can allocate data sets more easily. The data sets allocated through the Storage Management Subsystem are called system-managed. System-managed means that the system determines data placement and automatically manages data availability, performance, space, reclamation, and security. One of the most beneficial goals of System-managed storage is to relieve users of performance, availability, space, and device management details.

DFSMS stores data in a device-independent format so that it can easily move the data to any of the following devices:

- 3490E Magnetic Catridge Tape
- DASD for models 3380 and 3390

The migration and movement of data depends on such factors as:

- Management Class
- Data set Usage

- Minimum percent free space on a DASD Volume
- Request by storage administrator or user

DFSMS records the location of each dataset it moves in a control data set. The actual migration is handled by DFHSM (Data Facilities Hierarchical Storage Manager). DFHSM is a DASD management product tool for managing low-activity and inactive data.

Data sets that have reached the end of their retention period (expired) will be deleted. Data sets with a management class of STANDARD will be deleted if the data set has not been referenced for a period of eighteen months. A notification will be sent to the user after a STANDARD data set has not been referenced for six months informing the user of the STANDARD deletion policy. At this time the data set will be moved to tape.

All data sets that have a management class of RETAIN95 or STANDARD will be automatically backed up by DFHSM. Two copies of each will be kept. The change indicator will trigger the backup after the first backup is made. A user can use the HBACK command to add non-STANDARD and non-RETAIN95 data sets to this.

6.4.2.3 SPACE Parameter (Partitioned Dataset)

// SPACE=(TRK,(primary,secondary,directory))

This SPACE example allows a total of 40 tracks for the dataset with 1 block of space reserved for the directory.

```
// SPACE=(TRK,(10,2,1))
```

Total Space = (1*10) + (15*2) = 10 + 30 = 40 tracks
Directory = (1*5) = 5 members/dataset

This SPACE example allows a total of 100 tracks for the dataset with 8 blocks reserved for the directory.

```
// SPACE=(TRK,(25,5,8))
```

Total Space = (1*25) + (15*5) = 25 + 75 = 100 tracks
Directory = (8*5) = 40 members/dataset
* Note that the track capacity for 3380 is 1 track = 47,476 characters.

6.4.2.4 SPACE Parameter (Sequential Dataset)

// SPACE=(TRK,(primary,secondary))

This SPACE example allows a total of 53 tracks for the dataset.

// SPACE=(TRK,(8,3))

Total Space = (1*8) + (15*3) = 8 + 43 = 53 tracks

This SPACE example allows a total of 520000 bytes for the dataset.

// SPACE=(80,(5000,100))

Total Space = (1*400000) + (15*8000) = 400000 + 120000 = 520000 bytes
* Note that directory blocks are always 0 for sequential datasets; therefore, the directory parameter is NOT USED for sequential datasets.

6.4.3 Data Set Attributes

With SMS, you do not need to use the DCB parameter to specify data set attributes. ALL of the DCB keyword subparameters (record length, record format, and blocksize) can be specified without the need to code DCB=.

For example, the following DD statement:

// DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)

can be specified as:

// RECFM=FB,LRECL=80

The blocksize parameter can be omitted because SMS will select the optimum blocksize.

RECFM=xx specifies the record format. The format can be one or more of the following characters:

F	fixed-length records
V	variable-length records
U	undefined-length records
FB	fixed and blocked
FBA	fixed, blocked, with ANSI carriage control characters
VB	variable and blocked
VBA	variable, blocked, with ANSI carriage control characters

LRECL=yy specifies the length of records

equal to the record length for fixed-length records

equal to the size of the largest record plus the 4 bytes describing the record's size for variable-length records

omit the LRECL for undefined records

LRECL can range from 1 to 32760 bytes

BLKSIZE=zz specifies the blocksize if you wish to block records

- must be a multiple of LRECL for fixed-length records
- must be equal to or greater than LRECL for variable-length records
- must be as large as the longest block for undefined-length records
- BLKSIZE can range from 1 to 32760 bytes

6.4.3.1 Fixed Block

// RECFM=FB,LRECL=80,BLKSIZE=9040

This dataset will have fixed length records with a length of 80. There will be 113 records of data per block.

BLKSIZE/LRECL = 9040/80 = 113 records of data per block

.4.3.2 Variable Block

// RECFM=VB,LRECL=255,BLKSIZE=3120

This dataset will have variable length records with a maximum of 255 characters. The blocksize of the dataset will be 3120.

6.4.4 DCB and LABEL Parameters for Tapes

RECFM=xx specifies the record format. The format can be one or more of the following characters:

F fixed-length records
V variable-length records
U undefined-length records
FB fixed and blocked
FBA fixed, blocked, with ANSI carriage control characters
VB variable and blocked
VBA variable, blocked, with ANSI carriage control characters

LRECL=yy specifies the length of records
equal to the record length for fixed-length records
equal to the size of the largest record plus the 4 bytes describing the record's size for variable-length records
omit the LRECL for undefined records
LRECL can range from 1 to 32760 bytes

BLKSIZE=zz specifies the blocksize if you wish to block records

must be a multiple of LRECL for fixed-length records
must be equal to or greater than LRECL for variable-length records
must be as large as the longest block for undefined-length records
BLKSIZE can range from 1 to 32760 bytes

DEN=density measures the number of bits that are stored in a unit of measurement on the tape. This measurement is commonly referred to as BPI (bits per inch).

densities are represented in the DEN parameter as follows:

2	800 BPI
3	1600 BPI
4	6250 BPI
*	38K OR XF BPI

when adding files to an existing tape all new files will be written at the same density as the first file no matter if you specify differently.

The LABEL parameter tells the type of label, the relative file number on the tape, and whether the data set is to be protected for input or output.

// LABEL=(file#,labeltype,,mode)

file - the relative file number on the tape (1-4 digits)

type - the type of label on the tape

NL	No Label
SL	Standard Label
AL	American National Standard Label
BLP	Bypass Label Processing

mode IN/OUT parameter

IN protects the file from being opened for output
OUT protects it from being opened for input

// LABEL=(3,SL,,IN) File 3 on a SL tape can only be read

// LABEL=(1,NL,,OUT) File 1 on a NL tape is open for output

6.5 EXAMPLE PROGRAMS WITH DD STATEMENTS

6.5.1 Fortran Job for Reading from Disk Dataset

In this example, the userid UGIBM is executing a Fortran procedure with the Fortran program inserted after the FORT.SYSIN line, and the data is being read by UNIT 3 from the data set UGIBM.FOOD.

```
//FORTRUN JOB USER=UGIBM
//     EXEC FORTVCLG,REGION=2000K
//FORT.SYSIN DD *

        read(3,10) x,y
    10 format(1x,f4.1,1x,f4.1)

/*
//GO.FT03F001 DD DSN=UGIBM.FOOD,UNIT=SYSDA,DISP=SHR
/*
//
```

6.5.2 SAS Job for Reading from Disk Dataset

In this example, the userid UGIBM is executing a SAS procedure with the SAS program inserted after the SYSIN line and the data is being read from the data set UGIBM.DATA.

```
//SASRUN JOB USER=UGIBM
//     EXEC SAS,REGION=2000K
//OLDDATA DD DSN=UGIBM.DATA,UNIT=SYSDA,DISP=SHR
//SYSIN DD *

data one; infile olddata;
input x y z;

/*
//
```

6.5.3 Fortran Job for Writing to Disk Dataset

In this example, the userid UGABC is executing a Fortran procedure with the Fortran program inserted after the FORT.SYSIN line, and the results are written by UNIT 8 to the data set UGABC.NEWFILE. This dataset will only be retained for 7 days. If the user decides to keep the dataset for a longer period of time, the ALTER command can be issued. For example, ALTER 'UGABC.NEWFILE' MGMTCLAS(RETAIN14), will keep the dataset around for another week. Since the BLKSIZE is not specified, SMS will determine the most efficient blocksize.

```
//FORTRUN JOB USER=UGABC
// EXEC FORTVCG,REGION=2000K
//FORT.SYSIN DD *

       write(8,10) x,y
10    format(1x,f4.1,1x,f4.1)

/*
//GO.FT08F001 DD DSN=UGABC.NEWFILE,UNIT=SYSDA,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(40,10),RLSE),
// RECFM=FB,LRECL=80,MGMTCLAS=RETAIN7
/*
//
```

6.5.4 SAS Job for Writing to Disk Dataset

In this example, the userid UGXYZ is executing a SAS procedure with the SAS program inserted after the SYSIN line and the results are written to the data set UGXYZ.SAS.DATA. This dataset, based on the management class of standard, will be retained on the system as long as the user utilizes it.

```
//SASRUN JOB USER=UGXYZ
// EXEC SAS,REGION=2000K
//NEWDATA DD UNIT=SYSDA,
// DSN=UGXYZ.SAS.DATA,DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(40,10),RLSE),
// RECFM=FB,LRECL=80,MGMTCLAS=STANDARD
//SYSIN DD *

   data example;
   input x y z;
   cards;
   1 2 3
   4 5 6
   ;
   data _null_; set example; file newdata;
   put x y z;
   return;

/*
//
```

7. DD STATEMENT FOR PRINTED OUTPUT

The SYSOUT parameter provides a convenient means of routing output to printers or other devices.

//ddname DD SYSOUT=class

ddname - a 1 to 8 character name given to the DD statement

DD - DD statement identifier

SYSOUT=class - defines this data set as a system output data set, usually called a sysout data set and assigns this sysout data set to an output class. Here is a list of available output classes:

- A greenbar paper
- 5-9 TSO held output
- C-Z IBM page printer output classes. See chart in section "Sysout Classes for IBM Print" on page 28.

* same class as specified on the MSGCLASS parameter

* Note that if a sysout data set has the same class as the MSGCLASS parameter, the job log appears on the same output listing as this sysout data set.

7.1 SYSOUT CLASSES FOR IBM PRINT

SYSOUT* CLASS	PAPER	ORIENTATION	SIDES	IMAGES	CHAR/LINE	LINES/IMAGE
C	hole	landscape	1	1	132	62
D	hole	portrait	1	1	80	62
F	bond	landscape	2	1	132	62
G	bond	portrait	2	1	80	62
H	bond	landscape	2	2	90	80
I	bond	portrait	2	1	80	70
J	hole	landscape	2	1	132	62
K	hole	portrait	2	1	80	62
Q	bond	landscape	1	1	132	62
S	bond	portrait	1	1	80	62
W	hole	portrait	2	2	132	62
X	bond	portrait	2	2	132	62
Y	bond	landscape	2	1	132	62
Z	hole	landscape	2	1	132	62

Graybar Overlay: The paper used for classes C,F,H,J, and Q is shaded to mimic standard greenbar paper.

Service Site Page Printers: To route your output to one of the page printers in the Journalism or Aderhold sites, you must use a SYSOUT class or message class (MSGCLASS) with a paper type of "hole" (C, D, J, K, W, or Z).

Line Printer SYSOUT Class: To route your output to a line printer, use a SYSOUT class or message class (MSGCLASS) of A. Your output will be printed on greenbar paper.

Special Hold Classes: To route your output to a hold queue, use a SYSOUT class or message class (MSGCLASS) of 5, 6, 7, 8, or 9.
----

* SYSOUT class values may be used for message class (MSGCLASS).

8. OUTPUT JCL STATEMENT

The OUTPUT statement is used to specify processing options for a system output data set. These options are used only when the OUTPUT statement is explicitly or implicitly referenced by a sysout DD statement.

```
//name OUTPUT FORMDEF=fdef,PAGEDEF=pdef,CHARS=ch,FORMS=form,
// COPIES=n,DEST=dest,DEFAULT=dd
```

name - a 1 to 8 character name given to the OUTPUT statement

OUTPUT - OUTPUT statement identifier

FORMDEF=fdef - specifies whether to use duplex or simplex and overlay

DUP1	duplex, no overlay
DUP2	duplex, grey bar overlay
SMP1	simplex, no overlay
SMP2	simplex, grey bar overlay

PAGEDEF=pdef - specifies the logical page length and width, fonts, lines within a page, and multiple logical pages on a physical page

POR1	portrait, 62 lines at 6 lines per inch
POR2	portrait, 70 lines at 6 lines per inch
POR3	portrait, 2 up format, 62 lines per frame
LAN1	landscape, 62 lines at 8 lines per inch
LAN2	landscape, 2 up format, 80 lines per frame

CHARS=ch - names the character set

GT12	Gothic font at 12 characters per inch
GT13	Gothic font at 13 characters per inch
GT18	Gothic font at 18 characters per inch

FORMS=form - specifies the type of form

BOND	plain bond paper
HOLE	3 hole drilled bond paper

COPIES=n - specifies how many copies of the sysout data set are to be printed

DEST=dest - specifies a printer destination for the sysout data set

LOCAL	Boyd Graduate Studies
SSS02	Journalism Building
SSS03	Aderhold Building
NCT19	Brooks Hall

DEFAULT=dd - specifies that this OUTPUT statement can or cannot be referenced by a sysout DD statement

YES specifies that this is the default OUTPUT statement for for all print files within a job
NO specifies that this is not the default OUTPUT statement

* Note if you are going to HOLD a print dataset with an OUTPUT statement associated with it, you will have to fully specify all of the parameters on the OUTPUT statement. Contact the UCNS Helpdesk for more details.

8.1 EXAMPLES OF THE OUTPUT STATEMENT

To request that the "gray bar" overlay not print on a landscape sysout class for all print files, code:

```
//OUT1 OUTPUT FORMDEF=DUP1,DEFAULT=YES
```

To request simplex (one sided) printing for a duplex sysout class for all print files, code:

```
//OUT1 OUTPUT FORMDEF=SMP1,DEFAULT=YES
```

To request multiple copies of a print file, code:

```
//OUT1 OUTPUT COPIES=12
```

- with -

```
//ddname DD SYSOUT=F,OUTPUT=*.OUT1
```

9. JES3 CONTROL STATEMENTS

9.1 //*MAIN STATEMENT

The //*MAIN statement is used to define the processor requirements for the current job. It specifies what time the job will be executed, how many lines in the job, and where the job is to be printed.

//*MAIN CLASS=x,LINES=y,ORG=UGAIBM1.org

CLASS=x specifies the job class for this job

B	Batch (default) Anytime daily
NITE	6:00 PM until 7:00 AM daily

WEEKEND 6:00 PM Friday until 7:00 AM Monday
Z 1:30 AM until 7:00 AM daily, weekends, holidays
BL 6:00 PM until 7:00 AM daily, more than 8 MEG region
ZL 1:30 AM until 7:00 AM daily, more than 8 MEG region

LINES specifies the maximum number of lines of data to be printed from this job in multiples of a thousand (default = 5000)

ORG=UGAIBM1.org specifies a printer destination for the sysout dataset
LOCAL Boyd Graduate Studies
SSS02 Journalism Building
SSS03 Aderhold Building
NCT19 Brooks Hall

9.2 JOB SCHEDULING SPECIFICATIONS

Since TSO is a time sharing operating system, it allows many people to use the computer at the same time in such a way that each is unaware that the computer is being used by others. Time sharing attempts to to maximize an individuals use of the computer, not the efficiency of the computer itself. In order to do this, job scheduling is used to assign jobs to a certain class in order to maximize the resources available to each user.

```
                    Priority
                    -------
         ___________
BATCH  |             4
       |
       |__________  3
              ____
       NITE  |____  2
```

Any job can be scheduled to run at night or on the weekend by coding:

//*MAIN CLASS=NITE

or

//*MAIN CLASS=WEEKEND

9.2.1 Table of Resource Limitations for Job Scheduling

Priority	CPU Time (seconds)	Region (K)	Estimated Lines	Setups Required
6	0-30	0-2048	0-5000	0
4	32-120	2049-3072	5001-10000	0

3	121-300	3073-4096	10001-40000	1-3
2	301+	4097+	40000+	4+

9.3 //*OPERATOR STATEMENT

The //*OPERATOR statement is primarily used to issue a message to the operator requesting that the tape with the specific VRN, VSN and KEYWORD is to be mounted for the job.

```
//*OPERATOR VRN=#9999 VSN=U9999 KEY=HELP
```

VRN will be assigned when the tape is checked in. This identifies the tape in the tape library. Cartridge Tapes must have #C in the first two column positions on the vrn parameter.

VSN the volume serial number on the the internal label of a standard labeled tape. This must be the actual internal label. For non-labeled tapes, this can be any arbitrary name. In both cases, the VSN must match the VOL=SER parameter on the DD card.

KEY is a password that you will assign to the tape for security purposes. The operator will check the keyword in your JCL against the keyword on the tape before mounting the tape.

The //*Operator statement in the JCL stream is placed after the JOB or //*MAIN statement and before the EXEC statement. Each //*Operator statement should be referenced by a DD statement to read or write a file to the tape.

The following example illustrates the operator card with the referencing DD statement. Note that the VSN and the VOL=SER must be the same. In this example, the user is going to read the first file on a standard labeled (SL) tape with the name COWDATA. The tape is going to be read on a 6250 BPI tape drive.

```
//SEMINAR JOB USER=USERID
//*MAIN LINES=20
//*OPERATOR VRN=#1111 VSN=0211ZZ KEY=KFJD
//STEP1 EXEC ...
//INFILE   DD   UNIT=TAPE62,VOL=SER=0211ZZ,
//             LABEL=(1,SL,,IN),
//             DSN=COWDATA,
//             DISP=(OLD,PASS)
```

10. EXAMPLE FORTRAN AND SAS PROGRAMS

10.1 EXAMPLE FORTRAN PROGRAM

In this example, the userid UGA001 is executing a Fortran procedure with the Fortran program inserted after the FORT.SYSIN line and the data inserted after the GO.SYSIN line. This job will not begin executing until after 6:00PM, and the output will be held to the terminal by the MSGCLASS=6.

```
//CONVERT JOB USER=UGA001,MSGCLASS=6,NOTIFY=UGA001
//*MAIN CLASS=NITE,LINES=40,ORG=UGAIBM1.LOCAL
// EXEC FORTVCLG,REGION=2000K
//FORT.SYSIN DD *

      read(5,10) cent
   10 format(f6.2)
      fahr=(cent*9/5)+32
      write(6,20) cent,fahr
   20 format(f6.2,' cent = ',f6,2,'fahr')

/*
//GO.SYSIN DD *
100.00
/*
//
```

10.2 EXAMPLE SAS PROGRAM

In this example, the userid RESRCH is executing a SAS procedure with the SAS program inserted after the SYSIN line and the data is being read from the first file of a standard labeled cartridge tape with the file name of DATAFIL. The output will go directly to the printer with the sysout class of S.

```
//ANALYSIS JOB USER=RESRCH,NOTIFY=RESRCH,TIME=3,MSGCLASS=S
//*MAIN LINES=10
//*OPERATOR VRN=#C1234 VSN=WGCF KEY=KFJD
// EXEC SAS,REGION=2500K
//SAMPDATA DD UNIT=TAPECA,VOL=SER=WGCF,
// LABEL=(1,SL,,IN),
// DSN=DATAFIL,
// DISP=(OLD,PASS)
//SYSIN DD *

data one;
   infile sampdata;
   input a b c d;
proc print;
```

```
/*
//
```

11. SOURCES OF HELP

11.1 UCNS HELPDESK

The UCNS Helpdesk, located in the Computer Services Annex on the corner of East Campus Road and Cedar Street and on the first floor of the Boyd Graduate Studies Research Center, function as the initial contact point for all computer related needs associated with the University of Georgia computer systems. The primary purpose of the Helpdesk is to aid faculty, staff, and students with the use of our computer systems through general consulting and information requests. Assistance is provided through telephone contacts, walkins, and electronic mail.

Hours: Monday - Friday, 8:00AM - 5:00PM
Phone: (404) 542-3106

E-mail Address: HELPDESK@UGA

Internet Address: HELPDESK@ARCHES.UGA.EDU

Limited assistance is available at all staffed UCNS Computer Service Sites.

11.2 HELPFUL IDS DOCUMENTS

- Common IBM System Error Messages and Abend Codes
- The Case of the Sinister Syntax Error
- Facility Access and Services Guide

11.3 BASIC JCL REFERENCES

- IBM MVS JCL Publication No. GC28-1300 - This manual is the basic reference document on the syntax and usage of IBM JCL. This manual is available for reference at the Client Services Help Desk.
- System/370 Job Control Language by Gary Deward Brown - This reference is a standard and popular textbook for introducing IBM JCL to persons familiar with computers but not necessarily IBM 370 systems.

JCL Utilities:-

JCL (Job Control Language)

JCL or Job Control Language is used to communicate with the computer's operating system. A JOB is a unit of work the computer is to perform. A JOB STREAM consists of JCL statements, programs that are to be executed, and data that are to be processed. The data included in the job stream are called INPUT STREAM DATA

JCL statements :-

JOB statement :- The first statement in a job stream must be a job statement whose function is to identify the job to the system.

EXEC statement :- The job statement is usually followed by an exec statement. The exec statement names the program or the procedure that is to be executed.

DD statement :- Following the exec statement are the DD statements (data definition) statements. The DD statements describe the data used by the program.

The other JCL statements are delimiter, null, comment, proc & pend.

An EXEC statement and its DD statements make up a job step.

A job may consist of 255 job steps.
The job, exec & DD statements have a common format which is :-
//name operation operand comments

Job executes procedures, procedures have various steps to execute various programs, which use parameters and parameter files.

Naming conventions :-

Jobs will be held in datasets with .cntl or .jcl extension
e.g. ORSDEVP.REL#15.JCL

Procedures will be held in datasets with .proclib or .proc extension
e.g. ORSTEST.REL#15.PROCLIB

Parameters used will be held in datasets with .parm or .parmlib extensions
e.g ORSLIVE.REL#1.PARMS

Parameters :-

<u>There are two kinds of parameters in the operand field</u> :- positional and **keyword** parameters. The operating system recognizes positional parameters by their position in the operand field. Keyword parameters can be coded in any order. Positional parameters must be coded in a specific order before any keyword parameter.
<u>Consider the job statement</u> :-

Positional parameters	**Keyword parameters**
1. e.g.. accounting information & programmers name.	1. e.g.. class & msglevel.
2. accounting information must precede the programmers name.	2. MSGLEVEL=(1,1) indicates print all the input JCL & the JCL from catalogued procedures & print all allocation messages.
3. if a positional parameter /sub parameter is omitted a comma must be coded in its place.	3. TYPRUN=SCAN causes the job's JCL to be checked for syntax errors & prevents the job from executing.

Controlling step execution

The **<u>COND</u>** parameter.
The COND parameter offers a method of testing the return codes issued by the previous steps to determine whether a step will be executed. The format of a simplified version of the COND parameter is **COND =(value,operator)**. Value is a no. between **0** and **4095** and operator could be **gt**, **ge**,**eq**, **ne**, **lt**, **le**. The COND parameter causes the value entered to be tested against the return codes from the previous steps, using the operator provided. If the test condition is true,the step is skipped.If the test condition is false,the step is executed.

Symbolic parameters

Symbolic parameters offer a convenient way of changing a procedure to fit your requirement. Consider the example procedure-proglod, sysctl, sym,etc. are all symbolic parameters. Symbolic parameter names may consist of from one to seven alphanumeric or national characters preceded by an ampersand. The first character however must be alphabetic / national. Exec statement keyword parameters may not be used as names of symbolic parameters.For e.g.. & REGION. This restriction does not extend to DD statement keyword parameters.For e.g. &DSN can be used as symbolic parameter.

Linkage parameter

Linkage parameters are used in COBOL programs. Linkage parameters are passed on to the COBOL programs through JCL by coding 'parm=' parameter in the exec statement. Whenever linkage parameters have to be used, Linkage section has to be coded in the COBOL program after the data division and the procedure division should specify the use of linkage section.
//DD400S01 EXEC PGM=DD400900,PARM='&PROCTYPE&DBNME&DICTNME'

Generation Data Groups(GDG)

A **generation data group** is a collection,or group,of cataloged data sets having the same name and related to one another chronologically. Each of these datasets is called a generation data set or,simply, a **generation**.
Each generation data set is distinguished by others by the **generation number**.
The main advantage of using a GDG is that the same JCL can be reused without change.
Generation group index contains information on how many generations are to be retained and what to do when the index gets full.

Defining a generation group index using IDCAMS :-
IDCAMS is the name of the access method services utility program that performs functions vital to the virtual storage access method(VSAM).
IDCAMS requires a region of 300K, a SYSPRINT DD statement for messages and a SYSIN DD statement for control statements.
The **command** format :-**DEFINE GDG (PARMS)**.

Define GDG parameters

Parameter	Abbrv	Meaning
NAME		GDD name.
LIMIT	LIM	The no. Of generations permitted for this gdg The max is 255.
EMPTY	EMP	If empty is specified, all data sets are to beremoved from the index when the limit is reached.
NOEMPTY	NEMP	NOEMPTY is the default.
OWNER		User identification (optional).
SCRATCH	SCR	If Scratch is specified, the dataset is scratched when the dataset is removed from the index
NOSCRATCH	NSCR	NOSCRATCH is the default.
TO (date		Dataset retention period.
FOR(days)		Dataset retention period.

Creating and accessing a generation

The generation number may be relative or absolute.
If the DS name in the group is
DDSNUS.DEVTDV.DD460302,
then to access the current generation,
you code **DSN=DDSNUS.DEVTDV.DD460302(0)**.
To access the previous generation,
you code **DSN= DDSNUS.DEVTDV.DD460302(-1)**
the generation before would be
DSN= DDSNUS.DEVTDV.DD460302(-2).
If you want to create a new generation,
you code **DSN= DDSNUS.DEVTDV.DD460302(+1).**
The absolute no. used by the system is in the form **GXXXXVYY**,where **xxxx** is a **generation no.** from **0000 to 9999** and **yy** is a **version no.** from **00 to 99**.

Thus the generation specified as **DSN= DDSNUS.DEVTDV.DD460302(0)** might appear to the system, for e.g.. ,as
DDSNUS.DEVTDV.DD460302.G0006V00.
Then the data set with a relative no. of **(-1)** would be **DDSNUS.DEVTDV.DD460302.G0005V00**, **(+1)** would be **DSNUS.DEVTDV.DD460302.G0007V00.**

Libraries

Another name for a library is a partitioned data set(PDS).
Libraries can be created,accessed & modified using utility programs like **IEFBR14**, **IEBGENER**, **IEBPTPCH**, **IEBUPDTE**, **IEBCOPY**, etc.
IEFBR14, IEBGENER & IEBCOPY are the most frequently used utilities on any project.

IEFBR14

Strictly speaking it is not a utility program because it does nothing. It clears register 15 and BR(branch) 14.
It is used to create or delete a library based on the DD statement parameters.

IEBGENER

IEBGENER uses four data sets described by SYSPRINT, SYSIN, SYSUT1 & SYSUT2.
SYSPRINT is the **DDNAME** of the data set that IEBGENER uses to write messages.
SYSIN is the **DDNAME** of the data set that contains the control statements to tell IEBGENER how the input data set should be modified while it is being copied.
SYSUT1 is the **DDNAME** of the input data set that IEBGENER is to copy.
SYSUT2 is the **DDNAME** of the output data set that is to be created.

IEBCOPY

IEBCOPY is a utility program that may be used to copy one or more members from an existing PDS to a new or existing PDS, to make a backup copy of a PDS, and to reorganize a PDS in order to reclaim the unused space.

DFSORT

Sorting means putting records in a data set into a specified order.
Merging is closely related to sorting;it means combining records from two or more sorted datasets into one data set that is in the same order.
The sort program requires three pieces of information:where to find the input data,which fields in the records to do the sorting on, and where to put the sorted output data set.

Sort control statements

The most commonly used sort control statements are **SORT** and **MERGE**.
A typical control statement would look like : **SORT FIELDS=(1,10,CH,A)**.
The **syntax** for the fields parameter is **FIELDS=(position,length,format,sequence...)** or **FIELDS=(position,length,sequence...)**, **FORMAT=format**.
Position is the starting byte of the control field in the record.
Length is the length in bytes of the control field.
Format is the format of the data in the control field.
Commonly used values for format are ZD for zoned decimal, PD for packed decimal, BI for binary, AC for ASCII character, CH for EBCDIC character, AQ for EBCDIC character using alternative collating sequence.
Sequence is either A for ascending or D for descending.
<u>The include & omit statements</u> The include & omit statements are used to select the records to be included in the sort. By selecting records for the sort, you can decrease the amount of time the sort requires.
Only one include or omit statement is permitted in a sort. The syntax is **include cond=(test)** & **omit cond=(test)**
The only difference between include & omit is in the result. If test coded with include is true for a particular record,that record is included in the sort;whereas if test coded with omit is true,that record is omitted from the sort.
In either case test is coded as follows **test=(position,length,format,operator,value)** The position,length & format sub parameters specify a field in the record,just as in sort fields parameter.
The operator sub parameter may be **GT**, **GE**, **EQ**, **NE**, **LT** & **LE**.
The last sub parameter, value, may be either a field in the record-specified by its own position, length & format, or a constant.
Numeric constants are coded as simple values, such as 15 or -20. Character constants are enclosed in apostrophes and preceded by a c,as in c'dec'. Hexadecimal constants are enclosed in apostrophes and preceded by an x,as in x'f1f2c3'.
If more than one test is coded, the tests are joined by the boolean operator **AND** or **OR**..

<u>A typical include statement would look like</u> : **INCLUDE COND=(29,1,CH,EQ,C'L')**
The INREC & OUTREC statements : These statements are used to reformat the records.
The **INREC** statement is used before sort to shorten the records by eliminating unwanted fields. This results in faster sorting.
The **OUTREC** statement is used after the sort to improve readability.
<u>INREC & OUTREC have identical formats</u>: **INREC FIELDS=(nX,position,length,align,...)** **OUTREC FIELDS=(nX,position,length,align,...)**
The first sub parameter,**nX**,is optional; it specifies the no. of spaces to be inserted in the reformatted record.
The next two parameters, **position** & **length**, specify the field in the input record.
The third sub parameter, **align**, is optional; the possible values being H, F, and D specify alignment on a half-word, full-word, and double-word boundary respectively.

Merging

When you merge, you start with two or more sorted data sets and create a new data set that is in the same sorted order.
While merging you must specify more than one input data set. These data sets have the DD names **SORTINnn**, where nn may range from 01 to 16.
SKIPREC & **EQUALS** / **NOEQUALS** options are not used on the merge statement.
The sort control statements are normally coded in a parameter file.

VSAM Data sets

There are three types of VSAM data sets : key sequence data sets,entry sequence data sets, and relative record data sets. For VSAM data sets, unlike non-vsam data sets, a utility program, IDCAMS is used to perform functions like :

1. Allocating space on DASD
2. Supply description of the data set
3. Set the data set’s disposition
4. Load the data set
5. Print the data set

DEFINE CLUSTER

is the command using which all the descriptive information about the VSAM data set is supplied.

REPRO

command may be used to load data into a VSAM data set.

PRINT

command is used to obtain a listing of a VSAM data set.

FILE-AID

utility from ISPF primary menu(v.1) can be used to allocate / manage VSAM data sets.

JCL Error Codes

S001	An I/O error ocurred. Check reason code for exact cause. Examples are trying to read beyond End of File, trying to write to an input file or a file length error.
S002	Invalid I/O record, eg attempting to write a record that is longer than the maximum record length.
S004	Error occured during OPEN. Eg Invalid DCB.
S013	Error OPENing a dataset, eg PDS member does not exist, record length in program doesn't match dataset's record length.
S0C1	Operation Exception. Check for subscript errors, missing DD card, file not opened.
S0C4	Protection Exception/Storage Violation. Trying to access storage not available to the program. Can be caused by a subscripting error or reading/writing a file that isn't open.
S0C7	Program Check Exception - Data. Check for spaces in a packed decimal or numeric field. Check to see if record layouts or file layouts have been changed.
Sx22	Job has been cancelled. The value of x will vary depending on the way the job was cancelled. S222 means job was cancelled by a user or operator without a dump. If a TSO session times out you will probably get an S522 abend code.
S806	Unable Link or Load. The job was unable to find the specified load module. Check that the job is looking at the correct Load Libraries, specify a STEPLIB if required.
S80A	Not enough Virtual Sorage to satisfy a GETMAIN or FREEMAIN request.
S822	Unable to obtain to obtain enough space to satisfy a REGION= request May need to change REGION statement in the JCL..
S878	Not enough storage available to satisfy a GETMAIN or FREEMAIN request.. Job was anable to allocate an area of memory of the correct size. Try Specifying or amending the 'REGION=' JCL statement.
S913	You are trying to access a dataset which you are not authorized to use.
Sx37	Unable to allocate enough storage for a dataset. You might need to increase the amount of primary and secondary space to be allocated for a dataset in the 'SPACE=' parameter, or you may have to move the dataset to a different DASD devive which has enought space to store the dataset. 'x' will vary, likely Abends are SB37, SD37 or SE37.
U1020	I/O Logic error. Typical reasons are; trying to write to a file opened as input; Rewrite without a previous read. See the message IGZ020I for details of the exact reason.
U1035	Inavlid OPEN/CLOSE. Check there is a DD statement for the file. See the message IGZ035I for more detailed information.

001 I/O ERROR

002	I/O INVALID RECORD	
004	OPEN ERROR	
008	I/O SYNAD ERROR	
013	OPEN ERROR	
028	PAGING I/O ERROR	
	PROGRAM CHECK EXCEPTIONS:	
0CX	0C1	OPERATION
	0C4	PROTECTION / ADDRESSING
	0C5	ADDRESSING
706	NON-EXECUTABLE PROGRAM	
804	INSUFFICIENT VIRTUAL STORAGE	
806	UNABLE TO LOAD (LINK ETC) PROGRAM	
80A	INSUFFICIENT VIRTUAL STORAGE	
878	INSUFFICIENT VIRTUAL STORAGE	
737	I/O ERROR	
A14	I/O ERROR	
B37	INSUFFICIENT DASD SPACE	
D37	INSUFFICIENT DASD SPACE	
E37	INSUFFICIENT DASD SPACE	

VSAM Logical error codes

These codes indicate VSAM errors. They appear on the JOB log.

004(04)	Read past end of file
008(08)	You attempted to store a record with a Duplicate Key, or there is a duplicate record for an alternate index with the unique key option.
012(0C)	You attempted to store a record out of Ascending Key Sequence in Skip-Sequential Mode; record had a Duplicate Key; for Skip-Sequential processing your GET, PUT, and POINT Requests are not referencing records in Ascending Sequence; or, for Skip-Sequential Retrieval, the key requested is lower than the previous key requested. For Shared Resources, buffer pool is full.
016(10)	Record not found.
020(14)	Record already held in exclusive control by another requester.
024(18)	Record resides on a volume that cannot be mounted.
028(1C)	Data set cannot be extended because VSAM can't allocate additional Direct-Access Storage Space. Either there is not enough space left to make the secondary allocation or you attempted to increase the size of a data set while processing SHROPT=4 and DISP=SHR.
036(24)	Key Ranges were specified for the data set when it was defined but no range was specified that includes the record to be inserted.
040(28)	Insufficient Virtual Storage to complete the request. 044(2A) Work area too small. 064(40) All available strings are in use.
068(44)	You attempted to use a type of processing (Output or Control-Interval Processing) that was not specified when the data set was opened.

074(4A)	Trying to use keys on ESDS or RRDS.
076(4C)	You issued an Addressed or Control-Interval PUT to add to a Key-Sequenced data set, or issued a Control-Interval put to a Relative Record data set.
080(50)	Trying to delete from ESDS.
084(54)	Using OPTCODE=LOC for a PUT.
088(58)	You issued a Sequential GET request without having caused VSAM to be positioned for it, or you changed from Addressed Access to Keyed Access without causing VSAM to be positioned for Keyed-Sequential Retrieval; there was no Sequential PUT insert for a Relative Record data set, or you attempted an illegal switch between forward and backward processing.
92(5C)	A PUT for update or an ERASE was issued without a previous GET for update, or a PUTIX was issued without a previous GETIX.
96(60)	Changing the Prime Key or Key of Reference when making an update. 100(64 Trying to change record length.
104(68)	he RPL options are either invalid or conflicting. 108(6C) RECLEN specified was larger than the maximum allowed, equal to 0, or smaller than the sum of the length and the displacement of the key field; RECLEN was not equal to record (SLOT) size specified for a Relative Record data set.
112(70)	Invalid key length.
116(74)	Trying to update an empty dataset.
120(78)	Request was submitted by the wrong task. 132(84) An attempt was made in Locate Mode to retrieve a Spanned Record.
136(88)	You attempted an Addressed GET of a Spanned record in a Key-Sequenced data set.
140(8C)	Inconsistent Spanned record.
144(90)	Invalid pointer (no associated base record) in an Alternate Index.
148(94)	Maximum number of Alternate Index pointers exceeded.
152(98)	Not enough buffers available.
156(9C)	Invalid control interval.
192(C0)	Invalid Relative Record number in a RRDS dataset.
196(C4)	Addressed access to a Relative Record (RRDS) dataset is not allowed.
200(C8)	Addressed Access or Generic Backward processing by Key thru a path is not allowed.
204(CC)	Attempting a PUT in backward mode.
252(FC)	Record mode processing is not allowed for a Linear data set.

VSAM Open error codes

136(88)	Not enough Virtual-Storage Space is available for Work Areas, Control Blocks, or Buffers.
144(90)	An uncorrectable I/O error occurred while VSAM was Reading or Writing a catalog record.
148(94)	No record for the data set to be opened was found in the available catalog(s) or an unidentified error occurred while VSAM was searching the catalog.
152(98)	Security Verification failed; the password specified in the Access-Method Control Block for a specified level of access does not match the password in the catalog for that level of access.
164(A4)	An uncorrectable I/O error occurred while VSAM was Reading the Volume Label.
168(A8)	The data set is not available for the type of processing you specify, or an attempt was made to open a Reusable data set with the Reset option while another user had the data set.
176(B0)	An error occurred while VSAM was attempting to fix a page of Virtual storage in Real storage.
180(B4)	A VSAM catalog specified in JCL either does not exist or is not open, and no record for the data set to be opened was found in any other catalog.
184(B8)	An uncorrectable I/O error occurred while VSAM was completing an I/O request.
188(BC)	The data set indicated by the Access-Method Control Block is not of the type that may be specified by an Access-Method Control Block.
192(C0)	An unusable data set was opened for output.
232(E8)	Reset was specified for a nonreusable data set and the data set is not empty.
236(EC)	A permanent Staging error occurred in MSS (Acquire).
244(F4)	The Volume containing the Catalog Recovery area was not mounted and verified for output processing.

JCL FAQ's :-

Question: What is the parameter to be passed in the job card for the unlimited time, irrespective of the job class?

Answer: TIME=1440

Question: How do you submit JCL via a Cobol program?

Answer: Use a file //dd1 DD sysout=(*,intrdr)write your JCL to this file.

Question: Definition of COND p-r in JCL and a correction to a previously posted question

Answer: COND is a condition parameter, consists of 2 subparameters,

1st - return code from the previous step,

2nd - condition. If COND is true, the step on which COND is coded will be BYPASSED.

Question: Q) WHAT IS MEANT BY S0C7 AND S0C30 SYSTEM ABEND CODES.

Answer: A) S0C7 - Data exception error - you will get it whenever you are trying to move the low values or spaces into the numeric field, or compare the numeric fields with low values, or try to do some arithmetic operations on the low values. To avoid this you have to always initialize the numeric fields otherwise they will contain the low values.

B)S0C 30 - I have never heard of it, let you know if I come accross it.

Question: How to pass the temp dataset form one JOB step to another?

Answer: By specifying the DISP as PASS for the temp dataset

Question: What is a COND parameter in JCL?

Answer: COND means condition parameter. It is compared with system return code of previous step. //step1 exec pgm=abcd//step2 exec pgm=xyz, cond=(4,lt) step2 will be executed when system return code of step1 is less than 4.

Question: WRITE A JCL TO EXECUTE A JOB BY 7 A.M ON JAN 20,1986?

Answer: THE code IS: //*MAIN DEADLINE=(0700,B, 012086)

Question: HOW MANY TYPES OF LIBRARIES ARE THERE IN JCL?

Answer: LIBRARIES ARE OF THREE TYPES.

1.SYTEM LIBRARIES: SUCH AS SYS1.LINKLIB

2.PRIVATE LIBRARIES: SPECIFIED IN A JOBLIB OR STEPLIB DD STATEMENTS.

3.TEMPORARY LIBRARIES: CREATED IN A PREVIOUS STEP OF THE JOB.

Question: WHAT U MEANS BY INCLUDE STATEMENT IN JCL?

Answer: AN INCLUDE STATEMENT IDENTIFIES A MEMBER PF A PDS OR PDSE THAT CONTAINS.THIS SET OF JCL STATEMENTS IS CALLED AN INCLUDE GROUP.THE SYSTEM REPLACES THE INCLUDE STATEMENT WITH THE STATEMENTS IN THE INCLUDE GROUP.

Question: THE MAXIMUM NUMBER OF IN-STREAM PROCEDURE YOU CAN CODE IN ANY JCL IS?

Answer: 15.

Question: What you mean by skeleton Jell?

Answer: Jell which changes during run time i.e. the values for the jell such as pigmy name ,did name will change .i.e. same jell can be used for various job, equivalent to dynamic sql...

Question: How do you submit a JCL under CICS environment?

Answer: Edit the JCL in Extra partition TDQ and submit the same using some system command (not sure) under CICS subsystem. This is what i think, please clarify....

Question: what is JCL

Answer: It is interface between operating system (mvs) & application program. when 2 related programs are combined together on control statements is called job control language

Question: What is the max blocksize for a Tape file?

Answer: It is 32,760.Based on that we can calculate efficient number of Records in a Block

Question: What are the basic JCL Statements for a Job?

Answer: 1.JOB : Identifies a job and supplies accounting info.

2.EXEC: Identifies a job step by indicating the name of the program to be executed.

3.DD: Identifies a data set to be allocated for the job step.

4.Delimiter (/*): Marks the end of an in-stream dataset.

5.Null (//):Marks the end of a job.

6.Comments (//*): Provides Comments.

7.PROC : Marks the beginning of a procedure.

8.PEND : Marks the end of a procedure.

9.OUTPUT: Supplies options for SYSOUT processing.

Question: What does the statements: type run=scan and type run=hold doing a JCL statement

Answer: Typrun=scan checks the JCL for errors,

Typrun=hold holds the job until further notice.

Question: Which of the following is online transaction? CICS, DB2 and JCl

Answer: CICS

Question: How many PERFORM's are there in COBOL-II?

Answer: 5

Question: which is the most widely used batch performance monitor for DB2?

Answer: DB2PM

Question: What is QSAM error usually when it is occurs?

Answer: Usually it is occurs at the time of job submission.

Question: what is the purpose of include statement in a jcl?

Answer: It is used as an alternative for steplib.When we specify the dataset name in include, it will search in all the datasets specified in the include dataset.

Question: IS IT POSSIBLE TO KNOW THE REMAINING FREE SPACE IN AN CONTROL INTERVAL/CONTROL AREA ONCE AN INSERTION HAS BEEN MADE.

Answer: NOT POSSIBLE

Question: what does soc04 error mean?

Answer: This error is faced when we execute the cobol program.the main reason for this error is that a variable is defined with less characters and we are trying to move data which is larger than the actual storage space.

Question: What is JCL

Answer: JCL is Job Control Language and is used for Batch processing. The startup procedures of OS and standard products like CICS etc are written in JCL.

Question: In which table PLAN is registered in?

Answer: RCT

Question: GDG?

Answer: GDG - group of dataset that are logically or chronologically related, referred by name and a relative generation number - an integer which identifies the generation of a dataset and is coded in parentheses after dataset name. Absolute GDG name - GxxxxVyy, where xxxx-absolute gen.number, yy-version number. Can be sequential, direct, partitioned. (VSAM - no). Must always be cataloged. Advantage - all datasets have the same name and system keeps track of adding new and retaining previous generations and deleting oldest successive generation. To create a GDG we create a GDG index in the system catalog with IDCAMS utility and then a model (prototype, DSCB) on the same volume to supply DCB information. Empty - when limit is reached all members are removed from the index, otherwise-only oldest. Scratch-removed members are uncataloged & deleted, otherwise - removed & uncataloged, but remain in the system (not members of GDG any more). GDG number is updated at the end of the job. If number is not specified all generations will be processed from the beginning.

Question: what is jcl

Answer: It is used to communicate between the terminals.

Question: what do you mean By spooling? Expand SPOOL?

Answer: This is managed by JES.This is used for Queuing the Outputs that are intended for Printing and are first stored in SPOOLDASD. This can be managed Using

Question: How many Instream-Procedures(procs) can be Coded in a single Job?

Answer: The Answer is: 15

Question: FOR HOW LONG A JOB CAN BE EXECUTED CONTINUEOUSLY IN A MAINFRAME

Answer: 248 DAYS

Question: How may divisions are there in JCL-COBOL?

Answer: SIX

Question: MAX. NO OF DD STATEMENTS IN A JOB

Answer: 3273

Question: HOW MUCH SPACE OS ALLOCATES WHEN YOU CREATE A PS OR PDS?

Answer: 56 KB

Question: MIN NO OF DATASET NAMES(PDS) IN ONE DIRECTORY BLOCK?

Answer: SIX

Question: THE MAXIMUM NUMBER OF STEPS IN A JOB?

Answer: 255

Question: How much is memory space involved, when we code BLOCKSIZE, TRK & CYL

Answer: One block constitutes 32KB of formatted memory/ 42KB of Unformatted memory,6 blocks makes one Track & 15 Tracks makes one cylinder.

Question: What is DSNDB06?

Answer: This is the Place where DB2 Catalog resides.

Question: What is the use of DSNDB07 ?

Answer: This is the area where sorting takes place in DB2.

Question: What is the purpose of Identification Division?

Answer: Documentation.

Question: WHAT IS DATACOM DB?

Answer: IT IS A DATABASE USED WITH VSE.

Question: What is a Dummy Utility and what it does ?

Answer: IEFBR14 is a Dummy utility and it is used for the sake of EXEC PGM= .... Statement in JCL[when used it wouldn't perform any task]. e.g. While Allocating a dataset you don't have to run any utility [this could be done by giving disp=new inDD statement]. But for a PGM name must be given in EXEC statement, it is used.

Question: What 3 guidelines do we have to follow when concatenating DD statements?

Answer: 1. Datasets must be of the same type (disk or tape)

2. All datasets must have the same logical record length

3. The dataset with the largest block size must be listed first.

Question: On the DD statement, what is the main difference between creating a new sequential flat file and a partitioned dataset?

Answer: SPACE=(n,m) for a sequential file, SPACE=(n,m,p) for a PDS where n, m, and p are numbers. The p designates how many directory blocks to allocate.

Question: What is the difference between IEBGENER, IEBCOPY and REPRO in IDCAMS utility?

Answer: IEBGENER -- This is a dataset utility for copying sequential datasets which produces a PDS or a member from a sequential dataset.

IEBCOPY -- This is a dataset utility for copying one PDS to another or to merge PDSs.REPRO -- This is for copying sequential datasets.

Question: How do you submit JCL via a Cobol program?

Answer: Use a file //dd1 DD sysout=(*,intrdr)write your JCL to this file. Pl some on try this out.

Question: How to execute a set of JCL statements from a COBOL program

Answer: Using EXEC CICS SPOOL WRITE(var-name) END-EXEC command.var-name is a COBOL host structure containing JCL statements.

Question: What is the difference between static call & Dynamic call

Answer: In the case of Static call, the called program is a stand along program, it is an executable program . During run time we can call it in our called program. As about Dynamic call , the called program is not an executable program it can executed thru the called program

Question: What is the difference between catalouge procedure and In-Stream procedure?

Answer: In Stream procedures are set of JCL statements written between JOB and EXEC statements, start with PROC and end with PEND statement.Mainly used to test catalog procedures. Cataloged procedure is cataloged on the procedure library and is called by specifying the procedure name on the EXEC statement.

Question: What do you feel makes a good program?

Answer: A program that follows a top down approach. It is also one that other programmers or users can follow logically and is easy to read and understand.

Question: Can we browse or edit the GDG dataset if it is a tape entry?

Answer: No

Question: What are the maximum and minimum sizes of any CONTROL AREA (VSAM datasets) ?

Answer: Minimum Size : 1 track Maximum size : 1 cylinder

Question: HOW TO GET CURSOR POSITION FROM SYSTEM IN CICS ENVIRONMENT ?

Answer: GET IT FROM EIBCURPOS !

Question: How many parameters are there to a DISP statement and what are their uses.

Answer: There are three(3) parameters.

Parameter 1: current data set disposition(new, shr, old, mod)

Parameter 2: normal close action for data set (catlg, keep, delete)

Parameter 3:abend action for data set (catlg, keep, delete).

Question: What is the error code SOC01 indicate ?

Answer: Operation exception error For eg a dataset open error

Question: WHAT IS COMM?

Answer: COMM - HALF WORD BINARY

Question: What is a procedure?

Answer: A set of precoded JCL that can be modified through the use of parameters or override cards. Note: Procedures can be catalogued or instream.

Question: What is the difference between specifying DISP=OLD and DISP=SHR for a dataset?

Answer: OLD specifies exclusive use of a dataset, SHR allows multiple jobs to concurrently access the dataset Note: When updating a dataset, you would normally use OLD.

Question: What are the three basic types of statements in a job stream?

Answer: JOB(one per job stream)EXEC(one or more per job)DD(one or more per jobstep)

Question: What does SYSIN * indicate?

Answer: Instream data follows this card and is terminated when followed by a card containing // or /* in columns 1 and 2.

Question: What are three major types of JCL statements? What are their functions?

Answer: JOB, EXEC, DD. JOB - indicates start of jobstream to the operating system and through parms coded on it, certain details about the job (time, region, message level, job accounting data). EXEC - indicates the start of execution of a particular job step, be that step a program or a proc.DD - is a data definition, which is used to describe the attributes of a data set (name, unit, type, space, disposition).

Question: What happens if the Time specified is exceeded ?

Answer: The job ends with the ABEND code S322.

Question: Abbreviation of ST.

Answer: ST stands for STATUS.

Question: What is STEPLIB, JOBLIB? What is it used for? What is order of searching of the libraries in a JCL?

Answer: Specifies that the private library (or libraries) specified should be searched before the default system libraries in order to locate a program to be executed.

STEPLIB applies only to the particular step, JOBLIB to all steps in the job.First any private libraries as specified in the STEPLIB or JOBLIB, then the system libraries such as SYS1.LINKLIB. The system libraries are specified in the linklist.

Question: What is primary allocation for a data set?

Answer:The space allocated when the data set is first created.

Question: What is the difference between primary and secondary allocations for a data set?

Answer:Secondary allocation is done when more space is required than what has already been allocated.

Question: How many extents are possible for a sequential file ? For a VSAM file ?

Answer:16 extents on a volume for a sequential file and 123 for a VSAM file.

Question: What does a disposition of (NEW,CATLG,DELETE) mean?

Answer:That this is a new dataset and needs to be allocated, to CATLG the Data set if the step is successful and to delete the data set if the step abends

CICS Tutorial :-

CICS

INTRODUCTION

BATCH & ONLINE SYSTEM

ONLINE SYSTEM

DEFINITION : ONLINE processing allows a user to interact with a computer and access its resources via a terminal.
Example : Railway Reservation system.

BATCH & ONLINE SYSTEM

	BATCH	ONLINE
Input	Data from card tape, disk Batched, sequential, scheduled	Data from terminal random, concurrent
Start of A job	Operator (or operating system) initiates the job. Other jobs in the same region must wait.	Once CICS is initialized, entering transaction id triggers the transaction to start.
Processing Mode	Single task single thread. Priority in job scheduling	Multi task multi thread. Priority processing.
End of job	Each job	Each transaction. Once CICS is terminated, no transactions can be entered.
Output	printed reports, output files. User must wait for batch jobs to produce reports (day, week, month)	Message terminals updated files, system Instant feed back
Resource Usage	Less	More
Example of application	Monthly sales report	Airline reservation system

WHAT IS CICS?

- Customer Information Control System (CICS) was developed in 1960 by IBM
- ONLINE CONTROL SYSTEM
- General purpose data communication control system
- Provides services to handle all the special requirements for online processing

Note : Role of CICS is to interface between application programs and the DB/DC control system.

CICS SERVICES & THE OPERATING SYSTEM

- Requests for file I/P, O/P
- Requests for database I/P, O/P
- Requests for terminal I/P, O/P

CICS CONTROL PROGRAM AND TABLES

CICS CONTROL PROGRAM

- CICS CONTROL PROGRAM (IBM SUPPLIED)
- FCP (FILE CONTROL PROGRAM)
- JCP (JOURNAL CONTROL PROGRAM)
- KCP (TASK CONTROL PROGRAM)
- PCP (PROGRAM CONTROL PROGRAM)
- SCP (STORAGE CONTROL PROGRAM)
- TCP (TERMINAL CONTROL PROGRAM)
- TDP (TRANSIENT DATA PROGRAM)
- TSP (TEMPORARY STORAGE PROGRAM)
  OTHERS

CICS CONTROL TABLES

- CICS CONTROL TABLES (USER SPECIFIED)
- FCT (FILE CONTROL TABLE)
- JCT (JOURNAL CONTROL TABLE)
- PCT (PROGRAM CONTROL TABLE)
- PPT (PROCESSING PROGRAM TABLE)
- TCT (TERMINAL CONTROL TABLE)
- DCT (DESTINATION CONTROL TABLE)
- TST (TEMPORARY STORAGE TABLE)

CICS START UP

- CICS is submitted as a batch job.
- CICS System Initialization program (SIP) is the main job step
- SIP loads System Initialization Table (SIT)
- SIP further loads all control programs and tables
  - Perform initial housekeeping tasks

CICS SHUTDOWN

- Master terminal transaction is entered with shutdown option
- CICS job produces various logs, statistics, dumps and other reports and ends
- No transaction can be executed after that

ROLE OF CICS

- MULTI TASKING
  - More than one task can be executed concurrently.
- MULTI THREADING
  - Tasks share the same program under the multi tasking environment.
- RE-ENTRANT PROGRAM

    - Program when does not modify itself in any way during execution.
- QUASI RE-ENTRANT
    - Is a reentrant program under the CICS environment.

MAPS AND DISPLAYS

INTRODUCTION TO BMS

- To make the application program device independent and format independent CICS provides Basic Mapping Support (BMS)
- BMS is a standard facility, to deal with the formatted screen operations
- Screen defined through BMS is called a "MAP"

PHYSICAL AND SYMBOLIC MAP

Physical Map

- Primarily used by CICS
- Ensures device independence in the application program
- For input operations, it defines the maximal data length and starting position of each field to be read and allows BMS to interpret an input data stream
- For output operations it defines starting position, length, field characteristics
- (Attribute Bytes) and default data for each field, and allows BMS to construct an output data stream.
- Physical map is a program in the form of Load module
- Physical map is coded using BMS macros
- BMS macros are assembled separately and link edited into the CICS load library

SYMBOLIC MAP

1. Ensures the device and format independence to the application programs
2. A layout change in the formatted screen can be done independent of the application program coding as long as field name and length remain the same
3. Symbolic map is included in the program by issuing a COBOL COPY statement

USING MAPS IN A PROGRAM

SYMBOLIC MAP GENERATION

- Symbolic map is a copy library member
- Included in application program for defining the screen fields.
- BMS Macros are coded, assembled and catalogued into a COPY library

SYMBOLIC MAP **SUFFIXES**

L	Halfward binary Contain the length of data entered by the terminal operator
F	One byte flag field
I	Contains the data entered by the operator
A	One byte field that contains Attribute byte
O	Contains data to be sent to terminal

SYMBOLIC MAP **FORMAT**

- A 12 byte TIOA prefix is automatically provided.
- When performing input functions fields suffixed with "L", "F", and "I" are meaningful.
- When performing OUTPUT functions, the fields suffixed with "A" and "O" are meaningful.Contains the data to be sent to the terminal.

OUTPUT MAPPING

"<u>MAP ONLY</u>"

```
EXEC CICS SEND
     MAP ("mapname1')
     MAPSET ('mapset1')
     MAPONLY
END-EXEC
```

"<u>DATA ONLY</u>"

```
EXEC CICS SEND
     MAP ('mapname1')
     MAPSET ('mapset1')
     DATA ONLY.
END-EXEC.
```

1. MAP ONLY option.
    - Use the physical map only.
    - Field headings, attribute bytes, and the location of where all information is to be placed is sent.
2. DATA ONLY
    - Use the symbolic map only
    - only the data in the symbolic map is sent to the screen.
3. Neither " MAPONLY NOR DATAONLY"

```
EXEC CICS SEND
     MAP ('map-name1')
     MAPSET ('mapset1')
END-EXEC
```

    - The physical map and the data from symbolic map is sent to the terminal.

Other options of **SEND** command.

- ERASE: Current screen is erased before the map specified appears on the screen
- ERASEAUP: erase all the unprotected fields.
- FREEKB: to free the keyboard
- ALARM: to make an alarms sound.
- FRSET: to reset MDT to zero
- CURSOR: to place the cursor in a specified field

CURSOR POSITIONING

- Static positioning
    - If IC option is specified in the ATTRB field of DFHMDF macro the cursor will be placed at this field.
- Dynamic / symbolic positioning.
    - Place (-1) into the field length field ("L" suffix). Cursor will be placed in the field.
- Dynamic / Relative positioning.
    - Cursor (data-value) option is used.
    - Data-Value will have the value at which the cursor has to be positioned.
    - E.g.. EXEC CICS SEND

```
                MAP (…..)
                MAPSET (…..)
                CURSOR (100)
                ERASE
          END-EXEC.
```

- MAPFAIL condition will caused in RECEIVE MAP command.
    - If the data to be mapped has a length of zero.
    - If the operator presses any key (clear, PA, PF, ENTER, Keys) without entering any data.

ACCESSING AND DISPLAYING MAP FIELDS

MAP STORAGE AREAS

- Placing maps in the Program
    - Any of the three plans for redefinition of maps may be used with either of the 2 alternatives for placing maps in your program.
- WORKING - STORAGE SECTION
    - Copying a symbolic description map structure here makes the area automatically available whenever the program is invoked.

- LINKAGE SECTION
  - Copying a symbolic description map structure here does NOT mean the storage will be available. Some methods for providing storage are passing a COMMAREA, acquiring temporary storage with the SET option, or using a GET MAIN command.

SEND / RECEIVE

- **Sending from the Symbolic description map**

```
SEND MAP (`MAP1')
MAPSET(`SET1')
MOVE `MAP1' TO MAPVAR
MOVE `SET1' TO SETVAR
SEND MAP (MAPVAR)
MAPSET(SETVAR)
FROM (MAP1O)
```

  - **SEND MAP Coding Alternatives** You can code the SEND MAP command to locate the symbolic description map in several ways:
    - Using constants in the name field for MAP and MAPSET(FROM is not required)
    - Using variables in the name field for MAP and MAPSET (This makes FROM a required parameter)
    - Using only the MAP parameter. In this case the name in the MAP option must be the MAPSET name.
- **RECEIVING into the symbolic description map**

```
RECEIVE MAP(`MAP1')
MAPSET(`SET1') ... MAP1 I
Move `MAP1' to MAPVAR
Move `SET1' to SETVAR ..... MAP1 I
RECEIVE MAP (MAPVAR)
MAPSET(SETVAR)
INTO (MAP1 I)
Receive MAP (`SET1') ... SET1 I
```

  - **RECEIVE MAP coding alternatives** You can code the RECEIVE MAP command to locate the symbolic description map in several ways:
    - Using constants in the name field for map and mapset (INTO/SET is not required) This is the most commonly used format.
    - Using variables in the name field for MAP and MAPSET. This makes INTO (database) or SET a required parameter.
    - Using only the MAP parameter. In this case the name in the MAP option must be the MAPSET name.

```
Linkage Section.
   01 DFHCOMMAREA
   01 LST
      02 PTR-2-LIST PIC S 9(8) COMP.
      02 PTR-2-BMS PIC S 9(8) COMP.
RECEIVE MAP (`MAP1') MAPSET (`SET1')
SET (PTR-2-BMS) ..... MAP1 I
```

- Using the SET option requests CICS to get the storage and return a pointer to it. The symbolic description map must be in the LINKAGE SECTION.

OUTBOUND FUNCTIONS

```
SEND MAP (`MAPA') MAPSET (`SETA')
        [ERASE/ ERASEAUP]
        [FREEKB]
        [ALARM]
        [FRSET]
        [PRINT]
```

ERASE	Erase Buffer, place cursor in upper left corner then write
ERASEAUP	Erase all the unprotected fields before the Write
FREEKB	Unlock Keyboard after the write
ALARM	Active alarm with the write
FRSET	Set all MDT currently on to off
PRINT	Start the 3270 print operation.

Control Functions:

- Typically the first type of command in the program is a SEND MAP. Certain control functions may be included in that command.
- ERASEAUP will clear out each field whose attribute is unprotected. It will NOT alter any attribute settings.
- If you do not free the keyboard using FREEKB, the operator will have to press the RESET key before entering data.
- If you code FRSET, all attribute bytes currently having Modified Data Tags (MDT) set on will be set off. Selective resetting of the MDT's must be done another way.
- When sending data to a 3270 screen the actual printing from the buffer will occur when the PRINT function is requested.

Attributes :

- The `A' suffixed field is an attribute field which controls the following:

PROTECTED/UNPROTECTED
ASKIP
NUM
MDT
Non Display (dark)DISPLAY (normal/bright)

- If the color or highlighting of a field has to be changed, additional symbolic fields are needed which are called the EXTENDED ATTRIBUTES.

EXTENDED ATTRIBUTES

- DSATTS (for symbolic map) and MAPATTS (for physical map) support the extended attribute characteristics
- The MAPATTS allows you to set up the physical map with any of the characteristic(s) coded.
- The DSATTS will create appropriate suffixed labels for the attribute characteristic(s) coded.

INBOUND FUNCTIONS - AID/CURSOR CONTROL

- Attention Identifier (ID) and Cursor:
  - On a RECEIVE, CICS updates the EIB with the following information:
    - The screen cursor position relative to zero is placed is EIBCPOSN.
    - The name of the input key the terminal operator pressed is placed in the field EIBAID.

EIBAID/CURSOR

- WORKING-STORAGE SECTION.

```
           COPY DFHAID.
           ......
PROCEDURE DIVISION.
           IF EIBAID = DFHPF12 THEN
           ......
           IF EIBAID = DFHENTER THEN
           ......
           IF EIBCPOSN LESS THAN 80 THEN
           ......
```

- When you first enter your program as a result of a transaction id, you can test EIBAID and/or EIBCPOSN. This may be done prior to issuing a RECEIVE command, if so chosen.

AID/CURSOR - SAMPLE CODING (CONTROL)

- If the operator uses any PA key or presses the CLEAR key, no data is transmitted
- EIBAID is useful when function keys are defined for the user.
- for eg. PF12 may be the exit function
      PF3 may be an update function etc.
  Thus testing for the type of AID will alter the logic flow.
- EIBCPOSN can be used to determine where the cursor was positioned on the screen. This information is especially useful with screens containing an action bar.

CURSLOC

Sample Map

```
MAPSETA DFHMSD TYPE=&SYSPARM, MODE= INOUT,
                            TERM=ALL, LANG=COBOL, TIOAPFX=YES,
                            STORAGE=AUTO
 MAP1       DFHMDI SIZE=(24,80), LINE=1, COLUMN=1,
                           CURSLOC=YES
              DFHMDF POS(2,1), LENGTH=4, INITIAL=`NAME',
                            ATTRIB=ASKIP
 NAME       DFHMDF POS(2,6),LENGTH=20,
                            ATTRB=(UNPROT, IC)
               DFHMDF POS(2,27), LENGTH=1,
                            ATTRB=PROT
```

- CURSLOC= YES allows you to determine after a RECEIVE MAP command, which map field had the cursor in it.
  CURSLOC=NO is the default.
- CURSLOC=(NO/YES) may be coded on the DFHMSD or the DFHMDI macro. If coded on the DFHMSD macro, it will provide a default for all the maps in that mapset.
- When CURSLOC=YES, BMS will set the `F' suffix field to X'02' indicating that field contained the cursor. If the cursor is in a field for which there is no symbolic label i.e. a DFHMDF with no label the program will not be notified.
- Note : The `F' suffix field continues.
- To be used to indicate the operator pressed the erase to end of fixed (EOF) key by being set to X'80'.
- Therefore, if CURSLOC=YES it is possible to have both these conditions occur for the same field, in which case the `F' suffix field will contain a X'82'.

EDITING

- Map field Definition PICIN/PICOUT
- Built in function De-edit command.
- When data is sent out via the `O' suffix fields or received into the `I' suffix fields you may want a definition other than PIC x. PICIN AND PICOUT allows the user to use other COBOL PICS such as $,Z etc.
- If the date contains special characters, you may want to remove them using the BIF DEEDIT command.

PIC IN/PIC OUT

- If PICIN/PICOUT is not coded in the macro, the pic generated is always PIC X (length of field)
- By using PICIN/PICOUT BMS can be forced to generate the appropriate PIC.
- PICIN tells BMS how to move data into the `I' suffix field.

- PICOUT tells COBOL how to edit your data move to the `O' suffix field.

FIELD EDIT BUILT-IN FUNCTION

```
Amount    BIF   Amount
$5431.80 EDIT 00543180
     EXEC CICS
     BIF DEEDIT
     FIELD (amount)
     LENGTH (8)
     END-EXEC.
```

- BIF DEEDIT is used to remove the special characters from the input field.
- The Amount field displayed has a dollar sign and a decimal point
- By using BIF DEEDIT dollar sign and decimal point is removed. Thus the number can be used for arithmetic operations.
- COBOL compiler requires LENGTH specification:
- VS COBOL-II uses the implied length of the data-area used in the field parameter.

CICS PROGRAM COMPONENTS

OBJECTIVES

- Structure of CICS Application Program
- CICS Management Functions
- Starting a Task
- Conversational& Pseudoconversational transactions
- CICS Program preparation
- CICS Program testing & Debugging
- CICS commands
- Passing Data across tasks

STRUCTURE OF CICS APPLICATION PROGRAM

Identification Division
Program - ID required

- Other comments as below, are optional but recommended.
    - Author
    - Date-Written
    - Date-compiled
    - Remarks

Environment Division

- Only header is required

Other requirements:

- COBOL statements and CICS commands should be coded
- The following COBOL statements are prohibited.
- ACCEPT, CURRENT-DATE, DATE, DAY, DISPLAY, EXHIBIT, STOP RUN, TRACE
- Any I/O Statements(OPEN, CLOSE, READ, WRITE, REWRITE, DELETE, START)
- REPORT WRITER feature
- SORT feature
- CALL statement is allowed if the called program does not issue any CICS commands or inhibited COBOL statements mentioned above.

TERMINATION STATEMENTS

Notes : This is not the way to terminate a CICS program. CICS has a command for that purpose. Nevertheless, COBOL and VS COBOL II have three statements to conclude programs.

- Control must not be allowed to pass beyond the last statement of a CICS Program.
- STOP RUN in COBOL uses operating system facilities, and therefore, is discouraged.
- EXIT program is ignored if the program has not been called.
- CICS RETURN COMMAND and/or GOBACK statement is recommended.

CICS MANAGEMENT FUNCTIONS

- Transactions
- Task
- Program

Transactions

An exchange between a terminal and a data base representing an application process. For example, an inquiry or a deposit and balance update

Task

A specific instance of a transaction i.e. a unique unit of work.

Program

Prepared statements compiled or assembled into an executable module of machine instructions.

CONVERSATIONAL TRANSACTION

- Program uses a pair of SEND and RECEIVE commands.
- Program waits until the user responds.
- Resources are held until the user responds.
- Very inefficient way of conversing with the user.

PSEUDO CONVERSATIONAL TRANSACTION

- The task is terminated after a message is sent with a linkage for the next task. CICS provides a facility (COMMAREA) to made it easier to accomplish this.
- When the user completes response (by pressing enter) reset task is automatically initiated by CICS.
- The task receives the message from the terminal & processes it.
- This is a multitask operation from system's point of view.

PROGRAM PREPARATION

- Running the DB2 precompiler first is the preferred method. DB2 precompiler precedes another process, binding, not mentioned here.
- Output of the DB2 precompiler can serve as input to the translator.
- Output of the translator will be input to the compiler.
- Messages or warnings are provided on all the listings.
- TRANSLATOR recognizes EXEC CICS and EXEC DLI statements. They are commented out and replaced with statements in the appropriate language. Here, COBOL MOVE instructions and a CALL are inserted and passed on to the COBOL compiler.
- DB2 Precompiler is supplied by the relational data base managers, DB2 and SQL/DS. It recognizes EXEC SQL statements which it will comment out and replace with in our case. COBOL PERFORM and CALL statements.
- Output of the transaction is i/p to the compiler
- The o/p of the linkage editor is executable. The load module is placed in the CICS online program library.
- Messages or warnings are provided on all the listings. The compiler listing is or limited use if the translator listing would not process all commands.

TESTING

```
CEMT set program (prg1) New comp
Or
CEMT S PR (prg1) N
```

- After making changes to a program the new version replaces old version, but CICS which is currently executing has no way of knowing this automatically. The CICS processing program table PPT still points to the old-version.
- To avoid testing with the old version, you must use the CICS-provided CEMT transaction to update the pointer to the program.

COMMON FORMAT

- A CICS command consists of a keyword phrase, delimiter, function, options and their argument values.
- Be careful about periods. Avoid them after END-EXEC. Where you don't really want them.
  For eg. within an **If .... then ... else statement**
- The translator will place a period into the generated code if a period follows end-exec.

ARGUMENT VALUES

data - value	PIC S9(4) comp Halfword binary PIC S9(8) comp Fullword binary PIC X(15) character string constants permitted
data - area	COBOL data name (Not a constant) eg :- 01 Record-area. 05 Fld 1 05 Fld 2
name	COBOL data name Character string
label	Paragraph name
hhmmss	PIC S9(7) comp 3 Packed decimal
pointer-ref	BLL cell Usage pointer

RECEIVE COMMAND

```
EXEC CICS RECEIVE
    INTO (data area)
    LENGTH(ml)
END-EXEC
```

- RECEIVE command is used to receive incoming data from the terminal to which this CICS transaction is associated.
- A receiving area must be defined in working storage section and has to be specified in the INTO parameter.
- Length field must be defined in working storage section as a S9(4) comp. It has to be specified in length option.

SEND COMMAND

```
EXEC CICS SEND
    FROM (data area)
    LENGTH(ln)
END-EXEC
```

- The data to be sent must be stored in working storage section, and this field name has to be specified in the FROM parameter.
- Length must be specified the same as that of the Receive command.

EXCEPTIONAL CONDITIONS

- RESP option
    - Define a full word binary field S9(8) comp in the working storage section as the response field.
    - Place RESP option with the response field in any CICS command.
    - After command execution, check the response code in the response field with DFHRESP (xxxx)where xxxx is
        - NORMAL for normal completion or Any exceptional condition
        -

HANDLE CONDITION

- Handle condition command is used to transfer control to the procedure label specified if the exceptional condition, specified occurs.
- Remains active until the end of program or another handle condition request overrides it.

IGNORE CONDITION

- Ignore condition command causes no action to be taken if the condition specified occurs in the programs.
- Request by the IGNORE CONDITION command is valid until the subsequent HANDLE CONDITION command for the same condition.

NO HANDLE OPTION

- If NOHANDLE option is specified in any CICS command, no action will be taken for any exceptional condition occurring during execution of this command.
    - Eg : EXEC CICS SEND

```
          From (...)
         Length (...)
         NOHANDLE
      END-EXEC
```

FORMATTING TIME AND DATE

- ASKTIME Command
    - used to request the current date and time
    - EIBDATE and EIBTIME fields have the values at the task initiation time.
- FORMAT ***EXEC CICS ASKTIME END-EXEC***

FORMAT TIME COMMAND

- Used to receive the information of data and time in various formats.
    - Format

```
  [YYDDD (data - area)]
```

```
          [YYMMDD (data - area)]
          [YYDDMM (data - area)]
          [MMDDYY (data - area)]
          [DDMMYY (data - area)]
          [DATESEP(data - value)]
          [DAY OF WEEK (data - area)]
          [DAY OF MONTH (data - area)]
          [MONTH OF YEAR (data - area)]
          [YEAR (data - area)]
          [TIME (data - area)]
          [TIMESEP (data - value)]
```

- DATESEP represents data separator (default is "/").
- TIMESEP represents time separator (default is ":").
- The data area for the ABSTIME option of ASKTIME and FORMATTIME commands must be a 15-digit packed decimal data type.

DELAY COMMAND

- used to delay the processing of a task for the specified time interval or until the specified time.

```
FORMAT
    EXEC CICS DELAY
        INTERVAL (002000)
        TIME (152000)
    End - EXEC
```

- Task will be suspended for 20 minutes if INTERVAL is specified or until 15:20:00 if TIME is specified.

COMMAREA

- Passing data via the COMMAREA
- pseudo conversational task to task
- Linking program to program

PSEUDO CONVERSATIONAL

Pseudo conversational technique is uses the multiple transaction identifiers(pct entries) and multiple program (pct entries). It performsthe terminal conversation in the following way:

A conversational program is logically and physically divided into separte programs after sending a message and before receving the message. For each separate program, a unique cics trasction identifier is assigned. before terminating the program, each program issues the RETURN command with the next transaction identifier which is associated with the next program, unless it is the least return to CICS itself. in this way, a series of terminal conversations can be carried out continuously.

PASSING DATA TO NEXT TASK

- **Notes :**
    - The first time commarea is passed, it must begin as an area of storage in the working storage section of the program passing it.
    - A commarea parameter in the RETURN will pass the area to the program associated with the subsequent transaction. In this case, itself.
    - The subsequent program (in this case the same program) must define access to all the commarea that was passed to it.
- PAYROLL as both the sender and the receiver of the COMMAREA needs the working storage definition to send and the linkage section DFHCOMMAREA to receive.
- PAYROLL must therefore be able to distinguish between FIRST TIME into the program. When there is no COMMAREA and subsequent times in. Where one exists in the Linkage section. The EIB field, EIBCALEN indicates the length of the commarea.

EIBCALEN

- First time into the program no commarea exists, therefore EIBCALEN = 0
- While returning the control to CICS the working storage is loaded and this is sent via the commarea parameter in the RETURN Transid.
- On subsequent entry, commarea exists and is automatically made addressable by CICS in the linkage section of DFHCOMMAREA

PASSING DATA USING LINK

- To pass control from one program to another and then return to the original like executing a subroutine.
- The link command passes control to another program defined in CICS PPT expecting that the program will return to the linking program instruction following the LINK command. This happens when the linked program issues a RETURN command.
- Data may be passed using the commarea.
- The commarea is shared between the two program regains control may changes made to the commarea by the linked program are accessible.
- The two programs executive under the same task.
- The working storage section for the linking program is retained. Working storage for the linked program is automatically released after its RETURN command is executed.

- To pass control from one program to another and then return to the original like executing a subroutine.
- The link command passes control to another program defined in CICS PPT expecting that the program will return to the linking program instruction following the LINK command. This happens when the linked program issues a RETURN command.

Reading External Data

Functional overview

- DIRECT RETRIEVAL
  - VSAM DATA STRUCTURES
- DIRECT RETRIEVAL
  - RELATIONAL TABLE ROW
- BROWSE
  - VSAM DATA STRUCTURE
- BROWSE
  - RELATIONAL TABLE ROWS

- Entry for VSAM file has to be there in FCT (File Control Table)
- Each entry contains all descriptive information for the file it represents. So, programmer need not define the physical organization and other attributes of the files.
  - The File parameter coded in the program must be the same as the file name in the FCT.
  - Interface between CICS and Relational Database is called CICS attachment Facility. Statement are coded in SQL language in the application program to Communicate data requests to the database.

TOPICS

- DIRECT RETRIEVAL
  - VSAM FILE RECORD
  - RELATIONAL TABLE ROW
- BROWSE
  - VSAM FILE RECORDS
  - SET OF RELATIONAL TABLE WORKS

VSAM DATA STRUCTURES

- CICS uses the following VSAM structures
  - Key sequenced data set (KSDS)
  - Entry sequenced dataset (ESDS)
  - Relative record dataset (RRDS)

PROGRAM ORGANIZATION

- File attributes are defined in the FCT for each file
- Files are opened by CICS
  - Immediately after system initialization if specified in the FCT.
  - In response to a file access request from an application if the file is closed
  - In response to a master terminal CEMT request from an operation.
- Application program is not responsible for open / close of files

RECORD IDENTIFICATION

- RECORD KEY
- RELATIVE BYTE ADDRESS
- RELATIVE RECORD NUMBER
- PARTIAL KEY
    - key of the record to be read is specified in the RIDFLD. for KSDS
    - Key specified can be a full key or partial key
    - If partial key, key length has to be provided
    - RBA (Relative Byte Address)
    - Can also be used instead of actual key value
- For ESDS
    - RIDFLD contain a 4 byte RBA
- For RRDS
    - RIDFLD contains 4 Byte binary relative record number.

RECORD KEY DEFINITION EXAMPLE

```
WORKING - STORAGE SECTION.
     05 RECKEY PIC X(6).
PROCEDURE DIVISION.
     MOVE VALUE TO RECKEY.
```

- RIDFLD must be set to the value of the key of the record to be retrieved.
- RIDFLD must be large enough to hold a full record key even when a partial key is used.

READ COMMAND

- READ command with INTO Option. (FULL KEY)
    - Reads the record specified by the full key.
    - The data content of the record is moved into the specified data-area defined in the working storage section.
    - FORMAT

```
EXEC CICS READ
     DATASET (name) | FILE (name)
     INTO (data-area) |SET(ptr-ref)
     RIDFLD (data-area)
     [ LENGTH (data-values) ]
END - EXEC.
```

- DATASET / FILE names the file.
- It must be defined in FCT.
- INTO names the field in the working storage section where the data has to be placed.
- RIDFLD is the key field.
- LENGTH is half word binary.
- It indicates maximum length of the record to be read. It is optional.

EXCEPTIONAL CONDITIONS

- DUPKEY : If duplicate record is found for the specified key.
- NOTFND : If the record is not found for the key specified.
- LENGER R LENGERR : The specified length (in LENGTH OPTION) is shorter than the actual record length.
- NOTOPEN : When file specified is not open.

The exceptional condition can be trapped using RESP option in the READ command.

ADDRESSABILITY TECHNIQUES

```
   EXEC CICS XCTL
         PROGRAM (PROGRAM NAME)
         RESP (EXCEPTION)
   END-EXEC.
  IF EXCEPTION = DFHRESP (PGMIDERR)
```

- a module given control through the use of a CICS XCTL command will not return to the program that issued the XCTL.
- The required. program name is character string constant (max 8 characters)
- The PGMIDERR exception condition occurs when the name is not in the PPT.

LOGICAL LEVELS

- The linked to program runs at a new logical level and returns to a logical level back to the linking program.
- The linking program and its storage area remain available.

Notes :

- To quit the repeated execution simply RETURN without the TRANSID option.
- Any linked program could use the same COMMAREA of the parameters so indicated.
- The transid & commarea option easy enough to use to make this method practical.

PROGRAMS TO PROGRAM TRANSITION

- CICS LINK
- CICS XCTL
- COBOL CALL
- Alternative to XCTL or LINK ? COBOL CALL·
- COBOL CALL passes control to other programs.

PASSING DATA USING INPUT MSG

- INPUT MSG & INPUT LEN PARAMETERS USED WITH XCTL OR LINK.

- Receiver uses EXEC CICS RECEIVE command

PASSING DATA USING LINK

```
  AREA1 PIC x (200)  (prog1)
```
dotted
```
  Linkage Section         (prog2)
     01 DFHCOMMAREA
     05 AREA2 PIC X(200)
```

- 1ST Program - COMMAREA - Length 100
- Data violation as 2nd Program (receives) tries to move 200 char

PASSING A COMMAREA WITH XCTL

- If data is to be passed to the XCTLed program, a COMMAREA can be used.
- Data area is to be located in the Linkage Section of the receiving program.
- COMMAREA used with RETURN, LINK & XCTL

ADDRESSABILITY

- DFHCOMMAREA & DFHEIBLK : Addressable automatically by CICS
- Dynamically acquired storage : Addressable by program
- Not necessary to always do a EXEC CICS GETMAIN explicitly

CICS QUEUEING FACILITIES

- Two facilities to store data that are temporary in nature.
- This data is created or collected by one or more online transaction to be used later by the same transaction or by a different transaction or even later passed to a batch program.
  They are
    - Transient data Queue (TDQ)
    - Temporary storage Queue (TSQ).

TRANSIENT DATA QUEUE

- They are identified by a 4 character ID called destination ID
- Destination ID and other characteristics of TDQ are defined in the destination control table (DCT) by the system programmer.
  2 types of TDQ's
    - Intra Partition TDQ
    - Extra Partition TDQ
- Intra Partition TDQ - Processed only within the same CICS region
- Extra Partition TDQ - Individual Sequential Files processed between the transaction of the CICS region and the system outside of the cics region.

INTRA PARTITION TDQ

All Intra partition TDQ are stored in only 1 physical file (VSAM)

- Record from the queue can be returned sequentially.
- Record can be written sequentially.
- Records can be of variable length format
- Several tasks can write to the same TDQ but only one task can read from TDQ.

Intra Partition TDQ is used in application such as

- Interface among CICS transaction. Applicati on program 1 TDQ Appl . Pgm 2 report
- Automatic task Initiation (ATI)
- Message routing
- Message Broad cast.

EXTRA PARTITION TDQ

Extra partition TDQ is a separate physical file & may be a disk, tape or reporter.

- DCT determines the initial open / close status of a file while the file can be opened or closed through the master terminal transaction during CICS session.
- TDQ can be defined as an Input or output but not both.
- Records are fixed, variable, blocked or unblocked.

TRANSIENT DATA OUTPUT

- Appears only for Intra-partition TDQ.
- Deletes all records associated with the named destination.
- All associated storage is released.

EXCEPTIONAL CONDITIONS

1. Special handling required
   LENGERR - length specified is greater than the maximum record length specified in DCT
2. Qzero - Destination empties or end of TDQ error
   Qlderr - The dest ld specified cannot be found in DCT.

TEMPORARY STORAGE

- TSQ is a queue of stored records.
- Created & deleted dynamically by application program.
- Used as a scratch pad
- Queue ID is of length 1-8 bytes
- TSQ is of variable length
- Records can be stored in main or auxiliary storage
- The records once written remains accessible until the

- entire TSQ is deleted
- Records can be read sequentially or directly
- Records can be re-read & updated.

WRITEQ TS

- To write or re-write a record in TSQ
```
      EXEC CICS WRITEQ TS
          QUEUE (NAME)
          LENGTH (DATA-VALUE)
          [ITEM (DATA-AREA)
          [REWRITE]
          [MAIN | AUXILIARY]
    END-EXEC.
```
- ITEM - If this option is coded CICS write return the item number assigned to the record just written.
- REWRITE - is used to rewrite the record identified by ITEM.
- Main / Auxiliary - To specify the storage medium. Will be stored in main if auxiliary storage not supported.

READQ TS

- Can be used to read records either sequentially or directly.Syntax
```
      EXEC CICS READQ TS
          QUEUE (NAME)
          INTO (DATA - AREA)
          LENGTH (DATA-VALUE)
          [ITEM (DATA-VALUE) | NEXT]
          [NUMREC (DATA-AREA)]
    END-EXEC.
```
- NEXT - to retrieve the next - logical record in the TSQ. mutually exclusive to the item option
- NUMREC - the data area is defined as PIC 9(4) comp. to find the Total no. of records in the TSQ.
- Item - for direct access specify the item no of the record.

DELETEQ TS of TSQ

- EXEC CICS
```
        DELETEQ TS
        QUEUE (NAME)
    END-EXEC.
```
- All records is TSQ are deleted.
- All associated storage is released.

EXCEPTIONAL CONDITIONS

- Special handling required
  - Itemerr - Item number specified is not in the range of entry number assigned for the Queue.
  - Lengerr - Length specified is greater than the maximum record length.
- error
  Q iderr - specified is Queue id not found.

TESTING & HANDLING EXCEPTIONS

COMMANDS FOR TESTING APPLICATION PROGRAMS

- CECI (Command Level Interpreter) is a CICS - supplied transaction which performs syntax checking of a CICS command. If the syntax is satisfied, it will execute the command.
- CEBR (Temporary Storage Browse) is a CICS - supplied transaction which browses Temporary Storage Queue (TSQ).
- CEDF is a CICS - supplied transaction which monitors the execution of an application program as an interactive debugging aid.

APPLICATION PROGRAM SUPPORT

- RESP and NOHANDLE
- IGNORE CONDITION
- HANDLE CONDITION
- HANDLE AID
- HANDLE ABEND

COMMAND LEVEL INTERPRETER

For invoking CECI, type CECI with the CICS command to be interpreted.

- The first screen lists all the possible CICS commands.
- Giving question mark (?) before the command requests a syntax check only. No execution.
  Ex.
  ```
  CECI SEND MAP (`SPOOMPO')
            MAPSET (`SPOOMSO')
            ERASE
  ```

BROWSING CICS QUEUES

- CEBR can be invoked while you are already in the CEDF mode.
- Press the PF5 key to display the working storage section.

- Then, press PF12 key to invoke CEBR.
  CEBR allows to browse information in Temporary Storage (TS) queues.
- Help (PF1) give you a list of CEBR commands on the screen.
- TS queues are retained until purged.

EXCEPTION HANDLING
CICS to respond to exceptional conditions in one of three ways:

- RESP option: - The RESP option can be specified in any CICS command. Its function is similar to the return code in the batch program.
    - Define a fullword binary field (S9(8)COMP) in the working storage section as a response field.
        - Place the RESP option with the response filed in a command.
        - After command execution, check the response code in the response field with DFHRESP (xxxx), where xxxx is the - NORMAL
          - Any exceptional condition
- HANDLE CONDITION This command is used to transfer control to the procedure label specified if the exceptional condition specified occurs.
- IGNORE CONDITION This command causes no action to be taken if the condition specified occurs in the program.

```
 EXEC CICS HANDLE CONDITION
     Condition (Label)
     [Condition (Label)]
     [Error (Label)]
END-EXEC.
```
or
```
EXEC CICS IGNORE CONDITION
     Condition
     [Condition]
END-EXEC.
```
- ABEND CODE

  If an exceptional condition occurs during execution of a CICS application program and if the program does not check the exceptional condition, CICS may continue executing the program or terminate abnormally the execution of the program, depending on the exceptional condition and the command involved.

OPTION FOR EXCEPTION HANDLING

- CODE RESP keyword in commands: CHECK USER-SUPPLIED FIELD IN WORKING - STORAGE.
- CODE NOHANDLE KEYWORD IN COMMANDS.
- HANDLE CONDITION
  COMMANDS
- IGNORE CONDITION
  SYSTEM DEFAULT -- ABEND

CICS Adend Codes :-

Execute Interface Block and cics abend codes

Some of the more **common CICS abends** are briefly described below. These are only brief descriptions and do not cover all possible reasons.

ASRA

This is the most common abend in CICS. It indicates a Program Check Exception, roughly equivalent to having an S0C7 in a batch program. Check for spaces in a packed decimal numeric field and changes to the file and record layouts.

AEIx and AEYx

There are numerous abends that start with AEI or AEY. They indicate that an exception has occured, and RESP (or NOHANDLE) is not is use. The last character indicates the exact error

AEI0

indicates a PGMIDERR.

AEI9

is a MAPFAIL condition,

AEIO

indicates a duplicate key (DUPKEY) condition.

AEIN

indicates a duplicatebrecord (DUPREC) condition.

AEID

indicates an End of file condition.

AEIS

indicates that a file is not open (NOTOPEN)

AEIP

indicates an invalid request condition (INVREQ)

AEY7

indicates that you are not authorised to use a resource (NOTAUTH)

AICA

This abend usually occurs if your program is looping. There are CICS parameters that determine how long a task can run without giving up control. The ICVR parameter in the CICS SIT table can be used to specify a value for all tasks running in CICS, or you can specify a RUNAWAY value when you define a transaction . If a program is looping then you may not get an AICA abend, because the timer can be reset when certain events occur, eg some EXEC CICS commands may reset the timer to zero.

ATCH and ATCI

These abends indicates that the task was purged. The task may have been purged by someone issuing a CEMT command to purge the task, or by CICS because the Deadlock timeout limit has been exceeded or because there was not enough virtual storage available to run all the tasks in CICS (Short on Storage)

APCT

A program was not found or was disabled. Check the transaction definition to see if the program name was misspelled. Check that the program is enabled. Check that the program is in an appropriate Load Library (ie one defined to the current CICS system).

AKCP and AKCT

These abends indicate that a timeout of the task occurred. This may be due to a deadlock.

AFCA

A dataset could not be accessed because it was disabled.

ABM0

The specified map was not found in the specified mapset. Check that you have not misspelled the map name.

The Execute Interface Block (EIBLK) contains a variable called EIBFN.This contains a value that tells you what CICS command was last executed. This value can be displayed as part of an error message, to aid in the debugging of your code or when an exception condition occurs

The values for EIBFN are show below.

Code	Command
0202	ADDRESS
0204	HANDLE CONDITION
0206	HANDLE AID
0208	ASSIGN
020A	IGNORE CONDITION
020C	PUSH
020E	POP
0210	ADDRESS SET
0402	RECEIVE
0404	SEND
0406	CONVERSE
0408	ISSUE EODS
040A	ISSUE COPY
040C	WAIT TERMINAL
040E	ISSUE LOAD
0410	WAIT SIGNAL
0412	ISSUE RESET
0414	ISSUE DISCONNECT
0416	ISSUE ENDOUTPUT
0418	ISSUE ERASEAUP
041A	ISSUE ENDFILE
041C	ISSUE PRINT
041E	ISSUE SIGNAL
0420	ALLOCATE
0422	FREE
0424	POINT
0426	BUILD ATTACH
0428	EXTRACT ATTACH
042A	EXTRACT TCT
042C	WAIT CONVID
042E	EXTRACT PROCESS
0430	ISSUE ABEND
0432	CONNECT PROCESS
0434	ISSUE CONFIRMATION
0436	ISSUE ERROR
0438	ISSUE PREPARE
043A	ISSUE PASS
043C	EXTRACT LOGONMSG
043E	EXTRACT ATTRIBUTES
0602	READ
0604	WRITE
0606	REWRITE
0608	DELETE
060A	UNLOCK

060C STARTBR
060E READNEXT
0610 READPREV
0612 ENDBR
0614 RESETBR
0802 WRITEQ TD
0804 READQ TD
0806 DELETEQ TD
0A02 WRITEQ TS
0A04 READQ TS
0A06 DELETEQ TS
0C02 GETMAIN
0C04 FREEMAIN
0E02 LINK
0E04 XCTL
0E06 LOAD
0E08 RETURN
0E0A RELEASE
0E0C ABEND
0E0E HANDLE ABEND
1002 ASKTIME
1004 DELAY
1006 POST
1008 START
100A RETRIEVE
100C CANCEL
1202 WAIT EVENT
1204 ENQ
1206 DEQ
1208 SUSPEND
1402 WRITE JOURNALNUM
1404 WAIT JOURNALNUM
1602 SYNCPOINT
1802 RECEIVE MAP
1804 SEND MAP
1806 SEND TEXT
1808 SEND PAGE
180A PURGE MESSAGE
180C ROUTE
180E RECEIVE PARTN
1810 SEND PARTNSET
1812 SEND CONTROL
1C02 DUMP
1E02 ISSUE ADD
1E04 ISSUE ERASE
1E06 ISSUE REPLACE
1E08 ISSUE ABORT

1E0A	ISSUE QUERY
1E0C	ISSUE END
1E0E	ISSUE RECEIVE
1E10	ISSUE NOTE
1E12	ISSUE WAIT
1E14	ISSUE SEND
2002	BIF DEEDIT
4802	ENTER TRACENUM
4804	MONITOR
4A02	ASKTIME ABSTIME
4A04	FORMATTIME
5602	SPOOLOPEN
5604	SPOOLREAD
5606	SPOOLWRITE
5610	SPOOLCLOSE
5E06	CHANGE TASK
5E22	WAIT EXTERNAL
5E32	WAITCICS
6A02	QUERY SECURITY
6C02	WRITE OPERATOR
6C12	ISSUE DFHWTO
7402	SIGNON
7404	SIGNOFF
7406	VERIFY PASSWORD
7408	CHANGE PASSWORD
7E02	DUMP TRANSACTION

CICS FAQ’s:-

CICS FAQS

Question: what is difference between call and link ?
Answer: In case of call , whenever you do changes to the called program you need to compile the calling program also. In case of link , it is not needed .

Question: what are the differences between dfhcommarea and tsq ?
Answer: both are used to save data among tasks. but 1. commarea is private to that transaction only . like every transaction has its own commarea created by cics as soon as the transaction is initiated . however tsq , if qid is known can be accessed by other transactions also 2. commarea length is s9(4) comp ie 65k . but tsq can have any length.3. commarea is available only during the transaction is running. tsq if created with auxiliary option resides in aux memory and available even if main memory crashes.4.normally commarea is used to tranfer data from one task to another while tsq is used widely within the task as a scratch pad.

Question: What is Communication Area?
Answer: Communication Area is used to pass data between the programmer between the task.

Question: Which of the following statements correctly describe the syntax of CICS command language?
Answer:

1. If an EXEC CICS command must be continued onto a second line a hyphen (-) must be coded in column 7 of the continued line.
2. If an EXEC CICS command must be continued onto a second line an 'X' must be coded in column 72 of each line to be continued.
3. An EXEC CICS command CANNOT be coded within a COBOL IF statement,between the IF command and the period (.) ending it.
4. The END-EXEC delimiter is optional and never needs to be placed at the end of a CICS command.
5. The options specified within an EXEC CICS command can be in any order. For example 'EXEC CICS SEND FROM(MSG1) LENGTH(30) END-EXEC' can also be coded 'EXEC CICS SEND LENGTH(30) FROM(MSG1) END-EXEC'

Question: .A CICS program ABENDS with an ASRA ABEND code. What is its meaning?
Answer:

1. A link was issued to a program whose name does not exist in the PPT (Program Processing Table).
2. A program attempted to use a map that is not defined in the PCT (Program Control Table).
3. A security violation has occurred. The operator is not defined with the proper authority in the SNT (Sign-on Table) to use a particular file.
4. A program interrupt (0C0 or 0C1 or 0C2 or ...) has occurred in a CICS program.
5. An I/O error has occurred when attempting to use a VSAM file from a CICS program

Question: Which of the following commands, when issued by 2 different programs running at the same time, will prevent simultaneous use of resource 'SINGLE'?
Answer:

1. EXEC CICS PROTECT RESOURCE('SINGLE') LENGTH(6) END-EXEC.
2. EXEC CICS HOLD RESOURCE('SINGLE') LENGTH(6) END-EXEC.
3. EXEC CICS TASK SINGLE('SINGLE') LENGTH(6) END-EXEC.
4. EXEC CICS EXCLUSIVE RESOURCE('SINGLE') LENGTH(6) END-EXEC.

Question: How can you accomplish braykpoint in intertest?
Answer: U-for uncondishional braykpoint, C-for condishional braykpoint,and A-for automatic braykpoint

Question: how many ways are there for initiating a transaction?what are they?
Answer: There are six ways in initiating a transaction.they are as follows.

1. embedding four character transid on the top left most corner of the screen.
2. making use of EXEC CICS START TRANSID ( )
3. making use of EXEC CICS RETURN TRANSID ( )
4. By defining the transid in DCT (destination control table) to enable ATI (AUTOMATIC TASK INITIATION)
5. Making use of PLT (program list table)
6. By associating four character transid in PCT (program control table)

Question: which type of TDQ is read destructive?
Answer: intrapartition tdq is read destructive. extrapartition tdq is not read destrctive.

Question: The error code aeiv?
Answer: this is the error code for length,if length of the source data is more than the receiving field,this error will occur.this is the correct answer,previously i mentioned it as program id error.sorry for the wrong information.

Question: WHAT U MEAN BY AEIV ?
Answer: THIS IS THE ERROR CODE GIVEN BY THE SYSTEM ,IT MEANS PROGRAM ID ERROR.

Question: WHAT IS THE SIZE OF COMMAREA
Answer: THE DEFAULT COMMAREA SIZE IS 65K.

Question: What is ASRAABEND in CICS?
Answer: It occurs when program interuption takes place.e.g.: when alphanumeric string moved to numeric data itemOR when arithmetic calculations performed on nonnumeric data itemOR when an attempt made to read an occurance of a table beyond the defind occurances.

Question:What is a two Phase commit in CICS?
Answer: This occurs when a programmer Issues a Exec CICS Syncpoint command. this is called two phase because CICS will first commit changes to the resources under its control like VSAM

files. and the DB2 changes are committed. Usually CICS signals Db2 to complete the next phase and release all the locks.

Question: Answer to ANON's question, diference between TSQ & TDQ
Answer: TDQ is read destructive, TSQ is not. TSQ can be created dynamically, TDQ cannot be created dynamically. TSQ is temporary in nature (i:e it will be deleted when the program finishes execution, unless it is made permanent by making a entry in the Temporary Storage Table), TDQ is not. Hope this will suffice

Question: What is ENQ in CICS?
Answer: If any one want to restrict Trans-Id to single user, enter trans-id with ENQ. It won't allow any one else to use the same trans-id.

Question: In SYMBOLIC Cursor Positioning after moving -1 to the length field also the cursor is not positioned in that particular field.Give reasons?
Answer: You have to explicitly specify the word CURSOR between your EXEC CICS and END-EXEC in the program.

Question: What does EIB mean?
Answer: The EIB is the EXECUTIVE INTERFACE BLOCK. It is not the EXECUTE INTERFACE BLOCK. All TP monitors or transaction processors are know as EXECUTIVEs as they carry out process on behalf of a program module. CICS and DB2 are excutives.

Question: How many exceptional condition can be given in a HANDLE CONDITION?
Answer: Max. of 12 exceptional conditions can be given in a single HANDLE CONDITION.

Question: What command do you issue to delete a record in a transient data queue ?
Answer: READQ TD, the read is destructive. Yes it is correct but there is a restriction.U can deletethe records sequentially.. For example if one want to delete 10 th record directly it is not possible with this..

Question: How do you access the records randomly in TSQ ?
Answer: By specifying the ITEM option

Question: What command do you issue to delete a record in a transient data queue ?
Answer: READQ TD, the read is destructive.

Question: WHAT ARE DIFFERENT WAYS OF INITIATING TRANSACTION IN CICS
Answer: WE CAN INITIATE CICS TRANSACTION

1. BY GIVING TRANSACTION ID
2. BY GIVING CICS START COMMAND
3. AUTOMATIC TASK INITIATION.

Question: What is the difference between LINK and XCTL ?
Answer: The XCTL command passes control to another program, but the resources requested by the first program may still be allocated. A task does not end until a RETURN statement is

executed. While in LINK command, program control resumes its instruction following the LINK parameter. The disadvantage of LINK is that it requires that both the calling program and the called program remain in main memory even though both are no longer needed.

Question: What is the difference between CICS Program Control Table (PCT) and CICS Processing Program Table (PPT) ?
Answer: PCT contains a list of valid transaction ID. Each transaction ID is paired with the name of the program ,CICS will load and execute when the transaction is invoked. On the other hand, PPT indicates each program's location which pertains to a storage address if the program has already been loaded or a disk location if the program hasn't been loaded. PPT will also be used to determine whether it will load a new copy of the program when the transaction is invoked.

Question: What are the 3 common ways to create maps?
Answer: The first way is to code a physical map and then code a matching symbolic map in your COBOL program. The second way to create a physical map along with a matching symbolic map is to code only the physical map using the &SYSPARM option, CICS will automatically create a member in a COPY library. And the third way is to use a map generator such as SDF (Screen Definition Facility)

Question: What is Quasi-reentrancy?
Answer: There are times when many users are concurrently using the same program, this is what we call MultiThreading. For example, 50 users are using program A, CICS will provide 50 Working storage for that program but one Procedure Division. And this technique is known as quasi-reentrancy

Question: What is the difference between a physical BMS mapset and a logical BMS mapset?
Answer: The physical mapset is a load module used to map the data to the screen at execution time. The symbolic map is the actual copybook member used in the program to reference the input and output fields on the screen.

Question: How To Set MDT(Modified Data Tag) Thru Application Program?(Dynamically).
Answer: You have to move the following macro DFHBMFSE to the Attribute field of that particular Variable.

Question: What CICS facilities can you use to save data between the transactions?
Answer: COMMONAREA, TSQ & TDQ.

Question: How would you release control of the record in a READ for UPDATE?
Answer: By issuing a REWRITE,DELETE, or UNLOCK command or by ending the task.

Question: How would you release control of the record in a READ for UPDATE?
Answer: By issuing a REWRITE,DELETE, or UNLOCK command or by ending the task.

Question: What is the difference between a RETURN with TRANSID and XCTL ?For example prog. A is issuing REUTRN with TRANSID to prog B. Prog A. is issuing XCTL to prog B.
Answer: In RETURN with TRANSID the control goes to the CICS region and the user have to transfer the control to prog. B by pressing any of the AID KEYS.In XCTL the control is directly transfer to prog. B.

Question: What is the maximum number of exceptions that can be specified with a single HANDLE CONDITION command in CICS ?
Answer: SIXTEEN (16)

Question: WHAT WILL BE THE LENGTH OF THE EIBCALEN ,IF THE TRANSACTION IS USED TO CICS FIRST TIME?
Answer: THE LENGTH WILL BE 0(ZERO)
.
Question: WHAT IS DFHEIBLK?
Answer: DFHEIBLK is Execute Interface Block. It is placed in the linkage section automatically by CICS translator program. It must be the first entry in linkage section. CICS places values prior to giving control to the program and we can find almost any information about our transaction.

Question: What is the difference between the XCTL and LINK commands?
Answer: The LINK command anticipates return of control to the calling program, theXCTL command does not. Return to the calling program will be the result of the CICS RETURN command, specifying TRANSID(name of the calling program).

Question: What CICS command would you use to read a VSAM KSDS sequentially in ascending order?
Answer: First issue a STARTBR(start browse), which will position the browse at the desired record. Retrieve records by using subsequent READNEXT commands. Indicate the end of sequential processing with the ENDBR command. If the generic key is specified in the STARTBR command positioning in the file will be before the first record satisfying the generic key.For reading in descending order use the READPREV instead ofREADNEXT.

Question: What is the difference between pseudo-conversational and conversational?
Answer: Pseudo-conversational will start a new task for each input. By coding a CICS RETURN command specifying TRANSID(itself). Conversational will have an active task during the duration of the data entry.

Question: What is the COMMAREA(communications area)?
Answer: An area used to transfer data between diffrent programs or between subsequent executions of the same program. Needs to be defined in the Linkage Section.

DB2 Tutorial :-

Introduction to Databases

What is data?

- A representation of facts or instruction in a form suitable for communication.

What is a database?

- It is a repository for stored data.

What is a database System?

- An integrated and shared repository for stored data or collection of stored operational data used by application systems of some particular enterprise.
- Nothing more than a computer-based record keeping system

Advantages of DBMS over File Management Systems

- Data redundancy
- Multiple Views
- Shared data
- Data independence (logical/Physical)
- Data dictionary
- Search versatility
- Cost effective
- Security and Control
- Recovery ,restart & Backup
- Concurrency

Types of Databases (or Models)

- Hierarchical Model
- Network Model
- Relational Model
- Object-Oriented Model

HIERARCHICAL MODEL

- Top down structure resembling an upside-down tree.
- Parent child relationship
- First logical database Model
- Available on most of the Mainframe computers
- Examples: IMS

NETWORK MODEL

- Does not distinguish between parent and child.Any record type can be associated with any number of arbitrary record types
- Enhanced to overcome limitations of other models but in reality there is minimal difference due to frequent enhancement
- Example: IDMS

RELATIONAL MODEL

- Data stored in the form of table consists of multiple rows and columns.
- Examples: DB2,ORACLE,SYBASE

OBJECT-ORIENTED MODEL

- Data attributes and methods that operate on those attributes are encapsulated instructions called objects.

Types of Integrity

- Entity Integrity
- Referential Integrity
- Domain Integrity

Entity Integrity

- Is a state where no column that is part of a primary key can have a null values.

Referential Integrity

- Is a state where every foreign key in the first table must either match a primary key value in the second table or must be wholly null.

Domain Integrity

- Integrating of information allowed in column.

Entity Relationship Model

- E-R model is a logical representation of data for a business area.
- Represented as entities relationship between entities and attributes of both relationships and entities.
- E-R models are outputs of analysis phase.
- Example of relational Structure
  CUSTOMER places ORDERS
  ORDER has PRODUCTS
- Each order relates to only one customer (one-to-one)

- Many orders can contain many products (many-to-many)
- A customer can place any number of orders (one-to-many)
- In last example customer,order & product are called entities
- An entities may transform into tables.
- The unique identity for information stored in an entity is called a primary key
- Attributes which define the characteristics of the table.

DB2

OBJECTS

- Stogroup(Storage group)
- Database
- Table Space
- Table
- View
- Index

Stogroup (Storage Group)

- It is a collection of direct access volume, all of the same device type
- The option is defined as a part of table space definitions
- When a given space needs to be extended, storage is acquired from the appropriate stogroup.

Database

- A collection of logically related objects – like table spaces, index spaces,tables etc.
- Not a physical kind of object, may occupy more than one disk space.
- A STOGROUP & BUFFER POOL must be defined for each database.
- In a given database, all the spaces need not have the same STOGROUP.

TABLE SPACES

- Logical address space on secondary storage to hold one or more tables.
- It is the storage unit for recovery and reorganizing purpose
- Three types of table spaces are :
    - Simple
    - Partitioned
    - Segmented

Simple Table Spaces

- Can contain more than one stored tables
- Depending on application, storing more than one tables might enable faster retrieval for joins using these tables.

- Usually only one table is preferred. This is because a single page can contain rows from all tables defined in the database.

Segmented table spaces

- Can contain more than one stored tables , but in a segmented space.
- A ‘segment’ consists of a logically contiguous set of ‘n’ pages.
- No segment is allowed to contain records for more than one table.
- Sequential access to a particular table is more efficient.
- Lock Table on table locks only the table, not the entire table space.
- If a table is dropped ,the space for that table can be reclaimed within minimum reorganization.

Partitioned Table Space

- Primarily used for very large tables
- Only one table in a partitioned table space
- Individual partitions can be independently recovered and reorganized.
- Different partitions can be stored on different storage groups for efficient access.

TABLES

- A table is a collection of rows and columns.
- The data is stored on magnetic disks in a series of TABLES.
- Each COLUMN contains some specific information about suppliers and each ROW contains all the information about a particular supplier.
- For example SUPPLIER table looks like :

S#	SNAME	STATUS	CITY
001	PRASAD	20	CHENNAI
002	VASU	30	DELHI
003	SIVA	10	BOMBAY

VIEWS

- Views can be very practical ways of simplifying queries by reducing the number of different tables.
- Views can also hide sensitive data from users who don’t need access to it.

```
CREATE VIEW EMP_VIEW
AS
SELECT    EMPNO,NAME,DEPT,JOB   FROM    EMP
```

SYNONYMS

- Synonym is like a nick name to a table name and when no longer needed it can be dropped. Synonym access is specific to the user who has created it.

STRUCTURED QUERY LANGUAGE(SQL)

- A powerful database management language that performs the function of data manipulation, data definition and data control.
- A non-procedural language.
- The capability to act on a set of data and the lack of need to know how to retrieve it.
- It allows to specify WHAT the result should be, NOT HOW the result obtained.

SQL TYPES(Based on functionality)

- DDL - CREATE,ALTER and DROP
- DML - SELECT,INSERT,UPDATE & DELETE
- DCL - GRANT and REVOKE

SQL TYPES(Others)

- Static or Dynamic SQL
- Embedded or Stand-alone SQL

HOW DO YOU UTILIZE SQL?

- It is usually accessed on-line.
- It can be used interactively, such as SPUFI,QMF or it can be embedded in COBOL,PL/I,ASSEMBLER or FORTRAN program.

WHAT MIGHT YOU DO WITH SQL?

- As an end-user you will be using SQL on-line to access data and produce formatted reports.
- As an application programmer, you will use SQL to extract data from one or more tables and pass these data to another program for additional processing.
- SQL task is center about four basic functions & these functions correspond to the four basic SQL command (SELECT, UPDATE, INSERT, DELETE)

HOW TO CREATE A SIMPLE SQL QUERY?

- Search the table of your choice
- Select specific columns to be displayed from a table
- Select specific rows to be displayed from a table
- Specify the order in which to display data

How do you specify which tables you want to select from?

- Which tables contain data?
- Which columns to display from the table?
- Which rows to display?
- In which order to display the columns and/or rows.
- Any SELECT statement must specify a table
- If the table belongs to another user, you may have to prefix the table name with the other user's ID.

How do you specify which COLUMNS you want to display?

- Listing the name of the columns on a SELECT statement, separated by commas.

```
SELECT SNAME, STATUS
FROM   SUPPLIERS
```

Why not use the '*' in all your queries ?

- Using the '*' to indicate all columns should be displayed is inefficient if you don't need to see all the columns.
- You can put the columns in the order you want them.
- It is better to specify just those column you want.

How do you specify which rows you want to display?

- Using WHERE keyword .
- The format is WHERE followed by a condition or list of conditions.

```
SELECT SNAME
FROM   SUPPLIER
WHERE  STATUS=30
```

Can you put the data in Order?

- Using ORDER BY keyword to your query
- SQL automatically orders data in ascending order if you have any ORDER BY statement in your query.
- If you want your data in descending order, use the DESC keyword.
  SELECT *STATUS* ***FROM*** *SUPPLIERS* ***ORDER BY*** *STATUS DESC*

Can you order by more than one column?

- You can order by as many columns as you want.
- Simple add the columns you want to sort to the ORDER BY clause, separated by comma.

- Example:
  SELECT *SNAME ,STATUS, CITY* ***FROM*** *SUPPLIERS* ***ORDER BY*** *STATUS,SNAME*

Use of 'AND' and 'OR' to make a query more specific.

- Sometimes a single search condition is not precise enough to select only the rows you need.So you need to use the logical operator AND and OR.
- You can link as many search expression as you want using AND and OR.
- Using both operator together , searched expression linked by AND are paired up first.Parentheses can be used to force expression to be paired together.

How do you avoid duplicate data in your output?

- By adding the keyword DISTINCT immediately following SELECT , you suppress any duplicates. ***SELECT DISTINCT*** *STATUS* ***FROM*** *SUPPLIERS*

What are the relational operators?

- Here is the full list of operators:

SYMBOL	CONDITION
=	equal to
>	greater than
<	less than
>=	greater than or equal to
<=	lesser than or equal to
<>	less than or greater than
LIKE	containing certain characters
BETWEEN	within a range of values
IN	one of a set of values

Each condition also has a negative version.

IN

- Lets you retrieve data from each row whose columns has a value equal to one of the several listed values.
- Example: **WHERE** STATUS **IN** (20,10,30)
- Any row with STATUS column value of 20,10, 30 is selected.
- Each character value must be in quotes.
- Values are separated by comma & values list is enclosed in parenthesis.

BETWEEN

- Lets you retrieve data from each row whose columns has a value with two limits.
- Example: **WHERE** STATUS **BETWEEN** 10 AND 30
- Any row with STATUS column value of 10 thru 30 is selected.
- It can be used for both numeric and character values.

- The two values must be separated by AND, the lower value must be the first value.

LIKE

- Lets you retrieve data from each row whose columns has a value similar to the value specified in the WHERE CLAUSE.
- Example: **WHERE** SNAME **LIKE** ‘B%C’
- Any row with name column beginning with ‘B’ followed by ANY VALUE of ANY LENGTH, and ending with ‘C’ is selected.
- A ‘%’ represents any value of any length,where as an underscore ‘_’ represents any single character LIKE can’t use for any numeric column.

What is NULL?

- A NULL is a column value that does not exists.
- It is not a blank, not is it zero – it is a ‘non-value’.
- The columns must predefined to the DBMS as a column that may contain NULL values.

HOW DOES SQL TREAT NULL VALUES?

- Generally, NULL values are ignored.
- If you perform a calculation using a column with NULL values , those rows containing NULL's are ignored and a dash will output as the result.
- NULL means “VALUE UNKNOWN”, so it cannot meet any of the standard search conditions.

Example of NULL values

- To select all rows with a NULL value in the COMM column, you must use WHERE COMM IS NULL as your search condition
- To select all rows that do NOT have a NULL value, you must use WHERE COMM IS NOT NULL as your search condition
- Note:

WHERE COMM = 0 …	Who has earned no commission?
WHERE COMM IS NULL	Who is ineligible for commission?

ARITHMETRIC EXPRESSIONS

- An arithmetic expression is a calculation using numeric values and one or more of the arithmetic operators :addition (+), subtraction (-), multiplication (*), and division (/).
- Example: ***SELECT*** *SALARY + COMM,NAME,DEPT*
- Arithmetic expressions can also be used in a WHERE clause: ***WHERE*** *SALARY + COMM >5000.00*

RULES FOR CODING ARITHMETIC EXPRESSIONS

- Spaces on either side of the arithmetic symbols are optional.
  SALARY/12 OR *SALARY / 12*
- You can combine column names and constants in arithmetic expressions.
  *1998 - JOIN_YR, 1.0*SALARY*

HOW DOES SQL EVALUATE ARITHMETIC EXPRESSIONS?

- SQL works from left to right, evaluating multiplication and division before addition and subtraction (*, / , +, -).
- However any expressions in parentheses are evaluated first.

BUILT IN FUNCTION

- SUM
- MAX
- MIN
- AVG
- COUNT(*)
- COUNT(DISTINCT col name)

SUM

- Calculate the total value for a given column or arithmetic expression
  SELECT SUM*(SALARY)* ***FROM*** *EMP*

MAX and MIN

- Display either the highest or lowest value found in the rows selected by the query.
  SELECT MAX*(SALARY),***MIN***(SALARY)* ***FROM*** *EMP*

AVG

- Display the average of all non-NULL columns
  SELECT AVG*(SALARY)* ***FROM*** *EMP*
- AVG disregards any rows with NULLs in any column involved in the averaging expression
- Integer columns are not rounded, so result may be inaccurate.
- To convert an integer column to a decimal column multiply by 1.0 as part of the AVG expression.
  AVG*(1.0 * STATUS)*

COUNT(*)

- It counts all the rows selected by a query
  SELECT COUNT(*) FROM *SUPPLIERS*

COUNT(DISTINCT col name)

- It counts only those rows that have a unique value in the specified column
  SELECT COUNT(DISTINCT *STATUS****) FROM*** *SUPPLIERS*
- COUNT (*) counts all the rows selected
- COUNT(DISTINCT col name) does not count rows with a duplicate value in the specified column.
- COUNT(DISTINCT col_name) does not count rows that have NULLS in the specified column.

HOW DOES SQL CREATES SUMMARY DATA

- When you request a built-in summary functions, SQL goes through several steps to reach a result.
- First, all the rows and columns are selected. This intermediate result is kept in a temporary table.
- Next, the system process your GROUP BY request, creating a temporary table for each GROUP and putting the appropriate rows in each table.
- Then the built-in functions are calculated.
- If, for example you requested an AVG, then the average for each temporary sub group is calculated.

GROUP BY

- GROUP BY outputs a summary table – one line for each group.
- GROUP BY can only be used in conjunction with built-in functions.
- GROUP BY follows FROM and WHERE and precedes ORDER BY
- Example: ***SELECT*** *DEPT,* ***MAX****(SALARY)* ***FROM*** *EMP* ***GOUP BY*** *DEPT*
- It will produce a table listing the maximum salary value for each department.

HAVING

- It is similar in function to the WHERE clause
- Its purpose is to further define a 'group' specified in the GROUP BY statement ***SELECT*** *DEPT,* ***AVG****(SALARY)* ***FROM*** *EMP* ***GROUP BY*** *DEPT* ***HAVING*** *COUNT(*)>3*
- HAVING can only be used in conjunction with GROUP BY
- HAVING must immediately follow the GROUP BY clause
- HAVING can only compare built-in functions not individual columns

JOINING

- Joining tables involves matching the rows of one table with related rows of another table to produce a table consisting of rows with columns from both the original tables.

BASIC PRINCIPLES OF JOINING TABLES

- It will combine related information from two tables into a single row.

- If a value appears once in the “joining column” of one table, but several times in the other, the higher number of occurrences will appear in the output table.
- If a value exists in the “joining column” of one table, but not in the other, no row appears in the output table with that value.

JOINING COLUMN

- It is a column that is common to all the tables being joined and contains the row values that tie one table to next.

HOW DO YOU ACTUALLY TIE THE TABLES TOGETHER?

- With a special type of WHERE clause called the joining condition
  WHERE EMP.DEPTNO = DEPT.DEPTNO
- Example: ***SELECT*** *ITEM,PRICE,COLOR* ***FROM*** *Table1,Table2* ***WHERE*** *Table1.ITEM = Table2.ITEM*

MERGING

- Merging tables also involves combining data from two tables, but the rows are not joined together.
- Example:***SELECT*** *** ***FROM*** *EMP* ***UNION SELECT*** *** ***FROM*** *SUPPLIER*

MERGING QUERY CODING

- Select data from one table
- Specify UNION or UNION ALL to indicate MERGE
- Select data from second table
- Example: ***SELECT*** *** ***FROM*** *Table_1* ***UNION SELECT*** *** ***FROM*** *Table_2*
- The UNION or UNION ALL keyword must be entered after the first SELECT statement

UNION vs. UNION ALL

- UNION sorts the result tables, UNION ALL does not. In other words, UNION mixes rows together in the output. UNION ALL outputs all the rows from the first table that meet the search condition. UNION removes duplicate rows, UNION ALL does not.

WHEN TO MERGE TWO TABLES

- Each column selected from first table must be compatible with the corresponding columns from the second table.
- A numeric column are compatible with numeric column, a character column to a character column, and so on.
- Numeric data types need not be the same column length You must select the same number of columns from each tables.

CREATING AND UPDATING TABLES

- To create a new table you must use the SQL statement CREATE TABLE. You must specify the table name and the names and attributes of each column.

RULES FOR TABLE NAME

- Maximum 18 characters
- Permissible characters
  - letters of the alphabet
  - @, #, and $
  - numbers
  - underscore (_)
- The first character must be alphabetic or @,#, or $

DATA TYPES

- Data type depends on the nature of the data itself and will always be one of two types – NUMERIC or CHARACTER
  - Types of CHARACTER column
    - CHAR – Allows any values to be entered in the column. All entries are the same length
    - VARCHAR – Allows any values to be entered in the column. All entries are the varying length
  - Types of NUMERIC columns
    - INTEGER - For very large integers up to 2,147,483,647
    - SMALLINT - For small integer up to 99999 Table Space
- DECIMAL - For numbers with a fixed number of decimal places after the decimal point. The number can have a total of 15 digits
- FLOAT -For very large numbers with an undetermined number of decimal places after the decimal point
- Types of DATE/TIME columns
  - DATE - Dates are stored in YYYYMMDD format
  - TIME - Times are stored on a 24-hour clock in HHMMSS format
- TIMESTAMP – TIMESTAMP columns contain both date and time information, with the time value accurate to the millisecond.

GRANT & REVOKE

- Privileges in DB2 to control accessing data
- To grant access to a table or view you must run a query that specifies three things in the following order
  - The privileges you are granting (SELECT,INSERT,DELETE,UPDATE or ALL)
  - The name of the table or view
  - The user or users to whom you are granting access STOGROUP(Storage group)
- To grant everything, use GRANT ALL
- To grant a privilege to EVERYBODY , you grant TO PUBLIC

PRIVILEGES

- *GRANT ALL ON TABLE EMP TO PUBLIC*
- *REVOKE ALL ON EMP TO PUBLIC*

CAN YOU CHANGE A TABLE OR VIEW AFTER YOU CREATE IT?

- The only change you can make to a table or view after it has been created is to add an additional column.
- This is done with the SQL command ALTER
  - *ALTER TABLE EMP ADD COMM DECIMAL (7 ,2)*

DROP

- When you no longer need a TABLE, VIEW, or SYNONYM you can DROP them.
  - DROP TABLE SUPPLIER
  - DROP VIEW EMP_VIEW
  - DROP SYNONYM S

INSERTING A TABLE

- There must be one value for each column
- Separate values with commas
- Put character values in single quotes
- Numeric values do not require quotes
- If you want enter the values in a different order, there is an alternative way of inserting a row, by listing the columns along with their assigned values
  - ***INSERT INTO*** *SUPPLIER (S#,STATUS,SNAME)* ***VALUES****('S1',30,'VAST')*

UPDATE

- SQL provides two commands for updating row values – UPDATE and SET.
- UPDATE tells SQL which table you want to update and SET provides the name of the column to update and the new value for the column.
- You tell SQL which row to update with a WHERE clause can be very specific, so that a single row gets changed, or it can be very general, so that many rows get updated.
  - *<B>UPDATE SUPPLIER* ***SET*** *STATUS = 40* ***WHERE*** *S# = 'S1'*
  - ***UPDATE*** *EMP* ***SET*** *SALARY = SALARY + SALARY *.01*

DELETE

- Using the DELETE command you can delete rows and using WHERE clause you can specify which rows you want to delete.
  - ***DELETE FROM*** *SUPPLIER* ***WHERE*** *S# = 'S1'*

Complex SQL's

- One terms a SQL to be complex when data that is to be retrieved comes from more than one table.
- SQL provides two ways of coding a complex SQL
    - Subqueries and Join

SUB QUERIES

- Nested select statements.
- Specified using the IN or NOT IN predicate, equality or non-equality predicate and comparative operator.
- When using the equality, non-equality or comparative operators the inner query should return only a single value.
- Nested loop statements gives the user flexibility for querying multiple tables.
- A specialized form is a correlated sub query. It works on top-bottom-top fashion.
- Non correlated sub query works in bottom to top fashion.

JOINS

- OUTER JOIN
- INNER JOIN

OUTER JOIN

- For one or more tables being joined both matching and the non-matching rows are returned.
- Duplicate columns may be eliminated.
- Non-matching columns will have nulls.

INNER JOIN

- Here there is a possibility one or more of the rows from either or both tables being joined will not be included in the table that results from the join operation

QMF- Query Management Facility

- It is an MVS and VM-based query tools
- Allows end users to enter SQL queries to produce a verity of reports and graphs as a result of this query.
- QMF queries can be formulated in several ways: by direct SQL statements, by means of relational prompted query interface

SPUFI (SQL Processing Using File Input)

- Supports the on-line execution of SQL statements from a TSO terminal.
- Used for developers to check SQL statements or view table details.

- SPUFI menu contains the input file in which the SQL statements are coded , option for default settings and editing and output file.

Program Preparation

- Precompile
- Compile & Link
- Bind
    - Package
    - Plan

Precompile

- Searches all the SQL statements and DB2 related INCLUDE members and comments out every SQL statement in the program.
- The SQL statements are replaced by a CALL to the DB2 runtime interface module along with parameter.
- All SQL statements are extracted and put in a DBRM.
- Places a time stamp in the modified source and the DBRM so that these are tied.
- All DB2 related INCLUDE statements must be placed between EXEC SQL & END-EXEC key words for the precompiler to recognize them.

Compile & Link

- Modified precompiler COBOL output is complied.
- Complied source is link edited to an executable load module.

Bind

- A type of compiler for SQL statement.
- It reads the SQL statements from the DBRM and produces a mechanism to access data as directed by the SQL statements being bound.
- Checks syntax, check for correctness of table & column definitions against the catalog information & performs authorization validation.

Package

- It is a single bound DBRM with optimized access paths.
- It also contains a location identifier a collection identifier and a package identifier.
- A package can have multiple versions , each with it’s own version identifier.

Advantages of Package

- Reduced bind time.
- Versioning.
- Provides remote data access.
- Can specify bind options at the programmer level.

Plan

- An application plan contains one or both of the following elements
    - A list of package names.
    - The bound form of SQL statements taken from one or more DBRM.
- Every DB2 application requires an application plan.
- Plans are created using the DB2 sub commands BIND PLAN.

DB2 Optimizer

- Analyzes the SQL statements and determines the most efficient way to access data, gives physical data independence.
- It evaluates the following factors: CPU cost, I/O cost, DB2 catalogue statistics & the SQL statements
- It estimate CPU time , cost involved in applying predicates traversing pages & sorting.
- It estimates the cost of physically retrieving and writing data.

Steps involved in creating a DB2 Application

- Using Embedded SQL
- Using Host Variables(DCLGEN)
- Using SQLCA
- Precompile
- Compile & Linkedit the program
- Bind

Embedding SQL Statements

- It is like the file I/O
- Normally the embedded SQL statements contain the host variables coded with the INTO clause of the SELECT statement.
- They are delimited with ***EXEC SQL*** *&* ***END-EXEC***
- Example: EXEC SQL

```
    SELECT eno,ename INTO :h-eno,:h-ename
    FROM employee WHERE eno=1001
END-EXEC.
```

Host Variables

- These are variables defined in the host language to use the predicates of a DB2 table. These are referenced in the SQL statement.
- A means of moving data from and to DB2 tables.
- DCLGEN produces host variables the same as the columns of the tables.

DCLGEN

- Issued for a single table.

- Prepares the structures of the table in a COBOL copy book.
- The copy book contains a SQL DECLARE TABLE statement along with a working storage host variable definition for the table.

SQLCA

- An SQLCA is a structure or collection of variables that is updated after each SQL statement executes.
- An application program that contains executable SQL statements must provide exactly one SQLCA.

CURSOR

- Used when more than one row are to be selected.
- Can be used for modifying data using '**FOR UPDATE OF**' clause.

The Four Cursor control Statements

- DECLARE : name assigned for a particular SQL statement.
- OPEN: Builds the result table .
- FETCH : Returns data from the results table one row at a time and assign the value to the specified host variables.
- CLOSE: Releases all resources used by the cursor.

DECLARE:
```
EXEC SQL
    DECLARE empcur CURSOR FOR
    SELECT empno, ename, dept, job
    FROM emp WHERE dept='D11'
    FOR UPDATE OF job
END-EXEC.
```

OPEN: for the OPEN statement
```
EXEC SQL
    OPEN empcur
END-EXEC.
```

FETCH: for the FETCH statement
```
EXEC SQL
    FETCH empcur
    INTO :empno, :ename, :dept, :job     END-EXEC.
```

WHENEVER: for the WHENEVER clause
```
EXEC SQL
    WHENEVER NOT FOUND
    GO TO close-empcur
END-EXEC.
```

UPDATE: for the update statement using cursors

```
EXEC SQL
    UPDATE emp
    SET job=:new-job
    WHERE CURRENT OF empcur
END-EXEC.
```

DELETE: for the delete statement using cursor.

```
EXEC SQL
    DELETE FROM EMP
    WHERE CURRENT OF empcur
END-EXEC.
```

DB2 LOCKING

- Locking is used to provide multiple user access to the same system.
- DB2 uses locking services provided by an MVS subsystem called IMS Resource Locking Manager(IRLM).

Explicit Locking Facilities

- The SQL statement LOCK TABLE
- The ISOLATION parameter on the BIND PACKAGE command – the two possible values are RR(Repeatable Read) & CS (Cursor Stability).
- CS is the value specified if the application program is used in an on-line environment.
- The LOCKSIZE parameter – physically DB2 locks data in terms of pages or tables or table spaces. This parameter is specified in CREATE or ALTER table space option 'LOCKSIZE'. The options are Tablespace, Table, Page or Any.
- The ACQUIRE / RELEASE parameters on the BIND PLAN command specifies when table locks to be acquired and release.
    - ACQUIRE USE / ALLOCATE
    - RELESE COMMIT / DEALLOCATE

Catalogue Tables

- Repository for all DB2 objects – contains 43 tables
- Each table maintains data about an aspects of the DB2 environment
- The data refers to information about table space, tables, indexes etc…
- SYSIBM.SYSTABLES, SYSIBM.SYSINDEXS, SYSIBM.SYSCOLUMNS etc…

DB2 Utilities:-

UTILITIES

- CHECK
- COPY
- MERGECOPY
- RECOVER
- LOAD
- REORG
- RUNSTATS
- EXPLAIN

CHECK

- Checks the integrity of DB2 data structures
- Checks the referential integrity between two tables and also checks DB2 indexes for consistency.
- Can delete invalid rows and copies them to a exception table.

COPY

- Used to create an image copy for the complete table space or a portion of the table space- full image copy or incremental image copy.
- Every successful exception of COPY utility places in the table, SYSIBM.SYSCOPY, at least one row that indicates the status of the image copy.

MERGECOPY

- The MERGECOPY utility combines multiple incremented image copy data sets into a new full or incremental image copy dataset.

RECOVER

- Restore Db2 table spaces and indexes to a specific instance.
- Data can be recovered for a single page, pages that contain I/O errors, a single partition or an entire tablespace.
- Standard unit of recovery is table space.

LOAD

- To accomplish bulk inserts into DB2 table
- Can replace the current data append to it
- If a job terminates in any phase of LOAD REPLACE, the utility has to be terminated and rerun

REORG

- To reorganize DB2 tables and indexes and there by improving their efficiency of access re-clusters data, resets free space to the amount specified in the ‘create DDL’ ,statement and deletes and redefines underlying VSAM datasets for stogroup defined objects.

EXPLAIN

- Explain can be used to obtain the details about the access paths chosen by the DB2 optimizer for SQL statements.
- Used specifically for performance monitoring.
- When EXPLAIN is requested the access paths that the DB2 chooses are put in coded format into the table PLAN_TABLE, which is created in the default database by the user.
- The other method is specifying EXPLAIN YES with BIND command
- The PLAN_TABLE is to be queried to get the required information
- Since the EXPLAIN results are dependent on the DB2 catalogue, it is better to run RUNSTAT before running EXPLAIN.

Db2 FAQ's frame:-

DB2 FAQS

1. How would you find out the total number of rows in a table?
   - Use SELECT COUNT(*) ...
2. How do you eliminate duplicate values in SELECT?
   - Use SELECT DISTINCT ...
3. How do you select a row using indexes?
   - Specify the indexed columns in the WHERE clause.
4. What are aggregate functions?
   - Built-in mathematical functions for use in SELECT clause.
5. How do you find the maximum value in a column?
   - Use SELECT MAX(...)
6. Can you use MAX on a CHAR column?
   - Yes.
7. My SQL statement SELECT AVG(SALARY) FROM EMP yields inaccurate results. Why?
   - Because SALARY is not declared to have NULLs and the employees for whom the salary is not known are also counted.
8. How do you retrieve the first 5 characters of FIRSTNAME column of EMP table?
   - SELECT SUBSTR(FIRSTNAME,1,5) FROM EMP;
9. How do you concatenate the FIRSTNAME and LASTNAME from EMP table to give a complete name?
   - SELECT FIRSTNAME ¦¦ ' ' ¦¦ LASTNAME FROM EMP;
10. What is the use of VALUE function?
    - Avoid -ve SQLCODEs by handling nulls and zeroes in computations
    - Substitute a numeric value for any nulls used in computation
11. What is UNION,UNION ALL?
    - UNION : eliminates duplicates
      UNION ALL: retains duplicates
      Both these are used to combine the results of different SELECT statements.
12. Suppose I have five SQL SELECT statements connected by UNION/UNION ALL, how many times should I specify UNION to eliminate the duplicate rows?
    - Once.
13. What is the restriction on using UNION in embedded SQL?
    - It has to be in a CURSOR.
14. In the WHERE clause what is BETWEEN and IN?
    - BETWEEN supplies a range of values while IN supplies a list of values.
15. Is BETWEEN inclusive of the range values specified?
    - Yes.
16. What is 'LIKE' used for in WHERE clause? What are the wildcard characters?
    - LIKE is used for partial string matches. '%' (for a string of any character) and '_' (for any single character) are the two wild card characters.
17. When do you use a LIKE statement?
    - To do partial search e.g. to search employee by name, you need not specify the complete name; using LIKE, you can search for partial string matches.
18. What is the meaning of underscore ('_') in the LIKE statement?

- Match for any single character.

19. What do you accomplish by GROUP BY ... HAVING clause?
    - GROUP BY partitions the selected rows on the distinct values of the column on which you group by.
    - HAVING selects GROUPs which match the criteria specified
20. Consider the employee table with column PROJECT nullable. How can you get a list of employees who are not assigned to any project?
    - SELECT EMPNO
      FROM EMP
      WHERE PROJECT IS NULL;
21. What is the result of this query if no rows are selected:
    - SELECT SUM(SALARY)
      FROM EMP
      WHERE QUAL='MSC';
      NULL
22. Why SELECT * is not preferred in embedded SQL programs?
    - For three reasons:
      1. If the table structure is changed (a field is added), the program will have to be modified.
      2. Program might retrieve the columns which it might not use, leading on I/O over head.
      3. The chance of an index only scan is lost.
23. What are correlated sub queries?
    - A sub query in which the inner (nested) query refers back to the table in the outer query. Correlated sub queries must be evaluated for each qualified row of the outer query that is referred to.
24. What is a cursor? why should it be used?
    - Cursor is a programming device that allows the SELECT to find a set of rows but return them one at a time.
    - Cursor should be used because the host language can deal with only one row at a time.
25. How would you retrieve rows from a DB2 table in embedded SQL?
    - Either by using the single row SELECT statements, or by using the CURSOR.
26. Apart from cursor, what other ways are available to you to retrieve a row from a table in embedded SQL?
    - Single row SELECTs.
27. How do you specify and use a cursor in a COBOL program?
    - Use DECLARE CURSOR statement either in working storage or in procedure division (before open cursor), to specify the SELECT statement. Then use OPEN, FETCH rows in a loop and finally CLOSE.
28. What happens when you say OPEN CURSOR?
    - If there is an ORDER BY clause, rows are fetched, sorted and made available for the FETCH statement. Other wise simply the cursor is placed on the first row.
29. Is DECLARE CURSOR executable?
    - No.
30. Can you have more than one cursor open at any one time in a program?
    - Yes.

31. When you COMMIT, is the cursor closed?
    - Yes.
32. How do you leave the cursor open after issuing a COMMIT? (for DB2 2.3 or above only)
    - Use WITH HOLD option in DECLARE CURSOR statement. But, it has not effect in psuedo-conversational CICS programs.
33. Give the COBOL definition of a VARCHAR field.
    - A VARCHAR column REMARKS would be defined as follows:

```
10 REMARKS.

    49 REMARKS-LEN    PIC S9(4) USAGE COMP.

    49 REMARKS-TEXT   PIC X(1920).
```

34. What is the physical storage length of each of the following DB2 data types: DATE, TIME, TIMESTAMP?

DATE:	4bytes
TIME:	3bytes
TIMESTAMP:	10bytes

35. What is the COBOL picture clause of the following DB2 data types: DATE, TIME, TIMESTAMP?

DATE:	PIC X(10)
TIME :	PIC X(08)
TIMESTAMP:	PIC X(26)

36. What is the COBOL picture clause for a DB2 column defined as DECIMAL(11,2)?
    - PIC S9(9)V99 COMP-3.
      Note: In DECIMAL(11,2), 11 indicates the size of the data type and 2 indicates the precision.
37. What is DCLGEN?
    - DeCLarations GENerator: used to create the host language copybooks for the table definitions. Also creates the DECLARE table.
38. What is JOIN and different types of JOIN.
    - The ability to join rows and combine data from two or more tables is one of the most powerful features of relational system. Three types of joins:
        1. Equi-join
        2. Non-equijoin
        3. self-join
39. Can I alter a table (e.g. adding a column) when other user is selecting some Columns or updating some columns from the same table?

  - yes possible. until the updation or selection is committed db2 table will not be restructured. new column definition will be there but it will not be included until all the tasks on the table are committed.

40. What are the different methods of accessing db2 from tso?
    - There are three ways in establishing tso/db2 connection
        1. SPUFI
        2. QMF
        3. CATALOG VISIBILITY
41. How is the connection established between TSO & DB2?
    - A thread between TSO & DB2 is established while attempting to make Connection between tso & db2.
42. What is sqlcode -922?
    - Authorization failure
43. How do you do the EXPLAIN of a dynamic SQL statement?
    - Use SPUFI or QMF to EXPLAIN the dynamic SQL statement
    - Include EXPLAIN command in the embedded dynamic SQL statements
44. How is a typical DB2 batch pgm executed?
    - Use DSN utility to run a DB2 batch program from native TSO. An example is shown:

```
DSN SYSTEM(DSP3)
RUN PROGRAM(EDD470BD) PLAN(EDD470BD)
LIB   ('EDGS01T.OBJ.LOADLIB')
END
```

    - Use IKJEFT01 utility program to run the above DSN command in a JCL.
45. Is it mandatory to use DCLGEN? If not, why would you use it at all?
    - It is not mandatory to use DCLGEN.
      Using DCLGEN, helps detect wrongly spelt column names etc. during the pre-compile stage itself (because of the DECLARE TABLE).
      DCLGEN being a tool, would generate accurate host variable definitions for the table reducing chances of error.
46. Name some fields from SQLCA.
    - SQLCODE, SQLERRM, SQLERRD
47. How does DB2 determine what lock-size to use?
    - Based on the lock-size given while creating the table space
    - Programmer can direct the DB2 what lock-size to use
    - If lock-size ANY is specified, DB2 usually choses a lock-size of PAGE
48. What is the difference between CS and RR isolation levels?
    - CS: Releases the lock on a page after use
    - RR: Retains all locks acquired till end of transaction
49. Where do you specify them?
    - ISOLATION LEVEL is a parameter for the bind process.
50. How do you simulate the EXPLAIN of an embedded SQL statement in SPUFI/QMF? Give an example with a host variable in WHERE clause.

- Use question mark in place of a host variable (or an unknown value). e.g.

```
SELECT EMP_NAME
FROM EMP
WHERE EMP_SALARY > ?
```

51. What is ACQUIRE/RELEASE in BIND?
    - Determine the point at which DB2 acquires or releases locks against table and Table spaces, including intent locks.
52. In SPUFI suppose you want to select max. of 1000 rows, but the select returns only 200 rows. What are the 2 sqlcodes that are returned?
    - 100 (for successful completion of the query), 0 (for successful COMMIT if AUTOCOMMIT is set to Yes).
53. How would you print the output of an SQL statement from SPUFI?
    - Print the output data set.
54. What does it mean if the null indicator has -1, 0, -2?
    - -1 : the field is null
    - 0 : the field is not null
    - -2 : the field value is truncated
55. How do you retrieve the data from a nullable column?
    - Use null indicators. Syntax ... INTO :HOSTVAR:NULLIND
56. What else is there in the PLAN apart from the access path?
    - PLAN has the executable code for the SQL statements in the host program
57. What is lock escalation?
    - Promoting a PAGE lock-size to table or table space lock-size when a transaction has aquired more locks than specified in NUMLKTS. Locks should be taken on objects in single table space for escalation to occur.
58. When is the access path determined for dynamic SQL?
    - At run time, when the PREPARE statement is issued.
59. What are the various locks available?
    - SHARE, EXCLUSIVE, UPDATE
60. What is sqlcode -811?
    - SELECT statement has resulted in retrieval of more than one row.
61. What are the advantages of using a PACKAGE?
    - Avoid having to bind a large number of DBRM members into a plan
    - Avoid cost of a large bind
    - Avoid the entire transaction being unavailable during bind and automatic rebind of a plan
    - Minmize fallback complexities if changes result in an error.
62. What is REORG? When is it used?
    - REORG reorganizes data on physical storage to re-cluster rows, positioning oveflowed rows in their proper sequence, to reclaim space, to restore free space. It is used after huge updates, inserts and delete activity and after segments of a segmented table space have become fragmented.
63. How does DB2 store NULL physically?
    - as an extra-byte prefix to the column value. physically, the nul prefix is Hex '00' if the value is present and Hex 'FF' if it is not

64. What is CHECK PENDING?
    - When a table is LOADed with ENFORCE NO option, then the table is left in CHECKPENDING status. It means that the LOAD utility did not perform constraint checking.
65. When do you specify the isolation level? How?
    - During the BIND process. ISOLATION (CS/RR)...
66. What is a DBRM, PLAN?
    - DBRM: DataBase Request Module, has the SQL statements extracted from the host language program by the pre-compiler.
    - PLAN: A result of the BIND process. It has the executable code for the SQL statements in the DBRM.
67. Is DECLARE TABLE in DCLGEN necessary? Why it used?
    - It not necessary to have DECLARE TABLE statement in DCLGEN. This is used by the pre-compiler to validate the table-name, view-name, column name etc., during pre-compile.
68. What do you need to do before you do EXPLAIN?
    - Make sure that the PLAN_TABLE is created under the AUTHID.
69. What is a collection?
    - a user defined name that is the anchor for packages. It has not physical existence. Main usage is to group packages.
70. How can you quickly find out the # of rows updated after an update statement?
    - Check the value stored in SQLERRD(3).
71. What is EXPLAIN?
    - EXPLAIN is used to display the access path as determined by the optimizer for a SQL statement. It can be used in SPUFI (for single SQL statement) or in BIND step (for embedded SQL).
72. Where is the output of EXPLAIN stored?
    - In userid.PLAN_TABLE
73. Suppose I have a program which uses a dynamic SQL and it has been performing well till now. Off late, I find that the performance has deteriorated. What happened?
    - Probably RUN STATS is not done and the program is using a wrong index due to incorrect stats. Probably RUNSTATS is done and optimizer has chosen a wrong access path based on the latest statistics.
74. What are the contents of a DCLGEN? - GS
    - EXEC SQL DECLARE TABLE statement which gives the layout of the table/view in terms of DB2 data types.
    - A host language copy book that gives the host variable definitions for the column names.
75. Will pre-compile of an DB2-COBOL program bomb, if DB2 is down?
    - No. Because the pre-compiler does not refer to the DB2 catalogue tables.
76. What are the isolation (also called isolation parameters) levels possible?
    - CS: Cursor Stability
    - RR: Repeatable Read
77. What is the picture clause of the null indicator variable?
    - S9(4) COMP.
78. EXPLAIN has output with MATCHCOLS = 0. What does it mean?
    - A non-matching index scan if ACCESSTYPE = I.
79. I use CS and update a page. Will the lock be released after I've done with that page?

- No.

80. What are the various locking levels available?
    - PAGE, TABLE, TABLESPACE
81. What is ALTER?
    - SQL command used to change the definition of DB2 objects.
82. Can I use LOCK TABLE on a view?
    - No. To lock a view, take lock on the underlying tables.
83. What are the disadvantages of PAGE level lock?
    - High resource utilization if large updates are to be done
84. What are PACKAGES ?
    - They contain executable code for SQL statements for one DBRM.
85. Lot of updates have been done on a table due to which indexes have gone haywire. What do you do?
    - Looks like index page split has occured. DO a REORG of the indexes.
86. What is dynamic SQL?
    - Dynamic SQL is a SQL statement created at program execution time.
87. What is IMAGECOPY ?
    - It is full backup of a DB2 table which can be used in recovery.
88. What is QUIESCE?
    - A QUIESCE flushes all DB2 buffers on to the disk. This gives a correct snapshot of the database and should be used before and after any IMAGECOPY to maintain consistency.
89. What does the sqlcode of -818 pertain to? - GS
    - This is generated when the consistency tokens in the DBRM and the load module are different.
90. What happens to the PLAN if index used by it is dropped?
    - Plan is marked as invalid. The next time the plan is accessed, it is rebound.
91.
    -
92. What is FREEPAGE and PCTFREE in TABLESPACE creation?
    - PCTFREE: percentage of each page to be left free
    - FREEPAGE: Number of pages to be loaded with data between each free page
93. Are views updatable?
    - Not all of them. Some views are updatable e.g. single table view with all the fields or mandatory fields. Examples of non-updatable views are views which are joins, views that contain aggregate functions (such as MIN), and views that have GROUP BY clause.
94. What is COPY PENDING status?
    - A state in which, an image copy on a table needs to be taken, in this state, the table is available only for queries. You cannot update this table. To remove the COPY PENDING status, you take an image copy or use REPAIR utility.
95. What is an inner join, and an outer join?
    - Inner Join: combine information from two or more tables by comparing all values that meet the search criteria in the designated column or columns of one table with all the values in corresponding columns of the other table or tables. This kind of join which involve a match in both columns are called inner joins.

- Outer join is one in which you want both matching and non matching rows to be returned. DB2 has no specific operator for outer joins, it can be simulated by combining a join and a correlated sub query with a UNION.

96.
   -

97. What is the difference between primary key & unique index?
   - Primary: a relational database constraint. Primary key consists of one or more columns that uniquely identify a row in the table. For a normalized relation, there is one designated primary key.
   - Unique index: a physical object that stores only unique values. There can be one or more unique indexes on a table.
98. When do you use the IMAGECOPY?
   - To take routine backup of tables.
   - After a LOAD with LOG NO. After REORG with LOG ON.
99. What is RUNSTATS?
   - A DB2 utility used to collect statistics about the data values in tables which can be used by the optimizer to decide the access path. It also collects statistics used for space management. These statistics are stored in DB2 catalog tables.
100. How do you insert a record with a nullable column?
   - To insert a NULL, move -1 to the null indicator
   - To insert a valid value, move 0 to the null indicator

Changman frame:-

CHANGEMAN

Change Man is a system that manages and automates the process of migrating software, or applications, from a development environment to a production environment. Change Man can manage the installation of a variety of components, such as:

- JCL, PROC's, AND UTILITY CONTROL CARDS
- Copy code used to compile the programs
- Batch programs written in Assembler, COBOL, COBOL II, or PL/I
- CICS programs written in Assembler, COBOL, COBOL II, or PL/I
- Programs written in Easytrieve, etc.,

What is a Change Package?

A Change Man Change Package is the means by which a new application or change to an existing application is described, scheduled, prepared, and implemented into production. It may consist of one or more library components. Each package is identified by a unique Change Package ID consisting of the Application Mnemonic and a sequential number (e.g. HFC002939). All changes are referenced in the Package Master, which is Change Man's control file. It is a VSAM file that holds all significant data about current and previous Change Packages.

Change Man Library Environment

Change Man manages the transition from development to production libraries by using staging libraries, which are specific to each Change Package. Existing production components are copied into the development environment where they can be modified and tested. The Change Man staging libraries can be used for both editing and testing. If private libraries are used during editing, components will be copied into the staging libraries prior to testing.

NOTE: The baseline libraries contain a current production version (baseline 0) and up to 999 (max.) previous versions of the production component (baseline -1, -2, etc.). When a new production component is installed, the baseline libraries are rippled. This means that the new version becomes the current version, what was the current version becomes the –1 version, and so on.

Brief Overview of Change Man Life Cycle

An overview of the Change Man life cycle is given below. It is a step-by-step description of the actions necessary to migrate a change from development to production within Change Man.

- The first step is to Create a Change Package. A Change Package contains all of the elements to be edited and installed into production, and is identified by a unique package ID automatically generated by Change Man. When a Change Package is created, the information that Change Man needs in order to track and control the package is entered. This includes the implementation instructions, whether it is a temporary or permanent change, the installation date and time, etc.

- The typical next step is to Checkout components from baseline. With checkout, components from your baseline libraries are copied to either a Change Man staging library or to a personal development library where changes can be made.

- Changes may now be Edited in either the Change Man staging or in the development libraries.

- Once editing is completed, the components are ready to Staged. We use the STAGE function within Change Man to edit a component, compile and link edit a program. Components such as copy members are simply copied into the staging libraries, if they aren't there already. The Change Package is now ready for Testing.

- The audit process ensures that no unexpected problems will occur. For example, if a component in the production library has been changed since it was checked out, Change Man alerts you to the problem by creating an out-of-sync condition for the package.

- After successfully passing the audit, the next step is to Freeze the Change Package. This locks the package(prohibiting further changes), and makes the package available for the promotion and approval processes. Promotion allows a Change Package to be moved through various levels of testing (e.g. promote from system testing to acceptance testing).

- Once all of the necessary Approvals have been done, the package is ready to be installed. The installation process depends upon the option used for installation, while creating package.

- Once the package has been installed, Change Man will perform the Baseline Ripple. Baseline Ripple is the process that Change Man executes to version all package components, i.e.: 0 becomes –1, -1 becomes –3, etc. and the new baseline 0 version is installed.

The Change Man Primary Option Menu:

This panel is the full Change Man Primary Option Menu:

```
--------------- CHANGE MAN 4.1.0P PRIMARY OPTION MEN     NOTIFICATION UPDATED

 OPTION  ===>


  1  Build    - Create, update and review package data

  2  Freeze   - Freeze or unfreeze a package

  3  Promote  - Promote or demote a package

  4  Approve  - Approve or reject a package

  5  List     - Display (to process) package list

  C  CDF      - Concurrent Development Facility

  D  Delete   - Delete or undelete a package

  L  Log      - Browse the activity log

  N  Notify   - Browse the Global Notification File

  O  OFMlist  - Online Forms package list

  Q  Query    - Query packages, components and relationships

  R  Revert   - Revert a package to DEV status

  T  Tutorial - Display information about Change Man

  X  Exit     - Exit Change Man


 Press ENTER to process; enter END command to exit.
```

The Change Man Build Change Package Menu:

This panel is displayed when option 1 is selected from the   change man primary option menu.

```
------------------------------- BUILD OPTIONS -------------------------------

 OPTION  ===>


  0  Dates     - Display the installation calendar

  1  Create    - Create a new package

  2  Update    - Update package information

  3  Custom    - Create, update, approve or review custom forms

  4  Utility   - Rename and Scratch information

  5  Checkout  - Check out components from baseline or promotion

  6  Stage     - Stage, edit, browse and delete components

  7  Audit     - Audit a package

  8  Recompile - Recompile source code from baseline or promotion

  9  Relink    - Relink or delete load modules

  B  Browse    - Browse\print\copy baseline or promotion

  C  Compare   - Compare staging to baseline or promotion

  L  Listing   - Browse compressed listings

  S  Scan      - Scan baseline for character strings

  Z  Compress  - Compress change package Staging Libraries


 Press ENTER to process; Enter END command to exit.
```

When you create the Change Package, you are defining the outer structure of the change Package. You specify the date and time for installations, whether the package is temporary or permanent, the installation site, and implementation instructions, etc. Once you have completed the following four panels, the package is automatically assigned a unique Change Package ID.

Create a New Package:

This panel is to allow you to define package control information. This panel is displayed when option 1 is selected form the previous menu (Build Change Package)

Panel 1 of 4.

```
------------------------ CREATE: CREATE A NEW PACKAGE ------------------------

 OPTION ===>

      L  Long method  - Prompt for package description and special instructions

     PACKAGE TITLE ===>

     APPLICATION          ===>          (Blank or pattern for list)

 REQUESTER'S NAME        ===>

 REQUESTER'S PHONE        ===>

 WORK REQUEST ID         ===>

 DEPARTMENT              ===>

 PACKAGE LEVEL           ===>          (1-Simple, 2-Complex,

                                  3-Super,  4-Participating)

 PACKAGE TYPE            ===>          (Planned or Unplanned,

                                  Unplanned is for Emergency Loads )

 PACKAGE TIME SPAN       ===>          (Permanent)

 PACKAGE TO COPY FORWARD   ===>          (Optional package name)

 UNPLANNED REASON CODE     ===>          (* for list)

 TEMPORARY CHANGE DURATION ===>          (In days)
 Press ENTER to process; Enter END command to exit.
```

All the fields are required for the first time you create a Change Package. SO Enter the Relevant data and press Enter.

General Information:

This panel allows you to enter a description of the Change Package. It should be as complete and descriptive as possible.

Panel 2 of 4

```
------------------------ CREATE: PACKAGE DESCRIPTION ------ Row 1 to 12 of 12

 COMMAND ===>                                    SCROLL ===> CSR


 Press ENTER or END to continue or type CANCEL to exit.


     (minimum one line, maximum 46 lines)

 '''' ________________________________________________________________________
 '''' ________________________________________________________________________
 '''' ________________________________________________________________________
 '''' ________________________________________________________________________
 '''' ________________________________________________________________________
 '''' ________________________________________________________________________
 '''' ________________________________________________________________________
 '''' ________________________________________________________________________
 '''' ________________________________________________________________________
 '''' ________________________________________________________________________
```

'''' ____

'''' ____

****************************** Bottom of data *******************************

Implementation Instructions:

Use this panel to specify how the package is to be installed. This include the installation method,and specific implementation instructions to be used by in the individual overseeing the installation

Panel 3 of 4

```
--------------------- CREATE: INSTALLATION INSTRUCTIONS --- Row 1 to 12 of 12

 COMMAND ===>                                SCROLL ===> CSR



 Press ENTER or END to continue or type CANCEL to exit.



 CONTINGENCY ===> 1   1-Hold production and contact analyst



 SCHEDULER   ===> CMN      (CMN)


    (minimum one line, maximum 46 lines)

 '''' ______________________________________________________________

 '''' ______________________________________________________________

 '''' ______________________________________________________________

 '''' ______________________________________________________________
```

```
""____________________________________________________________
""____________________________________________________________
""____________________________________________________________
""____________________________________________________________
""____________________________________________________________
""____________________________________________________________
""____________________________________________________________
""____________________________________________________________
```

Create onsite information:

Use this panel to select the install date and time. Ensure that these fields are correct, and modify them or the contacts fields as necessary.

Panel 4 of 4

```
------------------------ CREATE: ON SITE INFORMATION ------------------------

 COMMAND ===>



 Press ENTER to create the package or type CANCEL to exit.



 INSTALL  DATE/TIME

 YYMMDD   FROM   TO    PRIMARY/BACKUP CONTACTS    PHONE NUMBERS

 990316   0400 0415    aftab ullah_______________  630-616-3022___

                       sabbi vamsidar____________  630-350-9483___
 WARNING: "From Install" time of 0200 will automatically set the

          "To   Install" time to 0230 . This is to enforce the 0200

          load time for online modules.
```

After the information in the above panel is filled, press ENTER Change man will then automatically create your change package, return you to the first CREATE panel, and display the change package number in the upper right hand corner of the panel.

The Checkout Process:

Once the package is created next process Is checkout. select option 1.5 from primary option menu. Use this panel to select the list of components to be viewed. Select from components in the baseline libraries, or components already defined to the Change Package from a previously coped package.

```
------------------------------ CHECKOUT OPTIONS -----------------------------

 OPTION  ===>



 PACKAGE ID ===> HFCG003413



   1  Base/Prom - Checkout from baseline or promotion libraries

   2  Package   - Checkout package components



 Press ENTER to process; Enter END command to exit.
```

Checkout:

Use this panel to enter the checkout information, such as the package to which the component is to be checked out, the type of component, how the component is to be checked out(foreground of batch), and the name of the component that is to be checked out, etc.,

```
---------------------------------- CHECKOUT ---------------------------------
```

```
OPTION ===>

  L - List libraries where component is located

PACKAGE ID       ===> HFCG003413 (Package or Application name)

COMPONENT NAME   ===>            (Name or Pattern for list; "ABCDEF" OR "ABC*")
                          Pattern of only "*" Checks Out all Components
                          related to Library Type.
LIBRARY TYPE     ===> CP1        (Blank for list)
LANGUAGE         ===>            (Applies to source code; * for list)
SOURCE LIBRARY   ===> 0          (Baseline 0 to -n; Promotion +1 to +n)

CHECK OUT TO     ===> S          (S-Staging library, P-Personal library)
PERSONAL LIBRARY ===>
LIBRARY DSORG    ===> PDS        (Personal library: PDS, SEQ, PAN, LIB)

CONFIRM REQUEST  ===> YES        (Y/N)
CHECKOUT MODE    ===> 1          (1-Online, 2-Batch)

Press ENTER to process; Enter END command to exit.
```

Stage:

Once the components have been check out, they must be STAGED. you will use the STAGE function within Change Man to edit a component, compile and link edit a program, or to submit another transformation type of process like a JCL scan etc.

Use the panels on the following page to complete the staging process for your Change Package components. Once these panels are completed, the component list will be displayed. From it you may browse ,edit,delete,recompile,display the source/load relationship, or stage a component for a list of valid selection codes.

The Staging Process:

Stage Options:

Use this panel to tell Change Man where to get the components that are to be staged. Choose from a personal library or from existing components within the Change Package option is 1.6

Panel 1 of 1

```
------------------------------- STAGE OPTIONS -------------------------------

 OPTION  ===>



 PACKAGE ID ===> HFCG003413



   1  Dev       - Stage components from development libraries

   2  Package   - Process package components

   3  Parms     - Specify component list selection criteria



 Press ENTER to process; Enter END command to exit.
```

Stage Package Components:

This panel is used to select the components in your change package that you want to edit, compile, delete, display history etc. OPTION 2 from previous screen

```
------------------------ STAGE: HFCG003411 COMPONENTS ----- Row 1 to 12 of 12
 COMMAND ===>                                        SCROLL ===> CSR
     NAME   TYPE  STATUS     CHANGED     LANGUAGE PROCNAME   ID     REQUEST
 __ ETRTDATA CP1  ACTIVE    990403 045047                   VEN214
 __ LCC261   HI3  INCOMP    990405 062822  COBOL370 COBOL370 VEN080
 __ LSADL05  HI3  ACTIVE    990403 042020  COBOL370 COBOL370 VEN214
 __ LSAETAD  HI3  ACTIVE    990403 042045  COBOL370 COBOL370 VEN214
 __ LSAL05P  HI3  ACTIVE    990403 045624  COBOL370 COBOL370 VEN214
 __ LSAL05PC CP1  ACTIVE    990403 045220                   VEN214
 __ LSAL051  HI3  ACTIVE    990403 042233  COBOL370 COBOL370 VEN214
 __ LSAUL05  HI3  ACTIVE    990403 042342  COBOL370 COBOL370 VEN214
 __ LSEL05M  HI3  ACTIVE    990403 042422  COBOL370 COBOL370 VEN214
 __ LSPL05   HI3  ACTIVE    990403 042546  COBOL370 COBOL370 VEN214
 __ LSPL05G  CP1  ACTIVE    990403 045057                   VEN214
 __ LSRL05M  HI3  ACTIVE    990403 042653  COBOL370 COBOL370 VEN214
 ******************************* Bottom of data ********************************
```

Compile and link Edit:

This panel is used to compile and optionally link a source component.

Panel 2 of 3.option is S on the previous menu CMD option.

```
------------------ STAGE: COMPILE AND LINK EDIT / JCL CHECK -----------------
 COMMAND ===>


      PACKAGE ID: HFCG003411      STATUS: DEV      INSTALL DATE: 990420
```

STAGED NAME: LCC261

LIBRARY TYPE: HI3 - IMS Online COBOL LE/370 load to ILM

LANGUAGE: COBOL370

DATASET NAME: D131.CP000000.CMN.HFCG.#003411.HI3

COMPILE PROCEDURE ===> COBOL370 (Blank for list; ? for designated procedure)

COMPILE PARMS OR JOB NAME ===>

(Enter JOB Name for PROCs requiring associated JOB for JCLCHECK)

LINK EDIT PARMS ===>

DB2 PRE-COMPILE ===> NO (Y/N)

OTHER OPTIONS ===> YES (Y to display other options)

JOB STATEMENT INFORMATION:

===> //VEN080U JOB (050D,WD1A),'CHNGMAN COMPARE',CLASS=W,____________________

===>
// MSGCLASS=T,NOTIFY=VEN080____

===> //*____

===> /*ROUTE PRINT PUNE____

Press ENTER to process; Enter END command to exit.

Stage User Options:

This panel is an optional panel that is customized for each customer site.Enter a ‘Y’ for the options that you wish to select. Press Enter on the preious menu.

---------------------------- STAGE: USER OPTIONS ----------------------------

COMMAND ===>

```
NAME: LCC261          TYPE: HI3       LANGUAGE: COBOL370


COMPILE ONLY        ===>         ??? ENTRY TO DLITCBL   ===> Y

CICS PRE-COMPILE    ===>            DROP INCLUDE STMTS ===>

EASYTRIEVE NO NAME  ===>            XPEDITER COMPILE   ===>

FORMAT TO IMSTEST   ===>            INCLUDE MQSERIES   ===>

  (TEST MFS IS DEFAULT)


JCK FULL LISTING    ===>           USER OPTION 10     ===>

USER OPTION 11      ===>           USER OPTION 12     ===>

USER OPTION 13      ===>           USER OPTION 14     ===>

USER OPTION 15      ===>           USER OPTION 16     ===>

USER OPTION 17      ===>           USER OPTION 18     ===>

USER OPTION 19      ===>           USER OPTION 20     ===>


 Use Y or N to select or deselect additional processing options;

 Press ENTER to continue; Enter END command to exit.
```

Package List:

Use the Package List panels as an alternative way to perform a variety of Change Man functions. Use this panel to select the Change Packages to be listed. If the Package Name field is left blank, you can select the application for which you would like to see a package list. Option is 5 from Primary change man screen

```
-------------------------- PACKAGE LIST PARAMETERS --------------------------

 COMMAND ===>
```

```
SPECIFY SELECTION CRITERIA:

PACKAGE ID            ===> HFCG       (Full name or pattern; blank for list)

PACKAGE STATUS         ===>            (DEV, FRZ, APR, REJ, DIS, INS,

                               BAS, BAK, OPN, CLO, TCC or DEL)

CREATOR'S TSO-ID      ===>

WORK REQUEST ID        ===>

DEPARTMENT            ===>

PACKAGE LEVEL         ===>            (1-Simple, 2-Complex,

                               3-Super,  4-Participating)

PACKAGE TYPE          ===>            (Planned or Unplanned)

PACKAGE TIME SPAN      ===>            (Permanent or Temporary)

FROM INSTALL DATE      ===>            (YYMMDD)

  TO INSTALL DATE      ===>            (YYMMDD)

FROM CREATION DATE     ===>            (YYMMDD)

  TO CREATION DATE     ===>            (YYMMDD)

OTHER PARAMETERS       ===>            (Y/N)


Press ENTER to process; Enter END command to exit.
```

The Change Package List:

This panel displays the Change Packages selected from the previous panel . There are 36 different line commands available for this panel, which can be placed on CMD filed.

```
----------------------------- CHANGE PACKAGE LIST ------------ Row 1 to 2 of 2

 COMMAND ===>                                   SCROLL ===> CSR

    PACKAGE ID STA INSTALL LVL TYPE WORK REQUEST DEPT PROMOTION AUD CREATOR

 __ HFCG003193 DEV 990320 SMPL PLN/PRM VEN214                    VEN214

 __ HFCG003411 DEV 990420 SMPL PLN/PRM VEN214                    VEN214

 ******************************* Bottom of data ********************************
```

Utilities:

Recompile:

The Recompile function can be used to resolve some of the out-of-sync conditions that can occur from the Audit process. The recompile is done from baseline or promotion, directly into staging libraries.

```
------------------------------- RECOMPILE SOURCE ------------------------------

 OPTION ===>



   R     - Recompile one component

   L     - List libraries where component is found

   blank - Display member selection list



 PACKAGE ID          ===> HFCG003411

 COMPONENT NAME      ===>            (Blank or pattern for list)
```

SOURCE LIBRARY TYPE ===>   (Blank for list)

LANGUAGE   ===>   (Blank for list)

LIBRARY LEVEL   ===>   (Baseline 0, Promotion 1 to n)

Press ENTER to process; Enter END command to exit.

After Entering the required data press Enter. You would get the Recompile job information screen where the data has to be entered and press ENTER. Your job is submitted.

Relink:

Any time a statically linked subroutine has been modified, relinking is required. Use think function to delete a relinked or recompiled load module from your Change Package.

------------------------- RELINK/DELETE LOAD MODULES -------------------------

OPTION ===>

R   - Relink single load module

D   - Delete relinked/recompiled module

blank - Display LCT member selection list for multiple member processing

PACKAGE ID   ===> HFCG003411

RELINK FROM   ===> S   (S-Staging or B-Baseline)

LCT MEMBER LIST   ===>   (Y - Yes or N - No; member list will be created with input library type)

INPUT LIBRARY TYPE   ===> BLM   (Blank for list)

TARGET LIBRARY TYPE ===> BLM   (Blank for list)

MEMBER NAME   ===>   (Blank for list; required for options R/D)

COMPILE PROCEDURE   ===>   (Blank for list)

LINK EDIT PARMS   ===>

DB2 PRE-COMPILE ===> (Y/N)

OTHER OPTIONS ===> YES (Y/N for additional user options)

JOB STATEMENT INFORMATION:

===> //VEN080V JOB (050D,WD1A),'CHNGMAN COMPARE',CLASS=W,____________________

===>
// MSGCLASS=T,NOTIFY=VEN080____

===> //*____

===> /*ROUTE PRINT PUNE____

Press ENTER to process; Enter END command to exit.

Fileaid :-

FILE AID

GENERAL INFORMATION

Designed for applications and systems programmers, File-AID is an interactive, full-screen system for editing, browsing, defining, copying, reformatting, comparing, and printing VSAM, IAM, ISAM, PDS, BDAM, and sequential files under TSO/ISPF.

KEY FEATURES

File-AID's functions enable you to significantly reduce the time required to perform day-to-day data file manipulation tasks such as creating and modifying test data, resolving production data problems, and performing ad-hoc file conversions and comparisons.

KEY FEATURES:

o Eliminates ISPF editing restrictions

- provides on-line editing of sequential, BDAM, PDS, ISAM, IAM, and VSAM files using

  formatted, full-screen displays

- eliminates record length restriction

  eliminates file size restriction by using selection criteria to limit the number of records to be edited

- supports files in compressed/encrypted format (with I/O exits)

- provides optional audit trail of all records updated, inserted, and deleted while editing a data file

o Uses existing COBOL or PL/I record layouts directly

- the record layout is the sole definition of files thus eliminating the need to redefine the file/data in another language

- superimposing the record layout over raw data brings meaning to that data

- files may be printed in a formatted manner using the record layout o Has powerful, easy to use, copying and reformatting capabilities

- these capabilities exist in both batch and on-line modes

- no language to learn and no programming required

- records may be selected for copying using selection criteria based on record counts and/or specific values in individual fields .

- selection criteria can be used in the Browse, Edit, Copy, Print, or Compare functions .

o Runs as a single dialog under ISPF

- uses menu-driven, full-screen, interactive ISPF-like displays

- has user defined PF (program function) key support

- can operate "split-screen" with ISPF/PDF

- easy to learn and use

- a fully interactive product with batch capabilities

- executes in storage available above the 16 megabyte line on systems running MVS/XA or MVS/ESA

GENERAL INFORMATION INVOKING File-AID

File-AID executes as a dialog under IBM's ISPF dialog manager facility. You can invoke it by selecting the File-AID option from a customized version of the ISPF primary option menu, or from a sub-menu panel your installation has set up.

If the File-AID execution CLISTS have been installed, you may invoke File-AID from any ISPF screen. For example, the command TSO FASTART suspends your current screen and displays the File-AID primary option menu. Upon exit from File-AID your suspended screen is restored.

The following will be presented only if selected by number:

1 - File-AID execution CLISTS

GENERAL INFORMATION SUMMARY OF OPTIONS 0-C

The File-AID primary options are:

OPTION 0 - SPECIFY ISPF AND File-AID PARAMETERS

This option consists of the ISPF-provided parameter option screens and several screens to specify File-AID default parameters.

OPTION 1 - BROWSE FILE

This option is used to display but not change the contents of files using any of three display modes - formatted, vertical, and character.

OPTION 2 - EDIT FILE

This option is used to create, display and change the contents of files using the formatted, vertical, and character display modes.

OPTION 3 - UTILITIES/E

This option accesses a menu of dataset management utilities. Included are:

3.1 Library (PDS directory management and CSECT info),

3.2 Dataset (Non-VSAM dataset allocation, information),

3.3 Copy (Selective record copying for all file types including PDS members),

3.4 Catalog (search catalog using pattern characters)

3.5 VSAM (Define VSAM clusters, indexes, and paths online or in batch),

3.6 Search/Update (Scan or perform global changes for any dataset),

3.7 VTOC (Search volumes for datasets using pattern characters, analyze space),

3.8 Interactive (run File-AID/Batch online),

3.9 Submit (create batch JCL).

OPTION 5 - PRINT FILE

This option is used to print data file contents, the selection criteria and rcd/layout XREFs created in options 6 and 7 of File-AID, formatted record layouts, and audit trail datasets created while editing a data file in option 2 (Edit) or 3.6 (Update) of File-AID.

OPTION 6 - EDIT SELECTION CRITERIA

This option is used to create and maintain selection criteria for use in the Browse, Edit, Copy, Print, and Compare functions.

OPTION 7 - EDIT RCD/LAYOUT XREF

This option is used to create and maintain rcd/layout XREFs for use in other File-AID functions.

OPTION 8 - VIEW RECORD LAYOUT

This option is used to display the contents of COBOL or PL/I layouts as interpreted by File-AID.

OPTION 9 - REFORMAT FILE

This option allows the records of an input file to be reformatted and written to an output file based on record layouts defining the input and output files.

OPTION 10 - COMPARE

This option is used to compare two files and report the differences between them.

OPTION C - File-AID CHANGES

This option documents changes made in the various releases of File-AID.

GENERAL INFORMATION SCREEN FORMATS

File-AID uses the full screen for display and entry of data.

The first three lines of each display, called the heading lines, have a common format for all File-AID displays. The remainder of the screen may contain a list of options, input fields and prompts, or scrollable data.

The first three lines of each display are formatted as follows:

```
             |--------------------------------------|-----------------------   --|

  line 1: |  TITLE                               |  SHORT MESSAGE |

          |-------------------------------------- |------|----------           ---|

  line 2: |  COMMAND/OPTION                           |  SCROLL |

          |-------------------------------------------- --|---------------   ---|

  line 3: |  LONG MESSAGE                                                  |

          |------------------------------------------------------------------|
```

The TITLE area (line 1) identifies the function being performed and, where appropriate, library or dataset information.

The SHORT MESSAGE area (line 1) is used to indicate:

- Current line or column positions
- Successful completion of a processing function
- Error conditions (accompanied by audible alarm)

The COMMAND/OPTION area (line 2) is used to enter a command. On an option selection menu, it may be used to enter either a command or an option.

The SCROLL area (line 2) contains the current scroll amount whenever scrolling is applicable. You may change it by overtyping.

The LONG MESSAGE area (line 3) is used to display an explanation of error conditions upon request (see HELP command). This line will normally be blank or will contain heading (non data entry) information. Dataset specification screens allow you to enter information such as dataset names, member names, and other parameters. The fields in which you may enter information are labeled and preceded with an arrow. If you fail to enter a required value or if you enter inconsistent values, you are prompted with a message.

The example at the right shows the edit dataset specification screen.

```
|*********************************************
| File-AID  -------------  Edit - Dataset Specifi
| COMMAND ===>
|
| Edit Mode                ===> C   (F=Format
|
| Specify Edit Information:
|    Edit Dataset name       ===> FASAMP.*
|    Member name             ===>     (Blank or
|    Volume serial           ===>     (If datas
|    Disposition             ===> OLD (SHR or O
|    Create audit trail      ===> N   (Y = Yes;
```

Several fields on dataset specification screens are pre-entered with values that you entered the last time on that screen or on a similar screen. If the values are correct, simply press ENTER. If the values are not correct, overtype the fields that need to be changed before pressing ENTER.

The pre-entered information comes from your user profile, which File-AID automatically builds and maintains across sessions. Information that is maintained in your user profile includes:

- Dataset names and member names
- Job statement information
- SYSOUT class for printed output
- Parameters entered via Option 0 (Parameters)
- PF-Key defaults

GENERAL INFORMATION MEMBER SELECTION SCREENS

A member selection screen is an alphabetic list of the members of a partitioned dataset. It is displayed when requested from any of the File-AID functions 1-10.

In most functions it is possible to obtain multiple member lists prior to entering the function.

The example at the right shows the selection of member TESTFIL3 from the partitioned dataset PROJECT.TEST.DATA

```
|***********************************************
| File-AID ---- EDIT - PROJECT.TEST.DATA --------
| COMMAND ===>
|    NAME        VER.MOD   CREATED   LAST M
|  TEST1          01.04    08/17/93  09/23/
|  TEST2          01.02    06/12/93  09/13/
| S TESTFIL3      01.13    07/29/93  09/20/
|   TESTFIL4      01.07    08/01/93  08/29/
```

Scrollable data display screens show file contents or record layouts and allow up/down scrolling, and in some cases left/right scrolling. On many scrollable data display screens, you can also update file contents by typing over fields on the screen.

The example below shows the edit formatted screen. You can scroll using the UP and DOWN commands, and can update file contents by typing over the FIELD VALUE column.

```
 File-AID ---- EDIT - PROJECT.TEST.DATA(TESTFL3) -------------------- LINE 00001

 COMMAND ===>                                       SCROLL ===> PAGE

 RCD SEQ NO  1

 --------LEVEL NUMBER/DATA-NAME--------- -OFFSET- ---------FIELD VALUE----------

 01  ORDER-LINE

   05  ITEM-NO                    1    PS/2 MOD50

   05  QTY-ORDERED               11    100

   05  LIST-PRICE                14    2400.00
```

GENERAL INFORMATION UNSCROLLABLE DATA DISPLAY SCREENS

Unscrollable data display screens show "nonrepeating" information which fits on a single screen. The Selection Criteria Options screen shown below is an example of this type of screen.

```
********************************************************************************

 File-AID ---- Selection Criteria - Options -----------------------------------

 COMMAND ===>



 Selection criteria options

                         First, start at the following record KEY

   Starting record key        ===>

     - OR -                            OR at the following RBA
   Starting RBA               ===>          (both blank for start of file


   Initial records to skip    ===> 0       then skip this many records
```

GENERAL INFORMATION COMMAND ENTRY

File-AID provides commands for commonly used functions. You may enter commands in one of two ways:

1) By typing the command in the command/option field (line 2 of the screen) and then pressing the ENTER key, or

2) By pressing a program function (PF) key to which you have assigned the desired command.

You can assign commands to PF keys using option 0.0.3 or the KEYS command. When you press the PF key, the processing is the same as if you typed the command in the command field and pressed ENTER.

Before you press a PF key, you can enter information in the command field. The PF key definition is concatenated ahead of the contents of the command field. For example, suppose you assign the DOWN command to the PF8 key. If you type "6" in the command field and press PF8, the results are the same as if you had typed "DOWN 6" in the command field and pressed ENTER.

You can stack multiple commands for execution in one interaction by entering a special delimiter between the commands. The default delimiter is a semicolon (;). You can change the delimiter using option 0.0.1. For example:

```
  COMMAND ===> DOWN 10;LEFT
```

In this example the DOWN and LEFT commands have been stacked.

GENERAL INFORMATION COMMAND SUMMARY

File-AID primary commands are divided into four categories listed below. Commands designated with an asterisk (*) have the same syntax and function as in ISPF.

In many File-AID functions, the information to be displayed exceeds the screen size. Scroll commands allow you to move the screen "window" in as many as three dimensions across the information: up/down, left/right, and forward/back. For a more detailed description of scrolling, see the SCROLLING section within GENERAL INFORMATION.

The scroll commands are:

UP * Causes scrolling toward the top of a record, file, record layout, or member list.

DOWN * Causes scrolling toward the end of a record, file, record layout, or member list.

LEFT * Causes scrolling toward the left margin of the data or causes backward scrolling towards the beginning of a file.

RIGHT * Causes scrolling toward the right margin of the data or causes forward scrolling towards the end of a file.

FORWARD Causes record scrolling toward the bottom of the file.

BACK Causes scrolling toward the top of the file.

LOCATE Causes up or down scrolling to the specified line number, label, or member name.

LOCATE Causes up or down scrolling to the record with a label (dataname) or exclude classification. Also causes scrolling to a data-name that contains, in part or full, the data name(occ) string.

LR Causes forward or backward scrolling to a specified record number, label, or any record with a label classification.

RECORD MANIPULATION COMMANDS

Record manipulation commands allow you to update file contents and to find and change occurrences of a string in a file.

The record manipulation commands are:

INSERT Causes "n" records to be inserted after the current record (not supported for BDAM files)

DELETE Causes "n" records to be deleted beginning with current record (not supported for BDAM files)

REPEAT Causes a record to be repeated "n" times (not supported for BDAM files)

FIND Causes one or all occurrences of a string to be found

RFIND * In Edit or Browse, causes the previous FIND command to be repeated

CHANGE Causes one or all occurrences of a string to be changed

RCHANGE * In Edit, causes the previous CHANGE command to be repeated

SAVE * Causes the data to be stored back into the edit dataset

UNDO Causes the last applied change to data to be reversed

When using an XREF to work with "segmented" records the following are valid:

ADD Insert a segment at a point in the record.

NEXT Display NEXT segment of record using correct layout

REMOVE Remove a segment at a point in the record.

TOP Return to start of record and display base

USE Display layouts and allow manual selection and relocation

SESSION CONTROL COMMANDS

Session control commands have to do with terminating the current function or switching from one function to another.

The session control commands are:

CANCEL * Causes editing to be terminated without saving the data

END * Terminates the current operation and returns to the next higher level screen (also used to indicate end of table input)

RETURN * Causes an immediate return to the primary option menu or to a specified option

Recursive commands: File-AID has special commands like F1 (Browse), F2 (Edit), F33 (Copy) that let you suspend your current display and start another function. Upon end from the new function, the suspended display is resumed.

The following will be presented only if selected by number:

1 - Recursive Commands

MISCELLANEOUS COMMANDS

Miscellaneous commands cover a variety of functions. They are listed and briefly described below.

KEY Causes a specified key to be retrieved or the key specification screen to be displayed (applies to BDAM, ISAM, VSAM-KSDS, VSAM-RRDS, and IAM files only)

HEX * Causes data to be displayed in either hexadecimal or character format

CAPS * Causes alphabetic data entered from the terminal to be either translated to upper case or left as-is

RESET * Causes a general resetting of intensified messages and incomplete line commands

CHAR Causes a switch from Formatted or Vertical editing modes to Character editing mode

FORMAT Causes a switch from Vertical or Character editing modes to Formatted editing mode (Aliases: FMT, MAP)

VFMT Causes a switch from Formatted or Character editing modes to Vertical editing mode

SPLIT * Causes split screen mode to be entered or changes the location of the split line

SWAP * Moves the cursor to wherever it was previously positioned on the other logical screen of a split screen pair

TSO * Allows a TSO command or CLIST to be entered from any screen

KEYS * Causes an immediate display of a screen on which current PF key definitions are displayed and modifiable

PANELID * Causes all subsequent screens to be displayed/not displayed with the panel identifier shown in the upper left corner of the screen

HELP * Displays additional information about an error or switches to the on-line tutorial

CURSOR * Moves the cursor to the first input field on line 2

PRINT * Causes a snapshot of the current screen to be recorded in the ISPF list file for subsequent printing

PRINT-HI* Same as PRINT except that high-intensity characters on the screen are printed with overstrikes to simulate the high-intensity display

JCL Causes the generated JCL to be displayed

GENERAL INFORMATION PA KEYS

The two program access (PA) keys have special meaning in ISPF and File-AID. They are not equated to ISPF or File-AID commands, and they may not be reassigned.

Normally, you should not use PA1 while operating in File-AID full screen mode. However, there are two exceptions.

If you press PA1 after the keyboard has been unlocked by File-AID, it is treated like PA2 (RESHOW). It does not cause exit from File-AID. However, if you press PA1 a second time without any intervening interaction, it causes the current processing to be terminated and the ISPF PRIMARY OPTION MENU is redisplayed.

If you press PA1 after the keyboard has been manually unlocked (by pressing the RESET key), it usually causes the current processing to be terminated. The ISPF PRIMARY OPTION MENU is redisplayed.

You can use PA2 to reshow the last full screen image displayed by File-AID.

You may need to reshow the screen if you accidentally press the ERASE INPUT or CLEAR key, or if you have typed data on the screen that you want ignored and have not yet pressed ENTER or a PF key.

DEFAULT ASSIGNMENT

PROGRAM FUNCTION KEYS

When you begin using File-AID, |-------------|---- --------|------------ -----|

commonly used commands are assigned | PF1/13 | PF2/14 | PF3/15 |

to the PF keys as shown on the right. | HELP | SPLIT | END |

| | | |

You can change these default assignments |--------------|-------------|-----------------|

in option 0.0.3 or via the KEYS command. | PF4/16 | PF5/17 | PF6/18 |

| RETURN | RFIND | RCHANGE |

```
|              |             |
      |
|---------------|------------|------------
------|
  |   PF7/19      |   PF8/20    |
PF9/21      |
|  UP        |  DOWN  |  SWAP
     |
|              |              |
        |
|---------------|-------------|-----------
------|
  |  PF10/22     |  PF11/23   |
PF12/24    |
   |    LEFT       |    RIGHT    |
RETRIEVE |
|               |             |
     |
|----------------|------------|------------
-----|
```

GENERAL INFORMATION SCROLLING

In many File-AID functions, the information to be displayed exceeds the screen size. Scrolling allows you to move the screen "window" in as many as three dimensions across the information: up/down, left/right, and forward/back.

You accomplish this using two basic types of scrolling:

PAGE SCROLLING

The page scrolling commands and default PF key assignments are:

UP (PF7/19) - to scroll toward the top of the data

DOWN (PF8/20) - to scroll toward the bottom of the data

LEFT (PF10/22) - to scroll toward the first column of the data

RIGHT (PF11/23) - to scroll toward the last column of the data

These commands function exactly as they do in ISPF/PDF. On all screens involving scrolling, a scroll amount is displayed on line 2 of the screen. This value indicates the number of lines or columns to scroll when you enter one of the four page scrolling commands. You can change the scroll amount by simply typing over the scroll amount field.

Valid scroll amounts are:

- a number from 1 to 9999 - specifies the number of lines/columns to scroll

- PAGE - specifies scrolling by one logical page

- DATA - specifies scrolling by one logical page less one line

- HALF - specifies scrolling by one-half logical page

- MAX - specifies scrolling to the top, bottom, left margin, or right margin, depending on which scrolling command is used.

- CSR - specifies scrolling based on the current position of the cursor. The line or column where the cursor is positioned is moved to the top, bottom, left, or right of the screen depending on which scrolling command is used. If the cursor is not in the body of the data, or if it is already positioned at the top, bottom, left, or right, a full page scroll occurs.

For scrolling purposes a "page" is defined as the amount of information currently visible on the logical screen. In split screen mode, for example, the edit character display might have 12 lines by 80 columns of scrollable data. In this case, a scroll amount of HALF would move the window up or down by 6 lines, or right or left by 40 columns.

The current scroll amount is saved in the user profile. When you type over the scroll amount, the new value remains in effect until you change it again.

You can override the current scroll amount on a given interaction by entering a scroll amount in the command field and using a scroll command or PF key.

For example, you can enter:

```
    COMMAND ===> DOWN 5           and press ENTER or



    COMMAND ===> 5                and press the DOWN PF key
```

Either form results in a temporary, one-time override of the scroll amount.

To reduce keystrokes, you can change the scroll amount field by typing over the first character(s) only:

- To change the scroll amount to PAGE, DATA, HALF, MAX, or CSR, type over the first character with P, D, H, M, or C respectively.

- To change the scroll amount to a number of lines or columns, type over the first character(s) with the desired number.

RECORD SCROLLING

The record scrolling commands are not directly assigned to a PF key:

FORWARD (or FWD)	- to scroll toward the end of the file
BACK	- to scroll toward the beginning of the file

The LEFT (PF10/PF22) and RIGHT (PF11/PF23) commands can be used in place of BACK and FORWARD, respectively.

These commands apply to the screens where one "record" per screen is displayed formatted with a record layout (edit/browse Formatted, edit formatted field selection criteria, define formatted XREF) and are used to move forward or backward "n" records (or criteria sets) in a file.

You enter the scroll amount for record scrolling in the COMMAND input area (unless you have assigned both the command and the scroll amount to a PF key). Valid scroll amounts are:

- a number from 1 to 2,147,483,647 - specifies the number of records to scroll FORWARD or BACK

- MAX or M - specifies scrolling to the last record (FORWARD) or the first record (BACK) in the file

GENERAL INFORMATION SPLIT SCREEN

There may be times when you want to perform another File-AID function or ISPF/PDF function without ending the current function. File-AID provides the ability to 'split' the physical screen image into two logical screens that operate independently of one another, as though you had two terminals.

In split screen mode, only one of the logical screens is considered active at a time. Any interactions, such as pressing ENTER or a PF key, are interpreted as having meaning for the active screen. The current location of the cursor identifies which of the two screens is active. To switch from one screen to the other, simply move the cursor to the desired screen or enter the

SWAP command.

GENERAL INFORMATION TERMINATING File-AID

You can terminate File-AID in any of three ways:

1) by entering the END command on the primary option menu

2) by entering option X on the primary option menu

3) by entering the =X jump command from any screen

When you invoke File-AID from the ISPF primary option menu, eventually terminate ISPF/PDF using the END command, and used either the log or list files, a termination screen is displayed.

GENERAL INFORMATION File-AID DATASETS

General information about the use of datasets by File-AID follows. Each topic is presented in sequence, or may be selected by number:

1 - Conventions for Specifying Datasets

2 - Data File Dataset

3 - Record Layout Dataset and source language support

4 - Layout XREF Dataset

5 - Selection Criteria Dataset

6 - Reformat Definition Dataset

7 - Member List Processing

All datasets processed on-line must reside on a direct access storage device or on a mass storage device. Uncataloged disk data file datasets are allowed in all functions and can be accessed by specifying a VOLSER.

Temporary data files and tape datasets are allowed as input to independent File-AID/Batch jobs for which you develop the JCL. File-AID/Batch JCL can be optionally generated with Copy (3.3), Search/Update (3.6), Print (5.x), Reformat (9), and Compare (10)

CONVENTIONS FOR SPECIFYING DATASETS

When a Dataset Name is required in File-AID, you may enter a fully qualified (quoted), unqualified (no quotes, userid prefix implied), or pattern (catalog search request).

File-AID saves the name of the most recently referenced dataset of each type (Data file, Record Layout, XREF, Selection Criteria) in your session profile and redisplays these names on all appropriate screens.

FULLY QUALIFIED NAMES

You may enter any fully qualified data set name by enclosing the name in apostrophes ('). Trailing quotes are never required.

Example: Dataset name ===> 'SYS1.MACLIB'

UNQUALIFIED NAMES

If you omit the apostrophe(s), your TSO prefix is left-appended to the entered data set name (unless the TSO profile option "NOPREFIX" has been specified).

If you specify a default VSAM INTERMEDIATE NAME with the 0.1 System Parameters function, and/or your installation uses the optional File-AID VSAM high level index, and no dataset can be

Found with the combination of "TSO-PREFIX.DATASET", a new "VSAM prefix" is constructed as follows:

VSAM high level index.VSAM INTERMEDIATE NAME.

The constructed "VSAM prefix" is then left-appended to the entered dataset name (VSAM prefix.dataset) and the catalog is again searched for the dataset. If allocating a VSAM cluster, with the 3.5 VSAM utility, the VSAM prefix is always constructed when an unqualified name is specified.

UNCATALOGED DATASETS

If the dataset is not cataloged, you must specify a Volume serial number.

USING PATTERN CHARACTERS - CATALOG SEARCH

You can use pattern characters in the dataset field to dynamically invoke File-AID's catalog search (3.4) to look for matching dataset names. Valid pattern characters are: ? and % (single character) * (multiple characters in one node), + (0 or more nodes). Pattern characters are used to represent any or all characters which are not part of the high-level node of a dataset name. The high-level node must be fully specified. Following are some examples of using the asterisk (*) to retrieve a list of datasets:

```
COMPARE DATASET ===> FILE.*

        or

COMPARE DATASET ===> 'userid.FILE.*'

        or

COMPARE DATASET ===> FILE.ABC*
```

USING PATTERN CHARACTERS - CATALOG SEARCH (cont.)

After specifying the pattern, press ENTER. The File-AID 3.4 Catalog Utility is invoked, and a list of datasets corresponding to the pattern that you specified is displayed. Type S in the COMMAND column to the left of the name of the dataset you want to select and press ENTER. The selected dataset will be used and will replace the pattern you specified on the panel for the function you are performing.

If you are not licensed for File-AID/SPF, you will receive an error if you use a pattern character in the dataset field.

For further information on pattern characters, explore the tutorial for the 3.4 Catalog utility or see the File-AID Online Reference Manual, CATALOG UTILITY.

PDS DATASET(MEMBER)

A member name enclosed in parentheses may follow the data set name (within the apostrophes, if they are present) for partitioned data sets. For selection purposes, specifying a member name for a partitioned dataset will cause the member to be processed as a sequential dataset.

In most instances a Member Name prompt is provided under the Dataset Name prompt. If a member name is specified in parentheses along with the dataset name, the member information is considered as if it had been entered in the Member Name field.

PDS MEMBER NAME MASKS

When referencing a PDS dataset, you may specify a Member Name of blank, or use a pattern to generate a list of members for selection.

PDS MEMBER NAME MASKS (cont.)

You can use a question mark (?) or percent (%) to represent a single character of the member name or an asterisk (*) to represent multiple characters in a member name. However, you must enter at least the first character of the member name before you can use the asterisk. (Exception: in 3.3 Copy or 3.6 Search Update, a member name of asterisk (*) means to process all members and to bypass all PDS

Processing Options screens and member lists.)

Using blank or a member name pattern will either cause a File-AID member list to be displayed or a PDS Processing Option screen to appear to provide additional member selection options. Once a list of members is presented, you can select a member from this list by using the S line command.

You can also enter the full member name, in parentheses, on the same line as the dataset name. If you do so, you can ignore the Member Name field.

GENERATION DATASETS

Generation data sets may be referenced by using a signed or

unsigned number enclosed in parens.

  Example: ===> 'GDS.TEST(0)'

This example references the most recently allocated data set in

the generation data group. Minus numbers reference previously

allocated data sets and positive numbers reference unallocated

data sets of the group.

CATALOG DATASET

On some panels a catalog dataset name may be entered. In this case whether the catalog dataset name is enclosed in quotes or not, the name is always considered to be a fully qualified name and no prefix is appended to it.

```
Catalog dataset name ===> 'CATALOG.DATASET'

Catalog dataset name ===> CATALOG.DATASET
```

In the above example, both entries are correct and treated equally.

DATA FILE DATASET

The data file dataset contains the data to be browsed, edited, reformatted from/into, copied from/into, searched, updated, printed or compared. It can also be a dataset you are defining or managing with the extended utilities.

File-AID supports user written I/O exit programs for data file datasets. I/O exits can handle compression/decompression, encryption/decryption, or can perform all physical I/O to your data file. See the File-AID Installation Guide for details and samples.

The next two tutorial pages display tables showing the combinations of dataset organizations and record formats allowed for data files. For sequential (PS), partitioned (PO), BDAM (DA), and ISAM (IS) files, the LRECL and BLKSIZE can range from 1 to 32,760 for each RECFM. For VSAM files, the LRECL can range from 1 to 32,767.

The table below shows the combinations of dataset organizations and record formats allowed for data files.

			RECFM			
DSORG	F	FB	V	VB	U	SPANNED
PS (Sequential)	X	X	X	X	X	X
PO (Partitioned)	X	X	X	X	X	X
IS (Indexed Sequential)	X	X	X	X	n/a	n/a
DA (BDAM)	X	X				

The table below shows the combinations of dataset organizations and record formats allowed for VSAM data files.

			RECFM			
DSORG	F	FB	V	VB	U	SPANNED
AM (VSAM-KSDS,ESDS)	n/a	n/a	X	n/a	n/a	X
AM (VSAM-RRDS)	X	n/a	n/a	n/a	n/a	n/a
AM (IAM)	X	n/a	X	n/a	n/a	X
AM (VSAM-LDS)			(allocate, delete, info only)			

1) VSAM-KSDS and ESDS files are treated as variable length files. Also, the concept of blocking, as applied to PS, PO, and IS files, does not apply to VSAM.

2) File-AID does not support ISAM files with OPTCD=L in the DCB information when the key starts in the first position (RKP=0 for fixed length files, RKP=4 for variable length files)

3) File-AID does not support fixed length unblocked ISAM files with a relative key position of 0 (RKP=0), or variable length unblocked ISAM files with a relative key position of 4 (RKP=4).

4) File-AID supports reading a VSAM-KSDS in an alternate key sequence via a VSAM PATH. A PATH name may be specified for the input data file name in the browse, edit, copy, print, and compare options.

5) File-AID treats RECFM=FB BDAM files as RECFM=F (i.e., records are not de-blocked.) Keyed BDAM files are processed by File-AID as if the key were concatenated in front of each data block.

6) IAM files exist as fixed or variable length files. Due to a constraint in the information available to File-AID from the IAM access method, all files are currently handled as variable length files. If the length is changed on an IAM file which is truly fixed, File-AID will display an error message. IAM files are processed as VSAM-KSDS files in all functions.

OS PASSWORD PROTECTED FILES

File-AID allows you to access data file datasets with OS password protection. Any dataset may be protected to permit read-only or read/write access. More than one password may be assigned to the same dataset. A dataset that is read/write protected, for example, might allow several authorized users to read it, but only one user to write.

If you attempt to access a dataset protected by an OS password, File-AID will present the OS-Password-Protected screen containing a field where you can specify a dataset password as follows:

```
Dataset password   ===>
```

A nondisplay input field is used so that the password does not appear on the screen while you are typing it.

Note that the File-AID 3.5 VSAM utility supports specification and modification of passwords for VSAM clusters.

RECORD LAYOUT DATASET

This dataset contains the record layouts which are used in many File-AID functions. The record layout dataset can be one of four types:

TYPE	DSORG	RECFM	LRECL
Partitioned Dataset	PO	F, FB, V, VB	80
PANVALET file	*	*	*
LIBRARIAN file	*	*	*

Standard sequential dataset	PS	F, FB, V, VB	80

Must conform to the requirements for a valid CA-PANVALET/LIBRARIAN file. CA-PANVALET and CA-LIBRARIAN are products of Computer Associates International, Inc.

A record layout may be a separate COPYLIB member or may be hard-coded within the source statements of a program. If the layout is in a program or if a COPYLIB member contains multiple record layouts, you can isolate the specific layout, or portion of a layout, that File-AID should use by specifying a starting data-name.

If you do not specify a starting data-name, File-AID uses every layout found in the program, or COPYLIB member, to format the data, treating the second through last layouts as additional layouts that may be manually selected and overlaid at any offset via the USE command.

Note: File-AID Release 6 "MAP" libraries containing compiled record layouts are acceptable as a record layout library.

LANGUAGE SUPPORT

File-AID supports source record layouts coded in COBOL or PL/I.

Layouts may be used when browsing, editing, printing, reformatting and comparing data files.

Layouts can also be used to define selection criteria and to specify tests for multiple record type layout usage.

File-AID automatically recognizes the source language whenever a layout is referenced.

File-AID supports Format 1 of the COBOL data definition statement and ignores Formats 2 and 3. When a 66 or 88 level item is encountered File-AID skips to the end of the sentence and performs no syntax checking.

```
      FORMAT 1                    FORMAT 2


   level number data-name           66 data-name-1 RENAMES clause.


     REDEFINES clause

     BLANK WHEN ZERO

     JUSTIFIED clause               FORMAT 3
```

OCCURS clause

PICTURE clause   88 condition-name VALUE clause.

SIGN clause

SYNCHRONIZED clause

USAGE clause

VALUE clause.

The following rules apply to Format 1:

1) The first data description entry in the COBOL layout need not have level number 01.

2) If multiple data description entries have level number 01, the second through "n" 01 level entries are treated as though they redefine the first 01 level entry. Each 01 level entry need not be successively longer than the previous 01 level entry.

3) The clauses may be written in any order.

4) The PICTURE clause must be specified for every elementary item with the exception of INDEX, COMP-1, and COMP-2 data items.

5) Each data description entry must be terminated by a period.

6) Semicolons or commas may be used as separators between clauses.

7) The EJECT, SKIP1, SKIP2, and SKIP3 reserved words may be imbedded between data definition statements within a COBOL layout. File-AID skips over these words.

8) File-AID also skips over the TITLE statement, which is valid in VS COBOL II.

9) Sort file description entry statements (SD) and file description entry statements (FD) may be imbedded within the COBOL layout. File-AID skips these statements and performs no syntax checking.

10) File-AID also skips debugging lines ("D" in column 7) and sentences which do not begin with a valid level number.

11) File-AID ignores the REDEFINES clause if it is on the first data description entry in the COBOL layout.

The level number and data-name are required in the COBOL data description entry. The format is:

Level number data-name

The level number may be any number from 1 through 49. You may not use level number 77.

Data-names must be 30 characters or less but are not otherwise validated. For example, you are not restricted to using characters A through Z, 0 through 9, and hyphen to form data-names, and you need not make data-names unique. The keyword FILLER is treated the same as other data-names.

The format of the REDEFINES clause is:

    level number data-name-1 REDEFINES data-name-2

The level numbers of data-name-1 and data-name-2 must be identical and can be any level number from 1 to 49. Between the data descriptions of data-name-2 and data-name-1, there may be no entries having numerically lower level numbers than the level number of data-name-2 and data-name-1.

The data description entry for data-name-2 may not contain an OCCURS clause. However, data-name-2 may be subordinate to an item containing an OCCURS clause with or without the DEPENDING ON option. Items subordinate to data-name-2 may contain an OCCURS clause without the DEPENDING ON option.

Data-name-1 and any items subordinate to data-name-1 may contain an OCCURS clause with or without the DEPENDING ON option. If data-name-1 contains an OCCURS clause, its length is computed to be the length of one occurrence multiplied by the number of occurrences.

The size of the redefining area must be less than or equal to the size of the redefined area. Multiple redefinitions of the same redefined area may reference the data-name of the redefined area or the data-name of any of the preceding redefining areas.

REDEFINES clauses may also be specified for items subordinate to items containing REDEFINES clauses.

File-AID ignores the REDEFINES clause if it is on the first data description entry in the COBOL layout.

The format of the BLANK WHEN ZERO clause is:

    BLANK (WHEN) ZERO

The word "WHEN" is optional. This clause is checked for valid syntax but it does not affect processing in any way.

The format of the JUSTIFIED clause is:

    JUSTIFIED (RIGHT)

```
     JUST
```

The JUSTIFIED clause is used to override normal positioning of data within a receiving alphanumeric data item. This clause affects processing in the reformatting function (option 9) only.

Normally data is left justified in the receiving field and either truncation or blank padding occurs on the right. When the JUSTIFIED clause is specified, data is right justified in the receiving field and either truncation or blank padding occurs on the left.

The OCCURS clause has two basic formats:

FORMAT 1

```
OCCURS integer-2 TIMES (DEPENDING ON data-name-1)

  (ASCENDING  KEY IS data-name-2 (data-name-3)...)...

   DESCENDING

  (INDEXED BY index-name-1 (index-name-2)...)
```

FORMAT 2

```
OCCURS integer-1 TO integer-2 TIMES (DEPENDING ON data-name-1)

  (ASCENDING  KEY IS data-name-2 (data-name-3)...)...

   DESCENDING

  (INDEXED BY index-name-1 (index-name-2)...)
```

The words "TIMES", "KEY IS", and "BY" are optional in both formats.

In FORMAT 1, integer-2 represents the exact number of occurrences if the DEPENDING ON option is not present, or the maximum number of occurrences if the DEPENDING ON option is present. Integer-2 must be greater than 0 and less than 32,768.

In FORMAT 2, integer-1 represents the minimum number of occurrences and integer-2 represents the maximum number of occurrences. Integer-1 must be greater than or equal to 0, and integer-2 must be greater than integer-1 and less than 32,768.

Three nested levels of OCCURS clauses are allowed. The OCCURS clause may not be specified on 01 level items.

The ASCENDING KEY, DESCENDING KEY, and INDEXED BY options are checked for valid syntax, but do not affect processing in any way.

Data-name-1, the object of the DEPENDING ON option:

- must be defined as an integer
- must not contain or be subordinate to an OCCURS clause

There can be multiple OCCURS DEPENDING ON clauses per COBOL layout.

The format of the PICTURE clause is:

```
PICTURE (IS) character-string

PIC
```

File-AID interprets all options of the PICTURE clause. The alphabetic, alphanumeric, alphanumeric edited, and numeric edited categories of data are all treated as alphanumeric data (i.e. treated as if the character string contained all X's). Picture characters

```
A  X  B  0  Z  ,  .  *     -  CR  DB  $  E
```

are all treated as the picture character X (XX for CR and DB). For alphanumeric edited and numeric edited data items, File-AID does not validate the character string according to the "precedence of symbols" as COBOL does.

External floating point items are fully syntax checked and treated as picture X items. Also, the scaling position character, P, is supported

The format of the SIGN clause is:

```
SIGN (IS) LEADING  (SEPARATE CHARACTER)

TRAILING
```

The word "CHARACTER" is optional in the SEPARATE option. File-AID treats the SIGN clause exactly as COBOL treats it.

The format of the SYNCHRONIZED clause is:

```
SYNCHRONIZED (LEFT)

SYNC        (RIGHT)
```

The SYNCHRONIZED clause may be specified at the 01 or elementary item level. When specified at the 01 level, every elementary item within the 01 level is synchronized.

File-AID treats all other aspects of the SYNCHRONIZED clause exactly as COBOL treats them, including computation of slack bytes for groups, defined with the OCCURS clause, containing synchronized items.

The format of the USAGE clause is:

```
          DISPLAY

          DISPLAY-1

          COMPUTATIONAL

          COMP

          COMPUTATIONAL-1

          USAGE (IS)    COMP-1

          COMPUTATIONAL-2

          COMP-2

          COMPUTATIONAL-3

          COMP-3

          COMPUTATIONAL-4

          COMP-4

          INDEX

          POINTER (VS COBOL II only)

          PACKED-DECIMAL (VS COBOL II only)

          BINARY (VS COBOL II only)
```

The USAGE clause can be specified at any level of data description. However, if the USAGE clause is written at a group level, it applies to each elementary item in the group. The usage of an elementary item may not contradict the usage of a group to which an elementary item belongs.

If the USAGE clause is not specified for an elementary item, or for any group to which the item belongs, usage DISPLAY is assumed.

COMPUTATIONAL-4 and COMP-4 items are treated as COMPUTATIONAL items.

The format of the VALUE clause is:

VALUE (IS) literal

The VALUE clause is checked for valid syntax but does not affect processing in any way.

COBOL LAYOUT ERRORS

When File-AID encounters an error during its on-line processing of a COBOL layout, an error messages is written to your ISPF log file. These messages contain an image of the last line File-AID read from the COBOL layout, plus the internal layout representation File-AID built before

the error occurred. By using these messages in conjunction with the error message File-AID displays at your terminal, you will be able to resolve most COBOL layout errors.

LAYOUT MEMBER SELECTION

Whenever you leave the layout member name blank or use a pattern, File-AID presents a member list. A partitioned processing options (PPO) screen usually appears first to let you filter the member list to improve performance. A pattern consisting only of an asterisk (*) tells File-AID to list all members and to bypass the PPO screen.

NOTE: Using a single asterisk (*) pattern to list all members of a large layout library (such as a Panvalet or Librarian library) may be time consuming.

Member list presentation occurs automatically in the XREF function when you use line commands, or enter a pattern in the member name field (with or without using the S or SU line command).

SOURCE STATEMENT SELECTION

SOURCE OPTIONS

When File-AID cannot compile your record layout with the default source options, the Source Statement Selection panel is displayed. This panel contains a Source Statement selection area and a compiler option section. Record layout extraction can be done by "Character string" or

"Statement number".

Character string selection is performed by entering the full or partial dataname of the first group item wanted. The ending string is only needed for PL/I or when a partial group is wanted.
Selection by number is performed by

```
 Please enter the beginning and term
Source statement line numbers or bl

Record layout selection by Character St
 Beginning string     ===>
Ending string        ===>
Record layout selection by Source State
```

entering the beginning and ending	Beginning number ===>
statement numbers. The numbers can	Ending number ===>
be standard (cols 73-80), relative,	Number type ===>
or COBOL (cols 1-6). Relative is	

The Source Options panel is also used to enter compiler options. The compiler options are Language, Starting level number, Literal delineator (COBOL only) and Use 48 character set (PL/I only).

Language is specified by entering	Compiler options:
COBOL or PL/I in the language field.	Language ===> COBOL
Starting level number indicates the	Starting level numb ===> 01
level number to be used to indicate	Literal delineator ===> QUOTE
the start of a new copybook to be compiled.	Use 48 character set ===> YES
Literal delineator indicates to the	--

COBOL compiler that either the quote (") or the apostrophe (') is the character used to delineate literals.

-Use 48 character set indicates to the PL/I compiler that the source is written in the 48-character set.

If File-AID cannot determine the language, or if compilation fails, it will display this panel to allow you to change source or compile options.

XREF DATASET

This dataset contains XREF members created using option 7 of

File-AID. XREF members contain information relating record data to a

record layout. The XREF dataset must have the following attributes:

TYPE	DSORG	RECFM	LRECL
Partitioned Dataset	PO	VB	300

LAYOUT XREF DATASET

You create XREF members to communicate information about the record layout dataset to File-AID in the following situations:

1) the data file you want to work with contains records with differing formats, requiring different record layouts to define the various formats

2) the record layout that defines the data file is hard-coded in a program

3) the record layout that defines the data file is stored in a COPYLIB member that contains multiple record layouts

4) each record in the data file requires two or more record layouts, presented in sequence, to define it

For the first situation, the XREF contains two pieces of information:

1) the name or the length and starting position of the field(s) within each data record that File-AID should use to identify the data record's type (referred to as the "record type field(s)")

2) a list of all the possible record type values and the corresponding layout library member which defines the data records with those values

For the second and third situations, the XREF contains the data-name that starts the data structure you want (referred to as the "starting data-name"). File-AID reads the layout from the source library member (containing either multiple layouts or a program) starting with the data-name you specify. File-AID stops reading when it finds a new data structure (equal or lower level number) or when the end of the record layout, library member, or program is reached.

For the fourth situation, the XREF contains the names of the layout library members for each segment of a data record. It also contains the rules for identifying each segment of the record by locating the "segment type field" and field values to use for selecting a segment layout (via NEXT command of Formatted Browse/Edit).

In any of the File-AID primary options which use layouts to define the data, the record layout dataset and XREF dataset information are specified as shown in the example below.

```
Specify Record layout and XREF Information:

Record layout usage        ===>   X      (S = Single; X = XREF; N = None)
Record layout dataset      ===>   'PROD.COPYLIB'
Member name                ===>   (Blank or pattern for member list)
XREF dataset name          ===>   FASAMP.XREF
Member name                ===>   ORDRFILE (Blank or pattern for member list)
```

RECORD LAYOUT USAGE indicates whether you will use a single record layout (usage "S"), a layout XREF (usage "X"), or no source layouts (usage "N") to define the file.

The table below shows the valid dataset and member entries for the record layout and layout XREF datasets, for each layout usage.

LAYOUT USAGE	RECORD LAYOUT DATASET	XREF DATASET
S	sequential file, or member of partitioned/PAN/LIB file	ignored

X	sequential file, or partitioned/PAN/LIB file; (required) member must be blank	member of partitioned file
N	ignored	ignored

SELECTION CRITERIA DATASET

This dataset contains existing "selection criteria information" created using option 6 of File-AID. You can use saved selection criteria in the browse, edit, copy, search/update, print, and compare functions.

The selection criteria dataset must have the following attributes:

TYPE	DSORG	RECFM	LRECL
Partitioned Dataset	PO	V, VB	300

REFORMAT DEFINITION DATASET

This dataset contains saved Reformat Definitions created using option 9 of File-AID. You can use a saved Reformat Definition online or in batch (REFORMAT function).

The Reformat Definition dataset must have the following attributes:

TYPE	DSORG	RECFM	LRECL
Partitioned Dataset	PO	V, VB	1570

MEMBER LIST PROCESSING

When partitioned datasets are specified on the dataset specification screens, you may display a member list by:

- Leaving the member name field blank

  File-AID displays a list of the members in the partitioned dataset

- Entering a pattern as the member name

A pattern is a partial member name containing one or more of the single character "wild cards" * (asterisk) ? (question) or % (pct). If any pattern character is present, the name becomes a mask

matching any member name starting with the pattern entered (as if the pattern contained all "wild cards" at the end). Therefore, one trailing pattern character is used only if it is the only "wild card" in the mask, otherwise it is assumed. (Ex. PR?G is the same as PR?G???? and PR?G*. However PRO and PRO* are different because PRO is a member name not a pattern like PRO* is.)

For example, the patterns: ?RO*, ?RO, P*, PR?G%, P?O?G, PROGRAM* all match and produce the following list:

PROGRAM1

PROGRAM2

PROGRAM3

Use the S (Select) line command to select the member you want. You may also scroll through the member list using the UP and DOWN commands. In addition, you can enter a LOCATE primary command and scroll directly to (or near) a specified member name.

MULTIPLE PDS DATASET MEMBER LIST CONVENTIONS:

On certain dataset specification screens, you can specify multiple partitioned datasets. Therefore, it is possible for you to receive multiple member list screens (one for each partitioned dataset specified without a member or with a pattern) before proceeding to the next screen in the function. The member lists are displayed in the sequence that the datasets appear on the dataset specification screen.

PDS PROCESSING OPTIONS FOR PROCESSING PARTITIONED DATASETS:

If the main dataset in Browse (1), Edit (2), Copy (3.3), Search/Update (3.6), and Print data (5.1), is a PDS and an explicit member is not specified, File-AID provides a PDS Processing Options (PPO) screen. The PPO screen lets you specify additional member qualification filters such as: member name range, last modified userid range, create date range, and modified date range.

If you wish to bypass the PPO screen and process all members, use a member name of * on the main entry screen (Ex. Edit DSN=MY.PDS(*)).

Note that when multiple PDSs are specified on the entry screen the list of members matching your PPO options may not appear until AFTER all other PDS member lists have been presented and a member selected.

ABNORMAL TERMINATION

File-AID can abnormally terminate in one of two situations:

1) when File-AID detects an unrecoverable error

2) when File-AID itself terminates abnormally

In both situations, File-AID intercepts the abend and displays an error screen. In the first case, a message is displayed describing the unrecoverable error encountered. In the second case, the system and user abend codes are displayed.

The first keyboard interaction on the error display screen causes the ISPF abnormal termination screen to be displayed.

File-AID attempts to close all open files prior to returning control to ISPF.

DATASET SECURITY

File-AID is completely compatible with any data security software your installation may have (Ex. RACF, CA-ACF2/TOP Secret, etc). File-AID processing does not circumvent your security software in any way.

File-AID provides your installation with the capability of creating its own security exit routine that File-AID will call during its processing. This enables your installation to limit access to certain datasets. Following are some examples of the optional security exit routine uses:

- To allow only authorized users to browse and/or edit selected data files

If you require information on your installation's security exit routine, contact the person responsible for File-AID at your installation.

In addition to the security exit, an audit exit is provided that can be customized for various uses including:

- To force the creation of an audit trail when option 2 of File-AID is used to edit selected data files, and/or to force the printing of an audit trail report upon completion of the data file updates

- To log all or selected dataset updates to the SMF log.

AUDIT TRAIL FEATURE

When you use option 2 of File-AID to edit a data file, you may specify that File-AID capture all of your update activity in an audit trail dataset. Every insert, repeat, delete, copy, typeover, and CHANGE command you perform causes a before and/or after image of the record(s) affected to be written to the audit trail dataset. When you complete your edit session, a screen is displayed giving you the option to print, delete, or keep the audit trail. If you specify that the audit trail be printed, a batch job is submitted which produces a formatted audit trail report.

File-AID also provides your installation with the ability to force the creation of an audit trail during any user's edit session through an audit exit routine. This is especially useful, for example, if your installation wants to ensure that an audit trail is created whenever File-AID is used to edit certain sensitive data files. When the edit session is complete, the audit exit routine can also be used to force the automatic submittal of a batch job to produce the audit trail report, thereby ensuring that a record of the edit session is printed.

In addition to the audit trail facility, File-AID provides the option to log all dataset updates to the system SMF datasets.

If you require information on how your installation may be using any of these audit trail facilities, contact the person responsible for File-AID at your installation.

COMPRESSION/ENCRYPTION EXIT

File-AID provides your installation with the capability of creating its own compression/encryption exit routine that will act as an interface between File-AID and any compression/encryption software you may have (e.g., SHRINK or DATA PACKER). This allows you to work with data files that are stored on disk or tape in a compressed or encrypted format in all File-AID options. File-AID invokes the compression/encryption exit for data files only, not for record layout datasets, record/layout cross references, or selection criteria.

If you require information on your installation's compression/ encryption exit, contact the person responsible for File-AID at your installation.

DOUBLE-BYTE CHARACTER SUPPORT

File-AID will automatically detect and present DBCS data characters if your terminal type is set to 3277KN or 3278KN from option 0.0.1 of File-AID, or option 0.1 of ISPF (ISPF terminal characteristics).

There are two ways File-AID presents data.

The two ways are presented in sequence, or may be selected by number:

1 - Formatted data presentation (Formatted and Vertical modes)

2 - Full-screen data presentation (Character mode)

File-AID assumes that all character fields contain a mixture of EBCDIC data and Double-Byte Character Set (DBCS) data. The DBCS characters are enclosed with shift-out (X'0E') and shift-in (X'0F') characters.

For PIC G(nn) fields in COBOL or GRAPHIC(nn) fields in PL/I File-AID assumes that the field contains only Double-Byte Character Set (DBCS) data. DBCS data does not include shift-out (X'0E') and shift-in (X'0F') characters and should be an even number of bytes in length.

Note: X'0E' and X'0F' are for IBM sites. Other platforms such as FSP and Hitachi may use other representations for shift-in and shift-out.

FULL-SCREEN DATA PRESENTATION

When data is presented without COBOL or PL/I record layouts (three-line hexadecimal or character format), each character line displayed is considered a character field.

File-AID assumes that all character fields contain a mixture of EBCDIC data and Double-Byte Character Set (DBCS) data. The DBCS characters are enclosed with shift-out (X'0E') and shift-in (X'0F') characters.

Use caution when entering DBCS characters to ensure that the shift-out and shift-in characters are accounted for and will not overlay other data values unexpectectly.

The command HEX ON is useful to view the location of the shift bytes.

Double-byte character support

File-AID will automatically detect and present DBCS data characters if your terminal type is set to 3277KN or 3278KN from option 0.0.1 of File-AID, or option 0.1 of ISPF (ISPF terminal characteristics).

There are two ways File-AID presents data.

The two ways are presented in sequence, or may be selected by number:

1 - Formatted data presentation (Formatted and Vertical modes)

2 - Full-screen data presentation (Character mode)

FORMATTED DATA PRESENTATION

File-AID assumes that all character fields contain a mixture of EBCDIC data and Double-Byte Character Set (DBCS) data. The DBCS characters are enclosed with shift-out (X'0E') and shift-in (X'0F') characters.

For PIC G(nn) fields in COBOL or GRAPHIC(nn) fields in PL/I File-AID assumes that the field contains only Double-Byte Character Set (DBCS) data. DBCS data does not include shift-out (X'0E') and shift-in (X'0F') characters and should be an even number of bytes in length.

Note: X'0E' and X'0F' are for IBM sites. Other platforms such as FSP and Hitachi may use other representations for shift-in and shift-out.

FULL-SCREEN DATA PRESENTATION

When data is presented without COBOL or PL/I record layouts (three-line hexadecimal or character format), each character line displayed is considered a character field.

File-AID assumes that all character fields contain a mixture of EBCDIC data and Double-Byte Character Set (DBCS) data. The DBCS characters are enclosed with shift-out (X'0E') and shift-in (X'0F') characters. Use caution when entering DBCS characters to ensure that the shift-out and shift-in characters are accounted for and will not overlay other data values unexpectectly.

The command HEX ON is useful to view the location of the shift bytes

Endevor:-

ENDEVOR SOFTWARE

use Endevor for

Source control

To ensure that we always work with the latest version of the source

Endevor Search

To search for a string in a large list of sources

SCL generation

This is required at the time of migration

Endevor environments

FIX	Alternate/Fix environment
TEST	Test environment
RELEASE	- Release environment
DEMOTEST	- Demo test environment
DEMOPROD	- Demo production envmt.

Endevor Options

0	DEFAULTS
1	DISPLAY
2	FOREGROUND
3	BATCH
4	PACKAGE
5	BATCH PACKAGE
U	USER MENU
T	TUTORIAL
C	CHANGES
X	EXIT

Source control in Endevor

A source may reside in any of these stages:

Environment FIX, System EAS0,	FIXTEST
	FIXSHIP
Environment TEST, System EAS0,	ACCEPTANCE
	INTEGRATION
Environment RELEASE, System EAS,	SHIP

PRODUCTION

Display Options in Endevor

The ‘Display element list’ screen can be accessed in all three environments of Endevor. In RELEASE, it can be accessed by entering option 1 ;1 (i.e, display; element)

Options B (browse), C (change), H (history), M (master) and S (summary) can be used to get information on an element.

Option ‘B’	will give a source listing of the element as it currently exists in the region specified
Option ‘C’	can be used to get details of changes made to the current source with respect to the previous version
Option ‘H’	Can be used to get details of all changes ever made to the source. For each line of the source code, the appropriate version number is indicated at the extreme left
Option ‘M’	Can be used to get information about the current status of the source (i.e. whether it has been retrieved, if so, by whom and on what CCID etc.)

Getting Summary Information

This will give a list of all versions of sources that have existed in endevor to date.

1. Enter the ‘Display element list’ screen (= en ; e ; 3 ; 1; 1)
2.Enter the element name and type, with environment as RELEASE and stage as PRODUCTION.
3. Type in option ‘S’ - summary. Press enter

any version may be selected from this list giving the options B, C or H

Display options with ‘Summary’

Option ‘B’	can be used to browse the selected version of the source
Option ‘C’	can be used to get details of changes made to the current (selected) version with respect to the previous version
Option ‘H’	can be used to get details of all changes ever made up to the selected version. For each line of the source code, the version number is indicated at the extreme left

Current Source Information

Enter the ‘Display element list’ screen (= en ; e ; 2 ; 1; 1)
Enter the element details, with environment as TEST and stage as Integration (this is the lowest level in endevor)
Set ‘Build Using Map’ option to ‘Y’. Press enter

This will give a listing of the sources **currently** existing in endevour (e.g versions of the same source may be existing in Integration, Ship and Production).

Retrieval

1. For purposes of modification, we will normally retrieve elements from Production.
2. Elements must always be retrieved without a signout.

Endevor Retrieval Screen

```
OPTION  ===> R
                        ELEMENT DISPLAY OPTIONS:

   blank - Element list        S - Summary   B - Browse    H - History

   R - Retrieve element        M - Master    C - Changes

 FROM ENDEVOR:                     ACTION OPTIONS:
  ENVIRONMENT ===> RELEASE          CCID            ===>
  SYSTEM      ===> EAS              EXPAND INCLUDES  ===> N (Y/N)
  SUBSYSTEM   ===> EAS              SIGNOUT ELEMENT  ===> N (Y/N)
  ELEMENT     ===> IOINTEG1         OVERRIDE SIGNOUT ===> N (Y/N)
  TYPE        ===> COBOL            REPLACE MEMBER   ===> N (Y/N)
  STAGE       ===> P       S - SHIP        P - PROD
  COMMENT     ===>

 TO ISPF LIBRARY:                  LIST OPTIONS:
   PROJECT ===> EASV11              DISPLAY LIST      ===> Y (Y/N)
   LIBRARY ===> QFIX                WHERE CCID EQ     ===>
   TYPE    ===> COBOL               WHERE PROC GRP EQ ===>
   MEMBER  ===>                     BUILD USING MAP   ===> N (Y/N)
                                    FIRST FOUND       ===> Y (Y/N)
 TO OTHER PARTITIONED OR SEQUENTIAL DATA SET:
   DATA SET NAME ===>
```

Endevor Search

Using Endevor Search

The search option can be used to search a large list of sources for the occurence of a given string. Follow the steps as given below:

1. Select the desired endevor environment (Usually RELEASE).
2. Select the BATCH option.
3. Select the ‘BUILD SCL’ option giving a temporary member name.
4. Select option 11 (To list elements).
5. Enter the the partial names of the of the elements you wish to scan, and the corresponding type.
6. Press enter- SCL will be generated; then press F3
7. Again press F3 to return to the Batch Options Menu
8. Enter option 3 to Submit the SCL

9. Scan the held output under DDNAME C1MSG1
10. Search for the return code of ‘0000’. Any element which has ‘0000’ against it has the string you are looking for.

(e.g. IO* & COBOL);
Set Display List to ‘N’;
Select Option ‘L’;
Type in the string to be searched for.

```
                        Batch Options menu
OPTION  ===> 1

   1 BUILD SCL - Build batch SCL actions
   2 EDIT      - Edit request data set
   3 SUBMIT    - Submit job for batch processing
   4 VALIDATE  - Check request data set for syntax errors
   5 BUILD JCL - Enter additional JCL to be included with the job

 REQUEST DATA SET:
   PROJECT  ===> AGP2269        APPEND      ===> N (Y/N)
   GROUP    ===> SOURCES        INCLUDE JCL ===> N (Y/N)
   TYPE     ===> COBOL
   MEMBER   ===> SEARCH

 OTHER PARTITIONED OR SEQUENTIAL DATA SET:
   DSNAME ===>

 JOB STATEMENT INFORMATION:
   ===>
   ===>
   ===>
   ===>
```

```
                        List Element Action
OPTION  ===> L
                  ELEMENT DISPLAY OPTIONS:

   blank - Element list        S - Summary   B - Browse    H - History

   L - LIST element action      M - Master    C - Changes

 FROM ENDEVOR:                   LIST OPTIONS:
  ENVIRONMENT ===> RELEASE           DISPLAY LIST      ===> N (Y/N)
  SYSTEM      ===> EAS               WHERE CCID EQ     ===>
  SUBSYSTEM   ===> EAS               WHERE PROC GRP EQ ===>
  ELEMENT     ===> IO*               BUILD USING MAP   ===> N (Y/N)
  TYPE        ===> COBOL
  STAGE      ===> P       S - SHIP        P - PROD

 TEXT STRING:
   ===> 'CONSOLLT'
 SCAN COLUMNS:
  START ===>    END ===>
  SHOW TEXT ===> Y (Y/N)

 ACTION TO BE GENERATED WHEN LIST IS CREATED ===>
 WRITE LIST TO OUTPUT DATA SET ===> N (Y/N)
 WHERE COMPONENT EQ ===>
```

SCL Generation

SCLs are generally used to migrate sources from the Development region to TEST. The steps involved in the preparation of an SCL are listed as the following

How to generate an SCL

1. Select option 2, i.e., ‘TEST’.
2. Select option 3, i.e., ‘BATCH’.
3. To build an SCL for the first time, enter options as given below and press ENTER. whenever you want to build a fresh SCL, set the append option to ‘N’. If you want to append elements to an existing SCL, then set this option to ‘Y”.
4. On pressing ENTER on the previous screen, the SCL GENERATION screen is brought up. Enter option ‘2’ (ADD/UPDATE) regardless of whether you are creating a new SCL or updating an existing one.
5. In the ‘Add/Update Elements’ screen, type in the fields as shown in the next slide. Pressing the enter key now will add a member to the SCL
6. Add the remaining members by repeating step 5. Make sure that the member name, type and processor group are correctly entered.
7. Press PF03 to save the SCL. Inform the on-site team giving them the name of the SCL that has been built [eg. ‘EASV11.QFIX.SCL(SCLTEST)’].
   SCL Submission will be done on-site.

Batch Options menu

```
OPTION  ===> 1

   1 BUILD SCL - Build batch SCL actions
   2 EDIT      - Edit request data set
   3 SUBMIT    - Submit job for batch processing
   4 VALIDATE  - Check request data set for syntax errors
   5 BUILD JCL - Enter additional JCL to be included with the job

 REQUEST DATA SET:
   PROJECT  ===> AGP390        APPEND     ===> N (Y/N)
   GROUP    ===> QFIX           INCLUDE JCL ===> N (Y/N)
   TYPE     ===> SCL
   MEMBER   ===> SCLTEST

 OTHER PARTITIONED OR SEQUENTIAL DATA SET:
   DSNAME ===>

 JOB STATEMENT INFORMATION:
   ===>
   ===>
   ===>
   ===>
```

```
                              Add/Update Elements
OPTION  ===> A

  blank - Member list    A - Add an element    U - Update an element

 TO ENDEVOR:                    ACTION OPTIONS:

  ENVIRONMENT ===> TEST          CCID             ===> EASY2KUU4

  SYSTEM      ===> EAS0          GENERATE ELEMENT     ===> Y (Y/N)
  SUBSYSTEM   ===> EAS            DELETE INPUT SOURCE  ===> N (Y/N)
  ELEMENT     ===> DC1005         NEW VERSION          ===>
  TYPE        ===> COBOL         OVERRIDE SIGNOUT     ===> Y (Y/N)
  STAGE:          I           PROCESSOR GROUP      ===> CIINCL
                           UPDATE IF PRESENT    ===> N (Y/N)
  COMMENT     ===>

 FROM ISPF LIBRARY:              LIST OPTIONS:
  PROJECT ===> EASV11             DISPLAY LIST      ===> Y (Y/N)
  LIBRARY ===> LR11
  TYPE    ===> COBOL
  MEMBER  ===> DC1005     THRU MEMBER ===>

 FROM OTHER PARTITIONED OR SEQUENTIAL DATA SET:
  DATA SET NAME ===>
```

Add/Update elements

1. Use option ‘A’ when you are adding new elements to the SCL. The same option will hold even when we are appending members to an existing SCL.
2. Ensure that the member names match in the ‘TO’ and ‘FROM’ libraries. This is very important !
3. The Processor group field will depend on the type of program. This field is mandatory when new sources are being added to the system. Use the following key:

Sorce Type	**Processor Group**
Batch COBOL Program	CIINBL
CICS COBOL Program	CIINCL
Copy Book	CPYNNN
Control card	SRCNNS
Bind card	BNDSUBX
Batch to Module (TDM)	CIITBL
CICS to Module (TDM)	CIITCL
Batch to Module (without TDM)	CIIDBL
CICS to Module (without TDM)	CIIDCL

NOTES

After building the SCL, it is advisable to manually browse it. For this, use the EDIT option shown in the first screen.
Check for :

1. Mismatched member name
2. Repeated entries for same element
3. Wrong Processor group

Banking and Finance Terms

Asset Management

Asset management is the service of making investment decisions for investors according to predefined guidelines and objectives. Investment management, portfolio management, fund management, and money management are synonymous terms.

Asset Allocation

The practice of deciding how investments are divided among asset categories.

Cash Equivalents

Investments that are fixed income securities with relatively short maturities (for example, less than one year). Examples of this are a money market fund and a treasury bill.

Counterparty Risk

The risk that the other entity in a financial agreement will be unable to fulfill the obligations of that transaction.

Custodian

Bank of other financial institution that keeps custody of stock certificates and other assets of a mutual fund, individual, or corporate client.

Defined Benefit Plan

A type of employee benefit program where the benefit is defined normally in terms of a promised monthly retirement income. An individual investment account for each participant is not maintained. The plan is normally funded through contributions made by the employer. Defined Contribution Plan A type of employee benefit program where the contribution is defined and an individual investment account is maintained for each participant reflecting accumulated contributions and earnings. Contributions can come from the employer, the employee or both. There is no promise regarding the level of retirement income.

Equity

A type of security representing ownership possessed by shareholders in a corporation. An equity investment differs from a debt investment in that the issuer makes no promise to pay the investor current income or any guaranteed amount at a future date. Additionally, although they are owners, shareholders do not usually participate actively in the management of the corporation, other than to elect directors, select auditors, and to vote on matters that may be required by law.

Fixed Income

A type of security that pays a fixed rate of return. Examples of this include government, corporate, or municipal bonds, which pay a fixed rate of interest until the bonds matures, or preferred stocks, which pay a fixed dividend.

Investment Trusts

A type of investment company that pools money from it shareholders. Investment trust is also known as a closed-end investment company, meaning only a fixed number of outstanding shares is offered

Mutual Fund

A type of investment company that pools money from its shareholders. A mutual fund is also known as an open-end investment company, meaning that there is a continuous offering of new shares and redemption of outstanding shares

Value Chain

A value chain breaks down an industry to its value activities to understand the behavior of costs, and the existing and potential sources of differentiation. It is used to compare and contrast competitors in an industry.

Commercial Banking

The commercial bank meets the financing and cash management needs of business customers. Customers range from small businesses to multi-national corporations in industries such as manufacturing, wholesaling, retailing, service or agricultural entities. The commercial bank provides credit and non-credit products and services to customers.

Accounts Receivable Financing/Collection

The process of obtaining funds for cash needs by selling or borrowing against accounts receivable. Cash Management Services Non-credit products and services that help meet business customers' information, transaction, and investment needs. Cash management services are also called non-credit or treasury management services, and help businesses optimize their use of cash through quicker collection and slower disbursement of business funds.

Cash Management Services

Non-credit products and services that help meet business customers' information, transaction, and investment needs. Cash management services are also called non-credit or treasury management services, and help businesses optimize their use of cash through quicker collection and slower disbursement of business funds.

Commitment Fees

Fees paid by a customer in return for the bank's legal commitment to lend funds that have not yet been borrowed by the customer. The customer pays a fee to ensure that additional funds will be available to them if needed.

Credit Scoring

A standardized credit rating system, which analyzes applicant criteria such as income, personal characteristics and other outstanding debt. The system produces a credit score intended to be indicative of a borrower's ability to repay a loan.

Lease

A contract granting the use of real estate, equipment, or other assets over a fixed period of time in exchange for payment. The borrower pays for the privilege of using the asset, and the ownership of the asset is usually transferred to the borrower at lease end.

Letter of Credit

A bank guarantee of payment for a buyer or seller of goods. Payment for goods is guaranteed and will be made when the terms of a contract have been met. Payment is also guaranteed if the applicant for the letter of credit fails to uphold the terms of the contract.

Loan Syndication Fees

Fees paid by a customer to a group of banks (called a syndicate) that acts jointly to loan money to the customer. A syndicate may result from a loan amount that is too large or too risky for one bank to make by itself.

Margin

The amount of funds that must be deposited when purchasing securities. Also, the equity in an investor's account.

Matched Funding

A process of matching maturing liabilities (customer deposits) with maturing assets (loans or investments) to manage the costs associated with fluctuating interest rates. As an example, a bank would try to fund a 3-year loan with a 3-year deposit to lock in the interest rate it will earn for the three-year period.

Relationship Manager

Relationship managers typically serve as the primary contact between the bank and customers. Relationship managers may also coordinate services from other areas of the bank for cross-selling opportunities.

Revolving Line of Credit

Money available to a customer to meet the on going needs of a business. A line of credit is similar to the concept of a credit card. Money is borrowed and repaid frequently. The bank's commitment is usually short term (less than one year), but it may be renewed yearly.

Securitization

The process of aggregating debt instruments, such as loans, in a group and issuing stocks or bonds backed by the group.

Spread

Income earned after interest expense. A bank earns interest income on its loans (for example, 9% lending rate) and incurs interest expense on the funds used to make loans (for example, 3% paid to depositors). The difference between the two rates is the bank's spread (9% - 3% = 6% spread).

Term Loan

A loan that is made to finance an asset over a period of time. The bank's commitment is usually long term (greater than 1 year) and is usually a secured loan.

Consumer Lending Services

Consumer lending services provides loan and credit services to retail customers. The goal of lending is to target customers who are able to repay the bank and to eliminate risky customers. The bank generates revenue from the interest on the loan.

Adjustable-Rate Mortgage

A property loan that is subject to periodic rate adjustments. The exact method, frequency, and basis for the rate change are based on the investment it is tied to, which is usually a widely published market/base rate of interest or index.

Authorization

Process that ensures that the credit card and cardholder are valid and have enough credit to pay for the item. An authorization occurs electronically at the point of sale, taking only seconds to complete.

Bad Debt

Open account balance or loan receivable that is proven uncollectable and written off.

Charge Card

A short-term funding mechanism. Charges must be paid in full at the end of the monthly cycle. An example is a corporate charge card.

Collateral

An asset pledged to a lender until a loan is repaid. If the borrower fails to make payments as required, the lender has the legal right to seize the collateral and sell it to pay off the loan.

Credit Scoring

A standardized credit rating system, which analyzes applicant criteria such as income, personal characteristics and other outstanding debt. The system produces a credit score intended to be indicative of a borrower's ability to repay a loan.

Fixed-Rate Mortgage

A property loan in which the payback rate does not vary with market conditions; it is constant over the life of the loan.

Home Equity Loan

A loan, which is secured by an independent appraisal of the property value of the customer's home. Essentially, the customer is borrowing against the equity value of their home that is not mortgaged.

Installment Loan

A consumer loan that has a predetermined loan maturity date (the date the loan is to be paid in full) and repayment schedule.

Loan-to-Deposit Ratio

Percentage representing a bank's total amount of outstanding loans to the amount of deposits.

Overdraft Credit

A line of credit attached to a checking account. When a customer withdrawals more money than in his account, the extra amount is covered by the overdraft credit. The customer pays interest on the overdraft amount.

Revolving Credit

Loan amount requiring monthly payments of less than the full amount due, with the balance carried forward subject to additional interest.

Branch/ATM

For some banks, branches are the primary distribution channel for selling products and services. Although differences exist among banks and across regions, most branches offer a range of products and services, including deposit accounts, transactional services, lending services, investment services, and insurance.

Commission

Fee paid to sales personnel for executing a trade based on the number of shares traded or the dollar amount of the trade.

Compensating Balance

Minimum balance required by a bank on certain deposit accounts. Banks require customers to keep these balances as a stable source of funds for the bank, and impose financial penalties if the account goes below the minimum requirement.

Computer Switch

ATM back-end equipment that allows network banks to share customer account information. These switches allow customers to use any participating ATM, and allow the network banks to verify funds for non-customers.

Deposit Accounts

These accounts allow customers to deposit funds that the bank must repay the customer at some future date.
Two major types:

1. Funds committed to the bank for a specified period of time. In return, the bank offers a higher interest rate relative to other types of deposits. A certificate of deposit is one common form.
2. Funds committed to the bank, which the customer has the option to withdraw at any time. A savings account is the most common form.

Insurance

Most countries have an independent insurance industry. In many countries, however, the insurance and banking industries are merging, resulting in the selling of insurance products in banks. Some of these products include life insurance, property insurance, and casualty insurance.

Investment Services

Banks offer services to help customers invest their excess funds. Some of these services include brokerage services, mutual funds, annuities, personal trust, and asset management.

Lending Services

Banks provide funds to customer while charging customers interest for the use of those funds. Some examples include mortgages, home equity loans, overdrafts, and credit cards.

Premium

Fee paid to a bank or insurance provider for insurance coverage.

Transaction Services

These accounts allow customers to deposit funds with the intention of making non-cash payments in the near future, usually with instruments such as a check or debit card.
Two major types:

1. Funds may be withdrawn on demand without advance notice. A checking account is the most common form.
2. Some accounts may carry a market interest rate and allow the depositor limited check-writing privileges. A money market deposit account is one common form.

Transfer Pricing

The internal charge to a business unit for goods or services provided by another business unit. An example of transfer pricing would be the deposits line of business charging a lending division for using customer deposits as the source for loans.

Electronic Banking

Electronic banking is one type of distribution channel used to deliver products and services to customers. It brings the bank to the customer-a strategy for penetrating different customer segments, encouraging customer loyalty, and reducing costs. Within electronic banking, several different types of delivery mechanisms exist, such as VRU, digital TV, mobile phone, kiosk, PC, and the Internet.

Digital TV (DTV)

To leverage the technology of interactive TV, some banks use the platform of the digital TV operators, including satellite and cable, to offer home banking services. DTV combines video, images, sound and textual information. The navigation requires the use of only five keys on the standard pay-TV remote control. Customers can browse through account information directly on the TV and perform transactions, such as transfers, blank check orders, or financial product subscriptions.

Electronic Bill Presentment and Payment

EBPP provides customers or businesses with the ability to make "electronic" payments to any vendor or any person at any time. Instead of receiving a paper bill monthly, customers receive an electronic version of the bill. Customers can review bills and pay them online electronically, eliminating the hassle of check writing and the possibility of lost payments or late charges.

Financial Institutions

Financial Institutions include banks, insurance companies and brokerage houses.

Internet Banking

Internet banking offers the ability for customers to conduct basic bank transactions (for instance, balance transfers and account inquiries) through direct dial-up connections to the bank. Unlike PC banking where access is limited to a PC equipped with the bank's software, Internet banking allows customer access to information and services from any location with an Internet connection.

The goal for a bank's Web site is to provide many, if not all, of the services offered at a branch-transactions, account information, advice, administration, and even cross-selling. The interactive nature of the Web not only allows banks to enhance these core services, but also enables banks to communicate more effectively and strengthen customer relationships.

Kiosk

The kiosk multimedia application is based on a virtual branch metaphor. Existing or potential customers can "enter" the branch by touching the doors on the kiosk's touch screen. Inside the virtual bank branch the customer can obtain information and go through the sales process of products. The customer will be guided by a virtual host and sales personnel through integrated video sales person and promotion video clips. The application offers a full range of retail banking products and services. In cases where real signature is required by law, the customer can print out the application for signature and submission.

Mobile Phone

Banking using the mobile phone is similar to using a touch-tone phone. Customers can access basic transactions such as check balances, loan rates, checks paid, deposit rates, and credit card payments. Many providers also offer Internet banking over Internet enabled mobile phones.

Origination

Origination refers to the customer acquisition process.

PC Banking

Customers can access accounts by installing the bank's application on the PC. This allows customers to access account information, pay bills, transfer funds, or to purchase investments while online. Some major providers of personal management (PFM) software are partnering with banks to offer additional services. This software may offer additional features such as creating account registers and checkbooks, balancing accounts, setting up loan amortization schedules, creating budgets, tracking investment performance, and estimating taxes.

PDA

Banking using a wireless PDA (personal digital assistant) offers all the capabilities of banking on the Internet (e.g., check balancing, transferring balances, paying bills, etc.), but has a scaled down user interface. It is very similar to doing Internet banking over a mobile phone.

Voice Response Unit (VRU)

An automated system accessed by telephone for providing basic customer service information, such as account balances or transaction history. Call centers provide VRU's as a way to reduce cost, since nearly 80% of customer inquiries can be handled through a VRU, and do not require the more costly CSR interaction.

White Label Solutions

A white label solution is a solution provided by a bank that can be branded and used by the end customer like Accenture. Accenture employees could use the 'Accenture Bank', even though all of the processing takes place at large bank, the web interface to the banking functions would be branded as Accenture.

Call Center

Call centers are lower cost delivery channels than branches to maintain high quality customer service. Call centers allow other delivery channels, such as branches, to focus on marketing and selling instead of service. Major functions supported by the call center include credit card services (such as transaction and balance inquiries), retail and commercial account information (such as transfers or payments), basic customer service (such as change of address) and website support.

Inbound

Telephone calls made from either a customer or an employee to the call center.

Outbound

Telephone calls made from the call center to a customer or potential customer.

Voice Response Unit (VRU)

An automated system accessed by telephone for providing basic customer service information, such as account balances or transaction history. Call centers provide Vero's as a way to reduce cost, since nearly 80% of customer inquiries can be handled through a VRU, and do not require the more costly CSR interaction.

Distribution Channels

Banks use a variety of channels to manage products and services. Banks must define policies for new and existing channels, determine the distribution strategies, support each channel's operations, and manage the branch network across regions.

While the branch is the most visible distribution channel, alternative channels, such as Internet banking and ATM's have grown substantially in the past decade. New channels have emerged due to increasing customer sophistication in technology and product knowledge, competition for new customers, the need to lower operating costs, to increase time in the branch for commercial activities and evolving technology.

Multi-Channel Strategy

Retail banks use a multi-channel approach to serve a wide range of customers. Channels include electronic banking, branch, ATM and call centers. This diversification increases bank revenues while lowering bank costs.

Strategy & Marketing

The purpose of strategy is to define the bank's sales objectives and target markets, and to determine product needs and enhancements. Marketing implements promotional strategies that support these initiatives and builds brand identity for the bank.

Private Labeling

A bank's purchase of a product from a third party for use and sale under the banks own name. Other banks can purchase the same product and private label it for their own uses. Although private labeling reduces product development costs, banks usually choose to develop products internally, rather than through private labeling, since new products often differentiate banks from their competition.

Product Bundling

Bundling offers a complementary group of products or services to a customer. Products and services may be bundled as a way to increase sales of a particular product (such as, bundling a poorly selling savings account with a popular checking account), or to target specific market segments (such as a group of products targeted at young professionals).

Risk Management

Risk management is the framework and related infrastructure by which banks plan for, acquire, and monitor risk to achieve superior performance. Risk management integrates human resources, technology, capital, and target marketing to balance risk against reward. Categories of risk include economic, credit, liquidity, and operational risk.

Asset-Liability Management

Matching the maturity of deposits with the length of loan commitments to avoid adverse affects from changes in interest rates.

Business Risk

Banks analyze a business customer's ability to repay a loan. Loan review and approval by more than one individual helps offset business risks.

Credit Risk

The uncertainty of realizing the return of principle and the expected reward from a customer or portfolio of customers.

Credit Risk

The uncertainty of realizing the return of principle and the expected reward from a customer or portfolio of customers.

Economic (Market) Risk

Uncertainty associated with currency, local, regional, global, business and political conditions. Examples include exposure or concentrations by industry, loan structure, product, interest rate, or sovereignty.

Individual Risk

Banks have replaced the manual review process for individual and small business loans with new analytical tools. Credit scoring and behavioral modeling are used to standardize the evaluation of loan applications and to project future risks.

Industry Risk

Loans are classified by industry, and limits are established on loan amounts outstanding by industry to offset industry risks.

Liquidity Risk

The ability to convert assets into cash or cash equivalents without experiencing significant loss.

Loan Loss Analysis

Analysis of non-performing and problem loans, including probability of default, loss rates by risk rating, recovery rates, and time from origination of the loan to non-performing status.

Non-performing Loans

Loans for which customers are not making scheduled payments. Commercial loans that are 90 days past due and consumer loans that are 180 days past due are usually classified as non-performing.

Operational Risk

Risk of loss associated with the failure to execute transactions due to:

1. Inadequate controls
2. Operating errors
3. Inaccurate information
4. Ineffective management
5. Natural disaster
6. Fraud
7. Regulatory issues
8. Legal considerations

Risk-adjusted Return on Capital (RAROC)

A technique for risk analysis that requires a higher net return for a riskier project than for a less risky project. The risk adjustment is performed by reducing the risky return rather than adjusting the charge for capital.

Risk Appetite

Industry term used to describe the general level of risk acceptable to a bank. A "small risk appetite" would mean that a bank would have little interest in a risky transaction.

Risk Rating

A rating system based on a scale of loan performance and loss experience. Limits are established for each rating level for individual, business, industry and country risks. Stress Testing and "What If" Scenarios Simulation of the impacts of trends and events (such as, interest rate changes, regulatory changes, economic growth, currency fluctuations) on a loan portfolio.

Finance and Corporate Planning

Finance and Corporate Planning provides external customers with financial and credit quality information. External customers include regulators, shareholders, and market analysts who monitor financial performance and stability. Finance and Corporate Planning also provides internal customers with financial management reporting. Internal customers include the CEO, the Board of Directors, and line of business leads.

Activity-based Costing

A method of determining the total cost of providing a service to customers. These costs are aligned to associated revenues by product, customer, and channel. Finance and corporate planning uses activity-based costing information to help determine product prices and set interest rates on savings products.

Capital Adequacy

A measure of an institution's capital (equity) in relation to its asset base. A higher capital adequacy means a stronger ability to absorb losses. A bank's capital usually ranges from 2-12% and varies based on factors, such as size and regulatory requirements.

Credit Quality

A measure of the quality of an individual loan or a loan portfolio. Good credit quality means that the customer continues to make full loan payments on time and is not at risk for defaulting on the loan.

Efficiency Ratio

A measure of how well a bank is managing its non-interest expenses. The lower the efficiency ratio, the better the bank is performing. This ratio is measured by dividing non-interest expense by the sum of the net interest income and the non-interest income. For banks, typical efficiency ratios are .55 or lower.

Earnings Per Share (EPS)

Earnings per share are defined as net income divided by the total number of outstanding shares of common stock. This ratio measures the net income earned for each share of stock outstanding.

Net Interest Income

Net income earned from the bank's lending activities, such as commercial loans, credit cards, mortgages, etc. Two-thirds of a bank's total income is usually derived from net interest income.

Non-interest Income

Income earned from non-lending services offered by the bank. Some examples include service fees and stock exchange commissions. Non-interest income is usually one-third of a bank's total income.

Return on Assets (ROA)

Return on Assets is defined as net income divided by total assets. This ratio identifies how well assets are working to produce income. Banks try to average a minimum ROA of at least 1%.

Return on Equity (ROE)

Return on Equity is defined as net income divided by total equity. This ratio identifies how well a company's equity is working to produce income. Banks try to average a minimum ROE of at least 15%.

Banking Services

The goal of banking services is to provide quality customer service, and efficient and accurate processing of information and materials.

Acquiring Bank

The financial institution where the check is deposited into the payee's account.

Check Truncation

Process in which the information on a check is converted into electronic images. Checking account customers receive images of checks they have written, rather than the physical check.

File Folder Imaging

Process in which documents are converted into images. The images are stored in electronic file folders, and are used the same way paper documents are used.

Issuing Bank

The financial institution that issues the payor's check.

Magnetic Ink Character Recognition (MICR) Encoding

Magnetic Ink Character Recognition Encoding allows high-speed, check-sorting machines to read and sort checks. Data is encoded onto the checks in this process.

Off-line Processing

ATM or debit card reader does not have a direct link to the customer's account file. Instead, the purchase information is captured for later processing, and does not immediately post to the account. Few off-line systems are used today.

Online Processing

ATM card or debit card readers are linked directly to the customer's files, which contain their individual account information. At the point of sale, these transactions link directly to the bank to determine whether the customer has funds to cover the purchase or a cash advance.

Point of Sale (POS)

Retail payment system that substitutes an electronic transfer of funds for cash, checks, or drafts in the purchase of retail goods and services.

Society for Worldwide Interbank Financial Communication (S.W.I.F.T.)

A predominant European payment processing system, which provides the link between, the issuing bank and the acquiring bank on an international level. The standardization of the Euro may change the payment processes between participating countries.

Smart Cards

Credit cards with microcomputer chips. Smart cards are a recent trend that will have a substantial impact on future banking. These cards offer a secure, personalized connection to Internet shopping and banking and are being considered for multiple purposes across different industries.

Private Banking

The traditional private banks target high net worth (HNW) individuals and provide customized, personal services for these clients. Although the minimum requirement varies among banks, they typically define this specific customer segment as individuals with assets under management of USD one million or more. Products offered include credit and deposit products, investment products, estate and financial planning services, and trust products.

Advisory Management

Accounts that require clients' authorization before executing investment decisions.

Discretionary Management

Accounts that give the relationship manager full authority to make investment decisions.

Fully-Dependent Investment Style

Clients who are unable to manage their own investment decisions, for different reasons, rely on professionals who know them well and are willing to pay for the convenience of doing so.

High Net Worth Clients (HNW)

Individual clients whose total assets under management are USD one million or more. Offshore Private Banking Private banks that focus on wealth held abroad by residents globally; these traditionally have been focused on avoiding taxes or foreign exchange restrictions.

Onshore Private Banking

Private banks that focus on services for individuals banking in their own country.

Opinion-Seekers Investment Style

Clients who are able to manage their own money but like to receive advice at times. Although they are fairly price-conscious, they will pay extra for the things that they value. Opinion seekers may shop around multiple providers for the best information.

Self-Directed Investment Style

Clients who consider themselves financially savvy, who are price-conscious and who do much of their own investment decision-making and money management.

Bank Performance

Bank performance is important to internal and external stakeholders. A bank's performance is often compared to other banks of similar size and product offerings.

Annual Revenue Growth

The increase in revenues from one year to the next, calculated by subtracting the previous year's revenues from the current year's revenues, and dividing by the previous year's revenues. Banks target revenue growth of 5-10% per year.

Asset and Liability Management

A technique necessary to counter fluctuations in interest rates and exchange rates. The bank's Risk Management group typically has responsibility for asset and liability management. Financial instruments such as options and swaps, which allow banks to limit the adverse affects of rate movements, are used to manage interest rate risk. Matching foreign currency loans with deposits in the same currency when appropriate is one way to manage exchange rate risk.

Efficiency Ratio

A measure of how well a bank is managing its non-interest expenses. The lower the efficiency ratio, the better the bank is performing. This ratio is measured by dividing non-interest expense by the sum of the net interest income and the non-interest income. For banks, typical efficiency ratios are 55% or lower.

Factoring

A type of financial service whereby a company sells its accounts receivable to a financial institution. The financial institution collects the debt or payments for the receivables, and assumes the risk that the receivable may not be repaid.

Gross Domestic Product (GDP)

A measure in the change of market value of all goods services and structures produced in the economy. This quarterly statistic shows the actual pace at which the economy is growing or shrinking. Personal consumption expenditures typically account for roughly 68% of the GDP. Investment, government spending and net exports account for the rest.

Inflation

The rate at which the general level of prices for goods and services is rising.

Interest Income

Net income earned from the bank's lending activities, such as commercial loans, credit cards, mortgages, etc. Two-thirds of a bank's total income is usually derived from net interest income.

Non-interest Income

Income earned from non-lending services offered by the bank. Some examples include service fees and stock exchange commissions. Non-interest income is usually one-third of a bank's total income.

Provision

Amount charged against the profit and loss statement (income statement) to reserve against loan losses or losses/charges. Bank management will make a "provision" for losses when they believe they will be unable to collect the loan amounts due from customers. When the loan is "officially" not repaid, the loan amount is "written off". If the original provision was insufficient to cover the loss, an additional charge against income will be taken. If the original provision was greater than the write-off, the bank will either credit income or apply the provision against another loan.

Return on Equity (ROE)

Return on Equity is defined as net income divided by total equity. This ratio identifies how well a company's equity is working to produce income. Banks try to average a minimum ROE of at least 15%.

Unemployment rate

Percentage of a country's workforce actively looking for work, but unable to find jobs. This metric is one of the most closely watched government rates, because it has long been thought to often give the clearest indication of the economy's direction. A rising unemployment rate is seen as a sign of a weakening economy, which may call for an ease of government monetary policy. A drop in the employment rate, on the other hand, shows that the economy is growing, which may spark fears of high inflation, in which the government may increase interest rates. While this has proven true quite often over the years, this theory did not hold true in the 1990's when the economy grew rapidly, yet both the unemployment rate and inflation remained below 6% for many years. Thus, rapid growth and the low unemployment rate did not spark fears of high inflation and increases in interest rates during much of the 1990's.

Weighted Risk Assets

To calculate the level of regulatory capital required, a bank's loans are first converted into weighted risk assets (WRA's). Amounts by loan category (such as mortgages, credit card loans, or loans to other banks) are multiplied by a weighting percentage that reflects the different levels of risk for each lending type. As an example, credit card lending is assigned a weighting of 100%, while loans to other banks are weighted at 20%, to reflect the higher risk of credit card lending relative to lending to a bank. The bank must retain a certain percentage of its capital based on the weighted amounts outstanding.

Write-off

Charging an asset amount, such as uncollected debt, to expense or loss. Bank management will make a "provision" for losses when they believe they will be unable to collect the loan amounts due from customers. When the loan is "officially" not repaid, the loan amount is it "written off". If the original provision was insufficient to cover the loss, an additional charge against income will be taken. If the original provision was greater than the write-off, the bank will either credit income or apply the provision against another loan

TELECOMMUNICATION

Telecommunication refers to long distance communication. The Greek word "tele" means "far off". At present, such communication is carried out with the aid of electronic equipment such as the radio, telegraph, telephone, and television. However, in earlier times, smoke signals, drums, light beacons, and various forms of semaphores were used for the same purpose. The information transmitted can be in the form of voice, symbols, pictures, data, or a combination of these. The physical equipment for a telecommunications system includes a transmitter; one or more receivers; and a channel or means of communication such as air, wire, cable, communications satellite, or some combination of these.

Components in a telephonic conversation

The telephone is one of the most amazing devices ever created. If you want to talk to someone, all that you have to do is pick up the phone and dial a few digits. You are instantly connected to the person with whom you would like to talk, and you can have a two-way conversation. The telephone network extends worldwide, so you can reach almost anyone on our planet.

The actual process of making a call involves several functions that are activated with the help of hardware and software components in the telecom network. The network comprises of telephone sets, transmission media such as cables, and switching systems.

Making a call

The telephone cannot by itself set up a call between two subscribers. Most of the functions occur in cooperation with the telephone switching equipment (exchange).

When an individual lifts the cradle, a signal is sent to the exchange indicating that the user wants to make a call. This signal is known as the hook off signal. The signaling process between the exchange and the telephone set commences in this manner. The exchange checks for certain parameters to authenticate the subscriber, and sends the dial tone to the user's telephone.

When the user dials a number, the exchange gets the destination address of the called subscriber. The numbering system within the exchange routes the call to the called subscriber. The exchange then determines the status, whether available or busy, of the called subscriber. Based on the status, the exchange sends either a ring back tone or a busy tone to the calling subscriber and the ring tone to the called subscriber. If the calling subscriber is available, a voice path is created and the two parties can now communicate with each

Analog and Digital Signals

Speech signals may be transmitted through the medium either as analog or digital signals.

An analog signal is a continuously varying smooth electromagnetic wave that can be propagated over a variety of media, including multi-wire cables, coaxial cables, and fiber optic cables. When a person speaks into the transmitter on a telephone, voice signals made up of sound energy are converted into electrical energy for transmission along a transmission medium. The conversion from voice to analog signal is accomplished inside your telephone.

Digital signal A digital signal is a sequence of discrete voltage pulses that are transmitted over a wire medium. The state of the digital signal is characterized by abrupt changes. For digital signaling, voice or data is converted to a stream of binary information (0s and 1s) for transmission. Instead of the voice signal being processed as an analog signal, it is converted into a digital signal and handled with digital circuits throughout the transmission process. When it arrives at the exchange that serves the called telephone, it is converted back to analog to reproduce the original voice transmission.

Introduction - Switching Systems

It is practically impossible to connect all telephones to each other using cables. This led to the introduction of switching systems, commonly known as exchanges.

With the introduction of switching systems, the subscribers are not directly connected to each other. Instead, all subscribers are connected to some switching system. Therefore, only one link is established between the subscriber and the switching system. When a subscriber wants to call another subscriber, a connection is established between the two at the exchange. This helps in administration, maintenance, and overall control of the telecommunication network.

Types of switching systems

The initial switching systems were manual. Many operators were involved with switching calls manually between various subscribers. As the number of subscribers increased, the need for automatic exchanges was felt. With the advent of computers, the exchanges were made completely digital.

Types of switching systems

- Manual
- Automatic
    - Electromechanical
    - Electronic (Stored Program Control)

Manual Switching System

The manual exchange allowed the caller to alert the exchange operator, and the operator to alert the call recipient. The operators, on receipt of this request from the caller, connected the caller's line to another line, and noted the call duration for billing purposes.

When a caller lifted the receiver, an electrical signal was transmitted down the line, causing a small metal shutter on the operator's board to drop down. A quick scan of the board showed the operator the lines that needed attention. A magneto bell, devised by Bell's assistant, Thomas Watson, was used to alert the person being called. To connect two lines, the manual exchange had a series of horizontal and vertical metal bars. Each line was connected to one horizontal bar and one vertical bar. At every cross point, each bar had a hole. To connect line 3 to line 14, the operator would push a metal peg into the hole where bar 3 crossed bar 14. That allowed the current to flow between the two lines and the conversation could begin.

Disadvantages of the manual exchange

- The manual exchange was unable to handle the increasing number of calls as subscribers grew in number.
- The manual exchange was unable to provide privacy as far as the subscriber was concerned.
- It was costly to handle long distance calls manually. Hence, further inventions resulted in the automatic exchange.

Automatic Exchanges

Due to the disadvantages of manual exchanges, automatic exchanges were invented. Automatic exchanges can be classified as Electromechanical and Electronic.

The electromechanical exchanges transmitted sound in the form of analog signals. Further, the electromechanical switching systems can be classified into

- Strowger (Step-by-step) exchange
- Crossbar exchange

Electronic Exchanges

The crossbar systems were found to be slow in call processing. It was realized that components of switching systems could be replaced by a single digital computer. In electronic switching systems, a computer performs the control functions. Hence, these systems are called Stored Program Control (SPC) Systems.

Stored Program Control Stored-program control is a broad term designating switches where common control is carried out to a greater extent or entirely by computer hardware. Control functions can be carried out entirely by a central computer, or partially or fully by distributed processing.

There are four basic functional elements in the SPC switching system.

- **Switching matrix:** The switching matrix is an abstraction of a switch. The function of a switch is to connect the calling and the called party. The method of connection differs based on the topology used. The infrastructure required for the connection is called the switching matrix.
- **Call store:** This is a temporary storage of incoming call information ready for use on command from the central processor. The call store is also referred to as 'scratch pad' memory. It also contains data on the availability and status of lines, trunks, service circuits, and internal switch circuit conditions.

- **Program store:** The Program store contains the software needed for the operation of the switching system.
- **Central processor:** The Central processor is the nerve center of the switch. It controls the operation of various entities such as the dial tone detector, the ring generator, and the switching matrix.

Digital Technology

The idea of digital technology is to convert analog signals to digital 1's and 0's. Sound is a set of analog events, which can be represented in a curve. A continuous analog wave can be converted to an equivalent digital mode. This process of modulation is called Pulse Code Modulation (PCM). The PCM Process is illustrated in the figure below.

A single conversation does not utilize bandwidth to the maximum extent. To utilize the bandwidth to the full extent, you can combine many such conversations to transmit through a same line by the process of Multiplexing.

The conversion of an analog voice signal for digital transmission involves three steps.

- **Sampling:** Sampling is the process of taking readings (or 'samples') on the speech curve. It is important to establish a set of discrete times at which the input waveform is sampled. The Nyquist Criterion (Nyquist theorem) specifies how frequently readings need to be taken. Thus, a suitable sampling frequency is determined. The result is a Pulse Amplitude Modulated (PAM) signal, where each pulse corresponds directly to the amplitude on the speech curve.
- **Quantization:** Quantization assigns a binary value to each voltage sample. A PAM waveform could have an infinite number of distinct voltage values between the maximum and minimum values of the voltages. To assign a different binary sequence to each voltage value, a code of infinite length would be required. To avoid this, we go in for quantization. The number of bits used for transmission depends upon the number of quantum values. For example, eight bits are sufficient to form a unique code for each quantum value when the number of quantum values is 256 (2^8 = 256). Of these, the first bit is used to denote the polarity of the sampled signal. The equipment needs only to differentiate between these digits, which are sent in the form of pulses.
  During quantization, the Quantizer
  - Limits the input signal amplitude between upper and lower voltage values - the dynamic input range of the ADC.
  - Sets a given number of discrete values between the maximum and minimum voltage limits.
  - Compares each sample with a set of quantum values in turn and assigns it to the one it approximates.

  Thus by definition, all samples in an interval between two quantum levels are deemed to have the same value.

The output of the ADC is in serial digital format, which is a unipolar Binary sequence of 0's and 1's. This sequence cannot be directly fed to any media such as PCM, Optical Fiber Cables (OFC), or Microwave (MW). Therefore it is given to the Line Coder, which encodes it in one of the following formats: AMI, HDB3, CMI, RZ, or NRZ.

- Coding: PCM is not specially designed to digitize speech waveforms. Speech waveforms have a lot of redundancy that can be used to design coding schemes. Speech signals typically have non-uniform amplitude distributions, sample-to-sample correlation, periodicity or cycle-to-cycle correlation, and speech pauses or inactivity factors. PCM outputs very high bit rates, which consume a lot of bandwidth. To overcome this problem, new voice coders such as ADPCM and LPC were developed. They give much lower bit rates and are therefore bandwidth efficient.

Digital Switching Systems

In digital switching systems, the operation is automatic and information transfer is digital. Digital switching systems can provide a better quality of service, as the time needed to connect subscribers is reduced. It can also be used for large exchanges. Here, the computer carries out the control functions of a telephone exchange. A set of instructions is stored within the computer's memory and these are executed automatically by the processor.

Digital Switching is the process of interconnecting time slots between many links. This enables digital routes at 2 Mbit/s or 1.5 Mbit/s, from other exchanges or digital PABXs, to be directly terminated on the digital switch. Digital switching involves the transfer of the PCM signal relating to the channel in a time slot on an inlet bus to a time slot on an outlet bus.

Digital switching system includes two types of switching systems.

- Digital Space switching
- Digital Time switching

Introduction - Call Process

You have learned that a telephone conversation involves many functions other than picking up the receiver and dialing the number of the subscriber.

The calling party's touch-tone telephone sends digits as tones instead of pulses through the subscriber line to the called party's telephone. The incoming call information is stored in the call store. The central control uses the program store to route the call to the destination. If the called party answers, the connection is established, and the connection terminates when one of the parties goes on-hook.

Numbering Plan

With the increase in number of subscribers, it became necessary to identify each subscriber to set up the call faster. The numbering plan provides every subscriber and every service including call forwarding, call waiting and call queuing, in the telephone network with a unique and simple code, which allows automatic call set up in the network.

It is necessary that each subscriber be able to call any other subscriber. This requires each subscriber to have a unique code. Also, for the convenience of the subscriber and the telephone exchange, a simple code is needed.

The need for a well-structured numbering plan is necessary to:

- Identify the calling subscriber in order to charge the call
- Guarantee expansion in line with the increase in the number of subscribers without sweeping changes in the numbering
- Create a long term numbering system for subscribers and services
- Create subscriber numbering series with as few digits as possible
- Create a uniform and simple method of dialing numbers for the whole country
- Create simple number codes for specific services, which are the same throughout the country Simplify coordination with other numbering plans

International Numbering Plan

The objective of a international numbering plan is to uniquely identify each subscriber connected to the telecommunication network. An international numbering plan or world numbering plan has been defined by CCITT. For numbering purposes, the world is divided into zones. Each zone is given a single digit code. The European zone has been assigned two codes as a large number of countries fall within this zone.

Every international telephone number consists of two parts:

- **Country code:** The Country code is the combination of one, two, or three digits characterizing the called country. The first digit in the country code is the zone code of the zone in which the country lies. For example, in zone 9, India has the country code '91' and Maldives '960'. The zone code 9 is the first digit in the country codes of both India and the Maldives.
- **National Number:** The number of digits in an international subscriber number is limited to a maximum of 12. However, in practice, world numbers are limited to 11 digits. Therefore, the number of digits in the National Number is limited to 11 - n where n is the number of digits in the Country Code. Therefore, the national number can contain 9-11 digits.

The National Number consists of 3 parts:

- **Area/Trunk code:** The Area or Trunk code is the digit or combination of digits that identifies a particular numbering area of the called subscriber in a country. This code determines the routing for a trunk call and the charge for it. <LI**</LI>Exchange Code: The exchange code identifies a particular exchange within a numbering area.**
- **Subscriber Line Number: The subscriber line number is used to select the called subscriber line at the terminating exchange. In CCITT terminology, the combination of the exchange code and the subscriber line number is known as the subscriber number. This is the number listed in the telephone directory.**

PABX (Private Automatic Branch Exchange)

PABX is a device installed on the customer's premises, that enables switching of multiple incoming and outgoing lines between multiple internal phones.

Business subscribers often have specific requirements for their telephone numbers. They must be simple to remember, and must also allow direct access to individual extensions after dialing the basic number. For example, at Sharda center, you would want to allow callers to directly dial the extension of a particular employee after calling the Sharda number.

From the point of view of telephone administration, the PABX (Private Automatic Branch Exchange) is a subscriber with several extensions, which is accessed by a number.

Routing

Routing is the process of delivering a message across one or more networks via the most appropriate path.

There are basically two main methods of routing:

- **Fixed routing: The call is routed based on a pre-defined route. There is no change in the route regardless of the time of the call, availability of the route, or call statistics.**
- **Dynamic routing: It is necessary to find optimal routing paths for the transport of information.**

In dynamic routing, the routing pattern varies based on factors such as:

- **Time: Calls are routed depending on the time of the day. For example, normally you may be using route b to route calls. However, between 8 to 10 a.m, you may decide to route 50% of the calls made via route a and 50% via route b. These routing patterns are pre-planned and are implemented at a fixed time of the day.**
- **State: Routing patterns can be varied based on network information collected by toll exchanges. For example, if a certain route or part of it is found busy, the toll exchange can route your call via another route.**
- **Event: Certain events such as natural calamities or special occasions such as Mothers Day can affect the call statistics. Routing patterns are updated locally**

based on call statistics. Once you dial a number, all exchanges involved in routing the call have to communicate amongst themselves. This is done through the process of signaling.

Introduction - Signaling

In telephony, signaling refers to the passing of information and instructions from one point to another, relevant to the setting up of a telephone call.

To initiate a call, the caller lifts the handset off the hook. This off-hook state is a signal to the exchange to be ready to receive the number of the called subscriber. As soon as appropriate receiving equipment is connected to the line, the exchange signals the dial tone to the calling subscriber. The subscriber can then dial the destination number. The subscriber receives either a ringing signal, an engaged or busy tone signal, or an equipment busy tone indicating congestion between the called exchange and the calling line. These are signals with which the telephone subscribers themselves are concerned.

Signaling involves sending signals between various switching systems, transmission systems, and subscriber equipments in a network.

Signaling Tones

In older exchanges, information is passed via a rotary dial by a series of makes and breaks of the subscriber's loop, interrupting current flow. In modern exchanges, voice-frequency musical tones are sent to the exchange as push buttons are pressed. These tones are usually called DTMF (Dual Tone Multi-frequency) as each time a button is pressed, two tones are simultaneously sent out to the line. Distinctive signaling tones are provided in all automatic exchanges.

Typically, there are four distinct signaling tones:

- **Dial Tone: The Dial tone is used to indicate that the exchange is ready to accept the digits dialed by the subscriber. The subscriber should start dialing the number only on hearing the dial tone. Otherwise, the initial digits of the dialed number may be missed by the exchange.**
- **Ringing Tone: When the line of the called subscriber is obtained, the exchange sends the ringing current to its telephone set. Simultaneously, the exchange sends a Ringing tone to the calling subscriber too. The ringing tone and the ringing current are two similar, but independent quantities, having a double-ring pattern. Therefore, it is possible that the calling subscriber hears the ringing tone, whereas the called subscriber may not hear any ring.**
- **Number unobtainable tone or Equipment busy tone: The Number unobtainable tone may be sent to the calling subscriber due to a number of reasons. The called subscriber line may be out of order or disconnected, or there may be an error in dialing. It may also be due to congestion between the called exchange and the calling line.**

- **Routing tone or Call-in-progress tone: The Routing tone or Call-in-progress tone is heard when the call is routed through a number of different types of exchanges.**

Signaling Techniques

The telecommunication network classifies signaling techniques on the basis of the following:

Type of signaling information

- **Local loop signaling: Local loop signaling refers to signaling in which signals are transmitted between telephone equipment and the exchange. For example, the Off-hook, On-hook, Seize, and Clear signals are transmitted over the local loop.**
- **Selection signaling: Selection signaling conveys information relating to the routing of the call.**

Originating and terminating points of signaling

- **Subscriber signaling: Subscriber loop signaling depends on the type of telephone instrument used, such as rotary dial telephone or a dual tone multi-frequency telephone.**
- **Inter-exchange signaling: This type of signaling is internal to the switching system. It depends on the type and design of the switching system.**

Physical path used to transmit signaling information

- **Channel associated signaling: Signaling information is transmitted in the same path (channel) where speech signals are transmitted. In the case of SPC (Stored Program Control) exchanges, transmission of signals is in the same path, but control is independent in the exchange.**
- **Common channel signaling: In Common Channel signaling, the physical tie between the signaling path and the traffic circuit is removed. All signaling transfer, related to a transmission link, takes place over a dedicated signaling channel. Hence, a common-signaling channel handles the transfer of signaling information for numerous traffic circuits.**

Technology

- **Loop disconnect signaling**
- **Voice frequency signaling**
- **Multi-frequency signaling**
- **Common channel signaling**

Introduction - Networks

Accurate network organization is very essential for call processing and routing. To have a well-organized network, you need to plan the network carefully. Over the years, network planning has kept varying with respect to the number of subscribers.

In the beginning, there were individual exchanges that connected limited subscribers in a specific area. Later, the neighboring areas were connected to these exchanges. With increasing subscribers, all the exchanges needed to be networked.

Structure of the Telephone Network

Initially, all exchanges were connected to each other directly. However, the network was later modified to create a system of different levels. It was no longer necessary for every exchange to be connected to every other exchange directly.

The levels in a telephone network are:

- **The Local exchange: The subscriber lines are connected to the local exchange.**
- **The Tandem switch: The Tandem switch is used basically for the routing operation. As an exception, subscriber lines may be connected to this exchange.**
- **The Trunk exchange: Long distance calls, which need the use of trunks, go through this exchange.**
- **The International gateway: This is the highest level of exchanges. International calls are routed through this exchange.**

Subscriber Line Networks

The Subscriber Line Network is the network that connects the subscriber to a local exchange. The local exchange has to be able to handle situations including new subscribers from the newly constructed residential layouts, and subscribers moving from one place to another. Therefore, the subscriber line network needs to be very dynamic. The subscriber line network accounts for about 50 percent of the whole investment cost of the telephone network.

Your telephone must have a permanent connection to the local exchange. Having a wire on a pole and a return via the earth does this. But when the number of telephones increases, the provision for the poles and their associated wires becomes expensive. The hunt for reduced costs and smaller wires has led to the usage of paired cables. (10, 50,100 or more pairs per cable). The evolution of subscriber line network began with the concept of Two-wire or twisted pair cables.

Then came the Distribution point. The Distribution point (DP) is the point from which the cables from the exchange are connected to your telephone. Then, direct lines from the subscriber's telephones are taken.

The invention of the Cross connection point or MDF (Main Distribution Frame) followed the Distribution Point. At the Cross connection point, we can have more number of incoming subscriber lines as compared to the number of lines connecting to the exchange. For example, we can have 30 pairs terminating at the cross connection point, and only 20 going out. As the probability of all lines being busy is low, we can have this degree of concentration.

Adding subscribers to the Subscriber Line Network

If subscribers are to be added to the existing subscriber network, the following techniques are used:

- **Cable laying: You can add subscribers to the network simply by laying more cables between the cross connection points or MDFs and the exchange.**
- **Concentrator: Concentrators or 'line concentrators' that consolidate subscriber loops are remotely operated and the concentration or expansion portion of the switch is placed at a remote location. Concentration can also be effectively carried out using carrier techniques, either pulse-code modulation (PCM) or frequency division multiplexing (FDM). As all subscribers will not be accessing the exchange resources simultaneously, they can be connected through a line concentrator.**
- **Remote Survivable unit: Remote Survivable Unit (RSU) is a part of the main exchange located away from the parent exchange. It caters to the subscribers just like the parent exchange caters to them, and can also work as a standalone exchange with local loops. The RSU is set up to share the load of the parent exchange. Initialization, administration, operation, and maintenance are done from the parent exchange.**

Introduction - Traffic Engineering

The basic purpose of traffic engineering is to determine the conditions under which adequate service is provided to subscribers while making economical use of the resources providing the service. Thus, it is important to consider various factors before setting up an actual telecom network. The design of a telecommunication network is based on the projected traffic or load the network is supposed to handle.

Traffic engineering helps to determine the equipment required to provide a particular level of service for a given network. Factors such as the number of subscribers using the network, traffic generated by them, and the load on the network should be considered for traffic measurement. The load varies during the day with heavy traffic at certain times and low traffic at other times.

Tele-traffic theory

The Tele-traffic theory is one of the most important aids for controlling investment in the network.

The theory aims to provide a mathematical order to

- **Make Tele-traffic measurements**
- **Provide well-defined units**
- **Calculate relation between quality of service and system capacity**

In Tele-traffic theory, certain probability distributions are derived. These are designed as mathematical models. The models are usually based on mathematical relation between three factors:

- **Grade of service**
- **Traffic interest**
- **Equipment requirement**

Traffic Intensity

The international unit of traffic is the Erlang, named after Danish mathematician Agner Krarup Erlang, who laid the foundation of the traffic theory. An Erlang is expressed as l/m where l is the average calls arrival rate and m is the average call service rate. From a practical standpoint, the Erlang is a measure of traffic intensity, where one Erlang represents one circuit occupied for one hour.

Traffic intensity can be defined in three ways:

- **The ratio between the period of time a switching device is occupied to the total period of time For example, if a trunk circuit is engaged on different calls for a total time of 25 minutes in an hour, the traffic intensity is determined as 0.4 Erlangs.**
- **The number of simultaneously occupied devices in a group of switching devices For example, five trunk circuits in a route are engaged at any given time. The traffic intensity is thus 5 Erlangs.**
- **The product of the number of calls per time unit and the mean holding time of calls (A = y * s) For example, in a local exchange, the total number of calls during 1 hour was 1800. The average holding time was 3 minutes. Therefore, the traffic intensity will be calculated as 1800 * 3/60 = 90 Erlangs.**

Traffic Concepts

There are three additional traffic concepts that affect the traffic intensity.

- **Traffic carried/traffic volume: The total traffic carried during a certain time is called traffic volume. Mathematically, traffic volume is an integral part of the traffic curve.**
- **Traffic offered: The concept of traffic offered is a theoretical quantity calculated on the basis of the number of circuits that are used. It is the estimated potential of the network.**
- **Traffic lost: The traffic lost is also a theoretical quantity calculated from the difference between traffic offered and traffic carried.**

Traffic Volume

The traffic volume keeps fluctuating during the day. Traffic volume is dependent on the subscribers' habit of using telephones. When dimensioning a network, you must consider the following factors:

- **Busy hour**
- **Occupation time**
- **Unsuccessful call attempts**
- **Grade of service**
- **Subscriber reactions**

Traffic Metering

Modern exchange equipment normally includes functions for traffic metering. These make it possible to collect data on the traffic in the exchange and store it on tapes for subsequent processing. It is also possible to collect data from many different exchanges in special operation centers. The measurements may be taken at set various factors concerning a network, such as call rate. You need to think about the media through which all the information travels and make a cost/benefit analysis of various media.

Formulae used in Traffic Engineering

There are two main formulae used in traffic engineering:

Erlang B and Poisson. The Erlang B formula assumes that all blocked calls are cleared. This means that the subscribers do not redial when they cannot get through. The Poisson formula assumes that the subscriber simply keeps on redialing. If the Poisson method is used, you will buy more trunks than if you use Erlang B. Poisson typically overestimates the number of trunks you will need while Erlang B underestimates the number of trunks that you will need. There are other more complex but more accurate ways of estimating the number of trunks needed. Erlang C and computer simulation can be used for this.

Introduction - Transmission

You need to design a transmission system for quality transmission and efficiency of operation. Once factors such as network and charging are determined, you need to determine the media through which information will travel.

Transmission systems include Transmission techniques and Transmission media.

Need for Transmission Techniques

The binary code generated for each sample by the PCM encoder is not suitable for transmission. Therefore, a suitable transmission technique has to be chosen to transmit the signal.

This is because the binary output from a PCM system normally has a wide range of bit patterns, including strings of 1s and even longer strings of 0s. However, for effective regeneration of the signal, the signal should have a frequency of 1-to-0 and 0-to 1 transitions.

Also, the transmission of the PCM binary signal, which is made up of zero and positive voltages, creates a line signal with a significant amount of energy in the DC and low frequency range. The transformers on the line system cannot handle this. There may also be other problems associated with the transmission of the low frequency power spectrum of binary signals, such as interference with audio signals on adjacent cables.

The most common line codes are HDB3, 4B3T, ADI/AMI, and CMI.

AMI (Alternate Mark Inversion) Technique

AMI is a transmission technique in which successive 1's are represented with opposite polarity. The efficiency of this code conversion is given by the ratio of the information carried to the capacity of the line code. AMI has a code efficiency of 67 percent

Transmission Systems: Multiplexers

Multiplexers, Optical Systems, and Wireless systems are some of the Transmission Systems.

A single conversation does not utilize bandwidth to the maximum extent. To utilize the bandwidth to the full extent, you can combine many such conversations to transmit through a same line by the process of Multiplexing. The process of conveying a number of signals simultaneously over a single bearer is termed as multiplexing. FDM systems can be used on different types of media such as symmetrical paired cables, coaxial cables, radio links, and satellite links.

Multiplexing can be done by two techniques:

- **Analog - Frequency division multiplexing (FDM)**
- **Digital - Time division multiplexing (TDM)**

Multiplexing in a Frequency Division Multiplexing (FDM) system: FDM systems can be used on different types of media including symmetrical paired cables, coaxial cables, radio links, and satellite links.

A carrier frequency is modulated by the signal from a speech channel. The result is a number of frequency bands. Only one of these frequency bands is needed. The others are filtered out.

Not even the carrier wave needs to be transmitted. It can be retrieved in the receiver as long as the frequency and phase are exactly correct.

FDM was frequently used in the past, but now has been replaced with the digital equivalent Time Division Multiplexing (TDM).

Multiplexing in a TDM system: In networks where high capacity is needed, it is more economical to group a number of PCM systems using higher-order multiplexing. This is done by bit interleaving. The bit rates at different multiplexing levels in a 30/32 channel system are shown below:

- **The idea is to combine 'n' input signals (from each of the 'n' channels) into a single multiplexed channel. The 'n' individual signals are separated and processed at the distant end and fed to the appropriate output channels. A time-division-multiplexed system has a common highway shared by a number of channels, each of which occupies the highway during periodic slices of time, known as 'time slots' (TS). This multiplexing technique requires that sampling be done to the input signal first. A number of such time slots are then combined to form a frame. Each of these time slots carries information from the corresponding channel.**

Transmission Media

A suitable medium is required to transmit speech signals regardless of the technique you use.
Transmission media can be differentiated into two main groups:

- **Wired media**
- **Wireless media**

Introduction - Call Charging

The set up and operation of a telecommunication network involves high investment. The capital investment includes land, buildings, switching systems, and cables. The operational cost includes salaries of staff, and costs incurred in the maintenance of the network. The telecommunication service receives its income from its subscribers. It is therefore necessary to design a suitable charging plan to recover the costs from subscribers.

Charging Plan

The charging plan should be designed according to a pre-defined policy. If telephone services are considered a social right and are to be made available completely free, no charging plan is needed at all. However, as soon as you decide to take some payment, a charging plan is needed.

Before you design the charging plan, you should have a policy that answers the following questions:

- **What portion of the costs should the subscriber pay?**
- **Should charging rates depend on the category of the subscriber?**
- **Should the rates vary with geographic location?**
- **Should the company make a profit?**

Types of charges in a charging plan

The charges levied on the subscriber can be categorized in various ways.

- **Initial charge for providing a network connection**
- **Rental charge**
- **Charges for individual calls**

There are usually three types of charges in the charging plan.

- **Installation charge: When the administration connects new subscribers, some initial costs are incurred such as the equipment at the exchange that has to be assigned to the subscriber, a telephone set, and the cabling. These charges are often the same for all installations and are independent of the actual cost of an installation.**
- **Subscription fee: This is a fixed cost charged to the subscriber for the maintenance of the network, and services such as providing the telephone directory. The subscription fee is usually charged on a periodic basis, such as monthly.**
- **Call charges: Call charges are generally in direct proportion to the number of calls and the duration of these calls. Call charges also vary depending on whether the call is a local call or a trunk call. Calls may be charged differently based on the time of the day the call is made. In some cases, there are no call charges for local calls. Instead, the subscription fee is significantly higher.**

Types of Charging Plans

There are different types of charging plans depending upon whether you decide to charge according to the number of calls, the duration of calls, or charge a fixed amount for a particular time period.

- **Flat rate: The simplest possible form of charging is the flat rate charge. In this charging plan, call charges are fixed for a period of time. Whether you make 5 or 25 calls a day, the charge is the same. One of the advantages of the model is to stimulate the use of existing installations. If the company decides to opt for this model, the charges for installation are usually quite high. To avoid having to lay down new cables and other costs associated with catering to new subscribers, the**

company may decide to charge a flat rate. Many consumers will then use the existing phones already installed at various places.

- **Unit fee charging: Call charges depend only on the number of calls, not on other factors such as the duration of the call or the distance involved. The exchange counts each subscriber's calls. Every call costs a fixed amount regardless of where the subscriber rings or how long the call lasts. The cost per call is always the same.**
- **Time charging: Under Time charging, the call is charged based on the duration of each call. Thus, call charges depend upon the amount of time the subscriber kept the exchange busy. The greater the distance between the called and calling subscriber, the more the equipment involved, and the more the cost of the call. For example, though Mumbai-Pune is a local call, these calls are time charged at a higher rate than a normal local call within the city.**

Some countries have formulated different charges for different times of the day, thus making the charging entirely dependent on the time of the day.

Charging Methods

Calls are charged by using either a metering instrument connected to each subscriber line or a metering register assigned to each subscriber in case of electronic exchanges. The count in the meter or register represents the number of charging units. The subscriber is billed by multiplying the number of charging units by the charging rate. The count in the meter is incremented by sending a pulse to the meter. Charging methods may be duration dependent or duration independent.

The two charging methods prevalent are:

- **Periodic Pulse Metering: Periodic Pulse Metering is based on periodically generated pulses from a centrally situated pulse generator. The charge for the call is proportional to its duration.**
- **Itemized Billing: For every call, the equipment at the exchange records the numbers calling and called, together with the duration, date, time, type of service, and charging rate. This information is used to charge subscribers according to the charging plan.**

Process of charging calls

A charging point is the exchange where a call is recorded. In principle, the charging points should be as close to the calling subscriber as possible. For local calls, the charging point is the exchange. When the call is within the same tariff zone, the call is recorded within the local exchange. When the call is going to an exchange in another tariff zone, a more refined analysis of the relevant charge has to be carried out. Hence, one exchange in the zone becomes the new charging point. As the recording has to be sent to the local exchange, charging pulses must be sent to the local exchange.

Glossary

Analog:

An analog signal is a continuously varying electromagnetic wave that can propagate through a variety of media.

Bit interleaving:

This is a form of Time Division Multiplexing (TDM) for synchronous protocols such as HDLC. Bit interleaving retains the sequence and number of bits to achieve correct synchronization between both ends.

Called subscriber:

In a telephone conversation, the person who receives the call is known as the called subscriber.

Calling subscriber:

In a telephone conversation, the person who initiates the call is known as the calling subscriber.

CCITT:

Consultative Committee on International Telegraph and Telephone. This is the principal international standards-writing body for digital telecom networks (ISDN).

Clear:

The clear signal is sent to the exchange after the subscriber hangs up.

Digital:

The conversion of voice or data into a stream of binary information (0s and 1s).

DTMF (Dual Tone Multi-Frequency):

A term used for push button or touchtone (an AT&T trademark) dialing. The pushed button generates a tone, actually the combination of two tones, one of high and the other of low frequency. They are necessary to access advanced network features such as call barring and call forwarding.

Hop distance:

Distance between 2 boosters or repeaters.

Multiplexing:

A technique that allows many signals to be sent along the same transmission path.

Numbering Area:

The area where two subscribers use identical dialing procedure to reach any other subscriber in the network.

Off-Hook:

A telephone set in use - the handset is removed from its cradle, thus sending an electric signal to the exchange that a circuit needs to be opened.

On-Hook:

The normal state of the telephone in which the handset rests on the cradle and the circuit to the exchange conducts no electric signal.

Quantum Values:

Discrete finite digital values corresponding to the specific instance of analog signal.

Seize:

The seize signal is sent to the subscriber while the exchange is checking for resources required for call setup.

Signal:

A signal is a sequence of electronic pulses.

Signaling:

Signaling is the transmission of information by means other than the unaided human voice. Signals, which can be visual, audible, or electric, can be made by a variety of means, including lighted torches, smoke, flags, lamps, drums, guns, telegraph, telephone, and radio. Signals are used to control railroad trains and vehicular traffic on roads and highways and to communicate with ships, airplanes, and other vehicles. Transmission of electric signals over great distances is known as telecommunication.

Switching System:

A switching system is used to establish a path between a set of input and output circuits.

Tele-traffic Theory:

It is a branch of applied probability. The Tele-traffic theory is the mathematical description of message flow in a communications network.

Trunk:

It is the line of communication between switching systems.

WAP (Wireless Application Protocol):

It is a technology that allows your mobile phone to browse the Web. It is a protocol for data transmission over low bandwidth wireless networks.

PL/I

INTRODUCTION

PL/I was developed by IBM in the mid 1960's and was originally named as NPL(New Programming Language)

It was first introduced in 1964. The name was changed to PL/I to avoid confusion of NPL with National Physical Laboratory in England.

Previous languages had focussed on one particular area of application, such as Science or Business. PL/I was not designed as to be used in the same way. It was the first large scale attempt to design a language that could be used in a variety of application areas.

PL/I is used significantly in both Business and Science applications.
Marathon Oil Company, Ford Motor Company, General Motors are some of the clients.
Unlike many other languages PL/I is completely free-form.
No reserved keywords I.e PL/I determines the meaning of the keywords from the context of usage.
E.g. : It is perfectly valid to declare a variable AREA even though it is also a PL/I keyword.

FORMAT OF A PL/1 STATEMENT

1 :- Column 2 to 72 are used for coding PL/1 statements. each statement should end with a semicolon.

2 :- First statement of a program should be PROCEDURE or PROC. each procedure statement must begin with a label, 1 to 31 chars.

3 :- The first character of a label should always be an alphabet and the label should be followed by a colon.

4 :- Last statement of a procedure always ends with END. END ends the program and passes the control to the operating system.

DECLARE Statement

DCL is the statement after proc which means DECLARE and is used to reserve storage for data or variables indicating the attributes of a variable.
In PL/1 there are 2 basic data types :-
1. Arithmetic data and
2. String data (char strings)

DCL Arithmetic data items

Form :- DCL variable_name mode scale base precision
Mode of an Arithmetic item is either REAL or COMPLEX. default is REAL.
Scale is either FIXED or FLOAT.
Base is either DECIMAL or BINARY.
Precision means number of figures that a fixed point data item can hold or in case of floating

point, the minimum number of valid digits to be regarded as significant.

DCL Decimal data type

e.g. :- 102.30   Decimal Fixed Point
        125.3E5 Decimal Floating Point

A floating point number consists of 2 parts fraction and exponent.
e.g. :- 168325E20 = 168325 * (10**20)

DCL Binary data items

Binary numbers consists of 1 and 0, to indicate that a constant is a binary, it is followed by a "B".
e.g. :- 101110010B [precision is (9,0)]

A binary fixed point variable is declared as
DCL mbinnum fixed bin(16,0);
The maximum number of digits in a binary number is 31. The default for fixed binary variable is 15,0.

DCL String data items

String data can be either character or bit string.
PL/1 recognises string data by the quotation mark which must appear at the start and finish of each string.
e.g. :- 'the world'

To use quotation in character string, code two of them.
e.g. :- don't as a char string is represented by 'don''t'

To repeat a character string use (n)'char string'
e.g. :- (80)'RAJ' (RAJ will be repeated 80 times.)

Declaration of character string variable is done as DCL name CHAR(15)
To vary not only the context but also the length of a character variable use the attribute VARYING or VAR.
e.g. :- DCL addr CHAR(33) VARYING;

The length can vary from 1 to 33 and no padding of spaces is done, if the length is less than 33.

BIT strings

Bit strings are like char strings except that the digits represent one bit (0 and 1). Bit strings must be enclosed in quotation marks.
Bit strings may too have varying attributes.

```
e.g. of bit string constants '1011101001'B
                             '111'B
                             (24)'0'B
e.g. of bit string variables DCL switch BIT(1),
                             DCL bitstr BIT(320) VARYING;
```

Explanation of some symbols used in Pl/1 program

Name	Symbol	- Explenation
COLON	:	- follows a label
COMMA	,	- separates list items
SEMICOLON	;	- ends a statement
PARENTHESES	( )	- encloses list items
ASTERISK	*	- multiplication

Difference between a binary number & a bit string

A binary number has a value and is treated like a number.
A bit string has no value and is treated like a string of bits.

DEFINED or DEF attribute

It is useful to refer to same pieces of string data (bit or char) by more than one name. it can be used to overlay or sub-divide areas of string data.

```
e.g. :- DCL idcode CHAR(8);
        DCL area CHAR(3) DEF idcode;
        idcode='A02B1506';
        PUT LIST(area);
```

Base variable idcode occupies 8 bytes of storage.
Variable area defined over it starts at the same address but it is only 3 bytes long.
Put writes out the first 3 characters of idcode 'A02'.

POSITION or POS attribute

With POS, you can make the defined variable start at any position of the base variable.

```
e.g :- DCL idcode CHAR(8),
       part1 CHAR(3) DEF idcode,
       part2 CHAR(3) DEF idcode POS(4),
       part3 CHAR(2) DEF idcode POS(7);
```

part1 overlays the first 3 characters of idcode;
part2 is over characters 4,5,6;
and part3 is over characters 7,8.

```
e.g :- DCL whole BIT(10),
       select BIT(3) DEF whole POS(5);
       WHOLE='1000111010'B;
```

contents of select are '111'B

PICTURE attribute & some special characters

Variables which are to hold both types of data must be declared using picture attribute or pic.
9 indicates numeric,
V indicates decimal point.
Picture edit characters must be enclosed in quotation marks.
Minus and Decimal point use storage spaces.
If the number is non-negative, it is replaced by a blank.

```
e.g :- DCL number PIC 'S999V.99'
           number = 123.45;
```

in storage, the number look like +123.45
To edit character strings, you may use 3 edit characters :-

1:- X Position may hold any character.
2:- A Position may hold alphabetic character or a blank.
3:- 9 Indicates that position may hold numeric characters or space.

```
e.g :- DCL fielda pic '999X99';
           fielda = '123.45';
```

INITIAL or INIT attribute

To assign value to a variable at the start of the program execution use the attribute initial or init.

```
e.g :- DCL switch BIT 1 INIT('1'B);
       DCL whole FIXED DEC(5) INIT(0);
       DCL title CHAR(13) INIT('Weekly News');
```

Examples of some constants

'101110'B - bit string

.25638E20	- decimal floating point number
'34862'	- character string
101101B	- binary fixed point number
10387	- decimal fixed point number

Expressions

IF, THEN , ELSE statement

Simplest if then else statement :-

```
form :- IF expression THEN then-unit-expression;
                      ELSE else-unit-expression;
```

```
e.g. :- if a=0 then counter = counter + 1;
```

```
e.g. :- if A <= B then A=B;
                   else A=B+100;
e.g. :- DCL switch BIT(1);
            switch = A;
            if switch then goto sub1;
```

DO in an IF statement indicates start of a DO group block.
END in an IF statement indicates end of a DO group block.
e.g. :-

```
x, y = 1;
if a > b then do;
                  x = 100;
                  y = 50;
             end;
```

e.g. :-

```
if a > b then     c = 100;
         else do;
                  c = z + 10;
                  x = y;
         end;
```

If character strings of different lengths are compared then the shorter one is padded on the right with blank spaces.
In case of bit strings; if length differs, the shorter string is padded to the right with '0'B.

NULL statement :- null statement consists of a semicolon. it is used where logically no statement is necessary.
e.g. :-

```
if x > y then if m = n    then;
                                          else z = 0;
           else  z = z + 1;
```

OPERATORS

comparison operators are as follows :-

<	less than
>	greater than
=	equal to
<=	less than or equal to
¬<	not less than
¬>	not greater than
¬=	not equal to
>=	greater than or equal to

Bit string operators are as follows :-

NOT	¬ alt-170
AND	&
OR	|

Concatenation character is ||

DO groups

Iterative do group used in an IF statement is also used to put several PL/1 statements together, but in this type of do group these statements are as a rule executed several times over a group.
e.g. :-

```
do    i = 1 to 10;
      ........
      ........
end;
```

The do group will run through 10 times.

BY statement:-

If the control variable is to be increased by a step other than 1, then you use by statement.
e.g. :- do count = 1 to 20 by 2;

in do groups you can also count downwards.
e.g. :- do i = 25 to 1 by -1;

in addition to constants; variables and expressions can also be written in specification.
e.g. :- do alpha = x to beta * 2 by delta;

Multiple Specification :-

multiple specifications can be used in a do statement. control passes to the next specification when previous has finished.
e.g. :- do i = 1 to 5, 20 to 25, 50;
This loop is executed 5 times for i = 1 to 5,
then 6 times for i = 20 to 25,
and then again with i = 50.
Hence the loop is executed totally 12 times.
Often do groups have to be executed until a certain condition is met. these cases can be handled with while statement.
e.g. :- do while (x < y);
This loop is executed as long as x is less than y.
e.g. :-

```
do while (x > y) until (z = 100);
...........  .....
endo;
```

This loop runs as long as x > y and z is not equal to 100.

Nested DO groups :-

Nesting of do groups to a maximum of 49 is allowed by PL/1 optimizing compiler.
e.g. :-

```
outer : do i =  1 to 5;
        inner : do j = 1 to 3;
                .....
        end     inner;
end     outer;
```

e.g. :- To leave an iterative do group.

```
do    i = 1 to 100;
      .....
      if x > y then leave;
      .....
end;
```

GOTO statement

With goto statement you can jump to any statement, which has a label, except an iterative do group.
e.g. :-

```
   i = 1;
a: if i > 10 then goto b;
   ......
   i = i + 1;
   goto a;
b: ......
```

Data Aggregates & Arrays

Data Aggregates

Two types of data aggregates are there :-

- arrays

- structures

An ARRAY is a collection of several elements with same attributes. They are arranged in a table or a matrix.
When you would like to put unequal data elements together in an aggregate, you should use a STRUCTURE.

Data Aggregates (Arrays)

e.g. :- dcl table(50) fixed decimal(3); Table is a one dimensional array of 50 elements. each element is a decimal fixed point with the precision 3,0. table is to be regarded as a string of 50 elements.
If you want to address a single array element.
e.g. :-

```
dcl countytab(50) char(10);
dcl countynm char(10);
```

countynm = countytab(2);

Second element of countytab is assigned to countynm.
e.g. :-
DCL numbers(5) fixed dec(3);

```
 do    i = 1 to 5;
       numbers(i) = i * 2;
 end;
```

Subscripts need not just consist of constants and variables, they can also be whole expressions.
e.g. :-
cost(i+j)
numbers(i+2)
numbers(costs(i))
costs(i+j*2)

Built-in functions :-

A built-in function replaces a number of statements which the programmer would otherwise have to write himself.
For e.g. function ALWAYS return one value and is recognised by PL/1 thru syntax.
e.g. :-
dcl (a,b) fixed dec(9);
dcl x(10) fixed dec(3);
dcl y(12) fixed dec(7);
a = b + sum(x) + sum(y);

Multi-Dimensional Array :-

The dimension attribute states the number of the dimensions for an array and the bounds of each dimension.
e.g. :- dcl table(7,2) fixed decimal(3);
The above mentioned array consists of 7 rows and 2 columns.
e.g. :- dcl parts(3,3,5) char(10);
The above declared parts array consist of 45 elements.
e.g. :-
dcl tab(3,5) fixed binary(15);

```
 do    j = 1 to 3;
       do k = 1 to 5;
       tab(j,k) = 2;
       end;
 end;
```

x = sum(tab);
Value of 'x' after execution is 30 because each of the 15 elements in this array is set to 2.
All element of array tab can be set to 2 by one single instruction. [tab = 2].

Cross section of an array :-

you can also address a cross section of a array.
e.g. :-

```
dcl figures(5,3) fixed dec;
dcl x(3);
x = figures(2,*);
```

in the above case * represents all the elements of row 2.
hence x = figures(2,*) is equal to

```
x(1) = figures(2,1);
x(2) = figures(2,2);
x(3) = figures(2,3);
```

Data Aggregates (Structures)

Simple Structures :-

When you would like to put unequal data elements together in an aggregate, you should use a structure.
e.g. :-

```
dcl 1 personnel,
      2 name char(30),
      2 salary fixed decimal(7,2);
```

Name of the whole structure is personnel. Each element in it, has it's own name and it's own attribute.

During processing elements can be addressed singly or all together using the structure name. A single element can be addressed by it's name.

for e.g. :-

```
name = 'Siva Mani';
put file(out) list(personnel);
put file(out) list(name, salary);
```

The above two statements have the same meaning. write to the file out the values of name and salary. Maximum number of levels in a structure is 15.

Arrays in a Structure & Qualified Names :-

```
dcl 1 month(12),
      2 max fixed dec(7,2),
      2 min fixed dec(5,2);
```

There are 12 occurence of month and each occurence has 2 elements.

If you want to refer the element min in 10th structure.
month(10).min (OR) month.min(10)
Qualified name is month(10).min (OR) month.min(10)

BY NAME option in a structure

There is one way by which you can use non-identical structures in structure expression. that is by name option.
By name is used to make assignments from one structure to another, whereby compiler processes only the elements whose fully qualified names in both structures are identical.
e.g. :-

```
                                     dcl 1 old,
   dcl 1 new,                              2   name,
         2    name,                        2   address,
         2    payment,                     2   payment,
1             3        gross,       2          3         gross,
              3        net,                    3         net,
              3        deductions;         2   deductions;
```

There is an assignment for all the elements which have the same qualified names in the structures new and old.

new = old, by name;

3 elements are assigned in this structure expression.

new.name = old.name;
new.payment.gross = old.payment.gross;
new.payment.net = old.payment.net;

Introduction to Input/Output

Basic features of Input/Output

GET :-

Transmits or gets data from external medium to internal storage.

PUT :-

Transmits data from internal to external storage.

2 types of transmission are there between internal & external storage :-

stream i/o with get & put :-

Data item is transmitted singly. advisable for small quantities i.e. screen & card readers.

record input/output with read & write :-

Complete record of data items is transmitted and copy of the record is usually kept in internal storage. It is preferable for larger qty of data.

Basic features of Input/Output

Syntax of 2 i/o statements :-

```
stream -> get file(input) list(a,b,c);
record -> read file(input) into(inarea);
```

opening & closing of files :-

```
 e.g. :- dcl master file record sequential;
         ..........
         open file(master) output;
         ..........
         close file(master);
```

Environment attribute

Environment attribute defines the physical organisation of data sets (e.g. record length, blocks) and the method of processing them.
Default value for the type of organization is consecutive. this type of organization can be processed only sequentially.

```
e.g. :- dcl purchase record sequential input env(consecutive);
        open file(purchase);
```

Record is an attribute in the declaration. Stream is default for the transmission type.

STREAM Input/Output

STREAM I/O Features

A string of characters is transmitted in stream i/o.
Character data type will not cause conversion if it is used in get or put.

Standard files for stream i/o :-

the get and the put statements not having data file declarations are linked with standard files for which there are prescribed file names.
SYSIN for input, SYSPRINT for output

```
hence        PUT LIST(DATA LIST);
is taken as  PUT FILE(SYSPRINT) LIST(DATA LIST);
```

3 Types of I/O Stream

list	- always requires data list
data	- requires no list
edit	- requires data and format list

List-Directed I/O :-

It is simplest form of stream I/O, and is ideal for testing input at the terminal.

```
e.g. :- get list(c1,c2,c3);
        put list(gross,net,gross-net);
e.g. :- dcl temps(7,2) fixed decimal(3);
        .....
        get list(temps);
```

14 elements would be read by the get statement.
Input in list directed i/o is in character form. The data items are separated by comma and/or blanks.

Data-Directed I/O :-

Input data can be read in the form of assignment statements in input stream. a semi colon or the end of data file ends the transmission of an i/o statement.

```
e.g. :- A=2, B='XY'; ------> get data(a,b);
        A=5;         ------> get data(a,b);
        B='UV';      ------> get data(a,b);
```

in first read 'A' is numerical and 'B' is character.
on second get 'A' = 5 and 'B' is unchanged.
on third get 'A' is unchanged and 'B' is 'UV'.

Edit-Directed I/O :-

It is the most commonly used type of stream i/o. a data list is needed here as well, to contain the names of data items, arrays or structures. in addition a format list is required in which the format of i/o data is described.

```
e.g. :- put file(output) edit(data list) (format);
        get file(sysin) edit(data list) (format);
```

e.g:-
get edit(qty, price, text) (f(3),x(6),f(5,2),x(5),a(10));

F is used for numeric data item, A for character string, X for space.

PAGE & SKIP in EDIT Directed I/O

For preparation of numerical data the format item P can be used with all picture specification.
e.g. :-

```
dcl no fixed dec(5),
 order fixed dec(7,2),
 text char(20);
put PAGE edit('Revision List') (A);
put SKIP(2) edit(no,order,text)
(f(5), x(2), P'(4)Z9V.99-', x(2), a);
```

After advancing to next page, a title is printed. then two lines are skipped and the three variables are printed with 2 spaces in between.
result is fffff ppppp.pp aaaaaaaaaaaaaaaaaaaa

LINESIZE & PAGESIZE in EDIT-Directed I/O

print is a file attribute which can be used in connection with stream output.
Some options which can be stated in open statement are :-

option	meaning	default
linesize(nn)	line length	120 chars per line
pagesize(nn)	page length	60 lines per page

e.g. :- open file(list) output stream print linesize(100) pagesize(50);

ON ERROR condition

PUT statements without data list, write out all the variables of a program block. they give programmers an excellent tool to use in searching for tools.

```
e.g. :- on error begin;
                 on error system;
                 put data;
                 goto exit;
           end;
```

The on error condition must also have been run thru atleast once before it can be executed.

On error system means that in case of an error, the system routine is executed that includes breaking off the program execution.

Record Input/Output

Overview & Differences between STREAM & RECORD I/O

RECORD I/O	STREAM I/O
The data in a data set is considered as a collection of record.	Data is considered as a continuous sequence.
Transmits records just as they are stored.	Data conversion is possible.
There are no standard files.	SYSIN and SYSPRINT are valid standard files
Allows various forms of data set organization.	Allows only sequential form of data set.

RECORD I/O Features

RECORD I/O offers 2 alternative modes of processing (move / locate).
<u>MOVE mode</u> :- In move mode, a logical record is transmitted by reading from the buffer to an area (e.g. structure) where it is then processed.
<u>FORM</u> :- read file(filename) into(area);
using read, a logical record is put into a variable declared in the program.

```
 e.g.:-     dcl   1 inrec,
                  2 part1 char(25),
                  2 part2 (10) decimal fixed(7,2);



            read file(material) into(inrec);
```

<u>IGNORE option</u> :- ignore specifies how many logical records are to be skipped.
<u>form</u> :- read file(filename) ignore(N);

'N' is the number of records to be skipped.

<u>Record output in MOVE mode</u>:- On each write, a logical record is transmitted from the area to a buffer. if record (or block) is full, a physical record (for block) is written into the data set.
<u>form</u> :- write file(filename) from(area);

write statement causes a record to be written from the area specified i.e. the records are put in blocks.
<u>declaration and opening of file</u> :-

```
 e.g. :- dcl pers file input sequential buffered
                  env(consecutive FB RECSIZE(100)
                  BLKSIZE(500) BUFFERS(3));
```

FB	means fixed record length, blocked
RECSIZE	means length of the record in bytes
BLKSIZE	means length of the block in bytes
BUFFERS	means number of buffers

The record has 100 characters and block has 500 characters, hence there are 5 records per block.

Environment attribute describes the physical properties of a data set. consecutive means that records are stored sequentially.

LOCATE mode

In this mode the logical records are directly processed in buffer. here a based variable or a pointer variable is needed (a structure with the length of a logical record) is like a mask which is laid over the buffer.

<u>record input in locate mode</u> :- Files to be processed using locate mode must, like other files be declared and opened. the first physical record has already been read into the buffer and opened.

<u>using</u> :- read file(filename) set(pointer);

The address of the first record logical pointer is put in the variable pointer.

POINTER Variable

A pointer variable is a variable (full word, 4 bytes) which contains a storage address. this address can be assigned to pointer variable in various ways.

```
 e.g. :- dcl xyz pointer;
         read file(input) set(xyz);
```

A pointer is used to point to the position of the next record in a buffer or to another variable.

BASED Variable

In the e.g. given below, read places the address of next logical record in the pointer ptr.

```
 e.g. :- dcl 1 efile based(ptr),
                 2 name char(30),
                 2 first_name char(20),
                 2 persno fixed dec(5);

          Read file(pers) set(ptr);
```

The based variable efile is overlaid on the logical record, as it has the attribute based and contains address as the pointer.

<u>Example of Record read in locate mode</u>

```
    F   dcl one file input sequential buffered
  env(fb recsize(200) blksize(1000) buffers(1));
  dcl q pointer;
  dcl record_a char(200) based(q);
  .....
```

```
read file(one) set(q);
```

- On 6th read, the 2nd physical record is retrieved from the data set and put in buffer.

- you recognize MOVE mode by the option FROM and LOCATE mode by the option SET.

<u>Record Output in Locate mode Records read in are written out</u>

<u>form</u> :- LOCATE based_variable FILE(filename);

- e.g. :- dcl in file record buffered input

```
     env(fb recsize(100) blksize(1000)
     buffers(2));
          dcl out file record buffered output
     env(fb recsize(100) blksize(1000)
     buffers(2));
   dcl inrec  char(100) based(q);
   dcl outrec char(100) based(p);

   loop : read file(in) set(q);
          locate outrec file(out);
          outrec = inrec;
          goto loop;
```

<u>Rewriting of a file in MOVE and LOCATE mode</u>

With REWRITE statement, records read in can be changed and written back to the same place they came from in the data set.

- e.g. :- dcl personnel file sequential buffered

```
               update;
   dcl record char(500) based(p);
   dcl p pointer;

   read file(personnel) set(p);
   .......
   change record read in .....
   .......
   rewrite file(personnel);
```

Variable length record

- If quantity of data per logical record varies a great deal (as for e.g. with addresses) it is a good idea to store the data in variable length record.

F note :- There is 4 byte record length field (RL)before every variable record.
There is a 4 byte block length field before every block.

F e.g. :- suppose the biggest logical record can be 1000 bytes long and that the records are to
be blocked in such a way that a block is likewise to have the maximum length of 1000 bytes, then environment attribute of file declaration would look like
env(vb recsize(1004) blksize(1008));

<u>Size statements are calculated as</u> :-

recsize = 1000 bytes + 4 bytes record length
blksize = recsize + 4 bytes block length

F In fixed format, the record length and the block lengthare fixed but in variable format, the record length and the block length vary

block length is size of the physical record
record length is size of the logical record
If data length of the record is 750, the record sizewill be 754 and the block size will be 758.

<u>SCALAR VARYING option</u>

If you are using record i/o with variable length strings and locate mode, then scalar varying option must be stated in env attribute for i/o, so that the 2 byte length area which goes before each variable length string are taken into consideration.

F e.g. :-
```
dcl pers-file file record output buffered
  env(vb recsize(106) blksize(110)
      scalarvarying buffers(1));

dcl outrec char(100) varying;
```

Handling VSAM Files

VSAM data sets

3 Types of VSAM data sets are there :-

1. Key Sequenced Data Set (KSDS)
2. Entry Sequenced Data Set (ESDS)
3. Relative Record Data Set (RRDS)

<u>Note</u> :- Sequential is the default file organization, hence vsam should be specified in the ENV option.

Vsam files are always record files. Records on a vsam file are not held in blocks, hence the blksize option even if specified is ignored. A recsize option will be checked against the details stored with the file.

The utility IDCAMS (access method services) is used to set up a vsam file initially.
<u>Various DCL for VSAM datasets</u>
e.g. :- dcl vfile file record direct keyed input env(vsam);

Vsam files are all direct access files. associated with every record is a key, which helps in accessing the record in any order.

e.g. :- dcl vfile file record sequential input env(vsam);

If you want to start at the beginning and process every record, just as with a consecutive file, specify the file as sequential.

e.g. :- dcl vfile file record sequential keyed input env(vsam);

The first read of the program specifies the key of the record you want to start at. subsequent sequential read statements are issued without a key.

e.g. :- dcl infile file record direct keyed input env(vsam);
to read a vsam file directly.

e.g. :- dcl infile file record direct update env(vsam password('master'));

When a file is originally set, it can be given one or more passwords. if attribute record is given, it means that it must be a file, hence the attribute file is implied and therefore optional.

Key Sequenced Data Set (KSDS)

KSDS is used to hold master file data. identification is by means of key so the attribute DIRECT implies KEYED.
e.g.:- dcl amast file record direct input env(vsam);
Sequential access is possible with ksds.

A ksds has a key which is part of a data record, this is known as an embedded key.
Keys are also stored in another part of data set called as prime index.
The keys are unique since duplication is not allowed.
<u>KSDS Features</u> **<u>Reading a ksds directly</u> :-** Records are accessed by specifying a value using the key option.

```
e.g.:- dcl amast file record direct input env(vsam);
       dcl 1 area,
             2 kskey char(5),
             2 ksname char(15);
```

```
    read file(amast) into(area) key('00010');
```

Updating a ksds directly :- The file should be opened as update. you cannot amend the key.

```
e.g.: dcl kmast file record direct update env(vsam);
   dcl 1 area,
      2 kskey char(5) init('00081'),
      2 ksname char(15);
   read file(kmast) into(area) key(kskey);
   /* amend */
   rewrite file(kmast) from(area) key(kskey);
```

Delete a record in ksds :- :- :- Use delete statement.

```
e.g. :- kskey = '00056';
    delete file(ksds) key(kskey);
```

Add a record in ksds :- :- :- Use write statement.

```
e.g. :- kskey = '00024'; ksname = 'Applets';
    write file(ksds) from(area) keyfrom(kskey);
```

F **First direct access & then sequential access :-**

```
e.g.: dcl ramp file record sequential keyed
          env(vsam);
    read file(ramp) into(area) key('123/999/b');
    read file(ramp) into(area);
```

GENKEY option

F e.g. :- dcl vsam file record sequential keyed

```
          env(vsam genkey);
    read file(vmast) into(area) key('123');
    read file(vmast) into(area);
```

Assume that the key is defined as nine chars long.

If genkey option is not specified in the environment statement, vsam searches for a record with a key of ‘123‘.

If genkey option is specified the first read reads the first record of data set beginning with 123.

Second read continues with the record immediately following it.

ENTRY Sequenced Data Set (ESDS)

It is the most likely consecutive non-vsam file. i.e. data is not in a particular order, hence the file is sequential.
An esds file which already has data on it, may only be opened with either input or update attributes.
e.g. :- dcl esds file record sequential update env(vsam);

<u>Updating an esds file </u>:-
It is possible to change records once they have been written to esds file.

```
e.g. :- read file(esds) into(area);
        ......
        rewrite file(esds) from(area);
```

Write can also be used and the new records added are always added at the end.

Delete is not allowed.

Keyed access to an ESDS file

An esds is a sequential file and has no defined key.
The information that can be used for keyed access is called relative byte address (rba).

```
record-1  : record-2  : record-3  : record-4
          :           :           :
50 bytes  : 40 bytes  : 60 bytes  : 30 bytes
          :           :           :
rba=0  rba=50      rba=90      rba=150
```

You have to get hold of rba by using KEYTO option on read and write.

```
     F    e.g. :- dcl esds file record sequential keyed update
                  env(vsam);
        dcl rba char(4);
        ......
        write file(esds) from(area) keyto(rba);
```

Rba must be a 4 byte character string. this will add a new record to the end of the file and place the relative byte address of the record in the field rba. this is then used as a key on a read or rewrite. file should be declared as keyed.

e.g. :- read file(esds) into(area) key(rba);

F e.g. :- /* example of esds read and rewrite */

```
dcl esds file record sequential keyed update
         env(vsam);
dcl rba char(4);
read file(esds) into(area) keyto(rba);
. . . . . .
rewrite file(esds) from(area) key(rba);
```

RELATIVE Record Data Set (RRDS)

Records are read and can be accessed sequentially or directly with or without keys.
The data set may be thought of as a set of numbered slots each of which may hold record. the numbering starts at 1 and goes to the maximum specified.
Record number is the relative record number. this is the key to the file.
The key must be declared as a character string of length 8.

F e.g. :- dcl relrecno char(8) init('00000145');

F e.g. :- dcl relrecno char(8) init('   145');

<u>Keyed access to an RRDS</u>
File may be declared as sequential or direct.
If the file is declared as keyed sequential all key options can be used. if you leave key reference, processing would be just for a sequential file.
e.g:-

```
dcl rrds file record sequential keyed update env(vsam);
dcl rrkey char(8);
rrkey = ('00000025');
read file(rrds) into(area) key(rrkey); /* read rel.rec.no 25 */
......
rewrite file(rrds) from(area);     /* rec.no 25 is rewritten */
......
read file(rrds) into(area) keyto(rrkey); /* requests a */
                                          /* record number. */
                                        /* the next record will */
                                        /* be presented and the */
                                        /* key value will be */
                                        /* placed in rrkey */
```

<u>WRITE in an RRDS</u>

F e.g. :- /* assume record 5 exists on file rrds. */

```
                dcl rrds file record sequential keyed update
                 env(vsam);

        dcl relrecno char(8);
        dcl 1 area,
              2 ...... ;

        relrecno = '00000005';
        read file(rrds) into(area) key(relrecno);
        ......
        delete file(rrds) key(relrecno);
        ......
        write file(rrds) from(area)
                         keyfrom(relrecno);
```

Processing of String Data

SUBSTR (Built-in Function)

Substr is used to access a part of a string.
<u>form</u> :- (string-var., start-pos., length-of-extract)

```
      F    e.g. :- dcl data char(50);
        dcl field char(10);
        ......
        data='INTERNATIONAL BUSINESS MACHINES';
        field=substr(data,6,8);
```

Field will be 'NATIONAL'
In SUBSTR parameter 2 and 3 can be expression, resulting to be an integer.
<u>note</u> :- if the parameters represent a value which lies outside the string the result is unpredictable.

INDEX (Built-in Function)

Index function is used to look for a certain character or a string of chars in a given string.

<u>form</u> :- pos. = index(string, search string);
Index function returns a binary number (15,0). If string to be looked for is not found, the return value is zero. only first occurence is shown.

```
      F    e.g. :- dcl town_country char(25);
        ......
        town_country = 'WEYBRIDGE,SURVEY';
        icomma=index(town_country,',');
```

icomma will be 10.

```
e.g. :- dcl (string,part1) char(30);
    ......
    ipos  = index(string,'/');
    part1 = substr(string,1,ipos-1);
```

VERIFY (Built-in Function)

<u>VERIFY function</u> :- Verify will see that a string will have only certain characters present.
<u>form</u> :- verify(x, y);

'X' is the string of chars to be checked in Y.
'Y' is the string containing the list of characters which are to be looked for.

<u>A binary fixed point number is returned</u> :-
- zero, if all characters in 'X' appear also in 'Y'.
- A value, which gives the position of the first character in 'X' which does not appear in 'Y'.

```
F   e.g. :- dcl test char(3);
    test='BIT';
    IX=verify(test,'ABCDEDFGHI');
```

3 would be returned, because the third character is not in the second parameter of verify.

TRANSLATE (Built-in Function)

<u>form</u> :- translate(x, y, z)

'X' = character string to be checked
'Y' = string of replacement character
'Z' = chars in 'X' which are to be replaced if found

result is modified character string 'X'.

```
F   e.g. :- dcl (old, new) char(10);
    dcl icharacter char(10) init('FIRTUX');
    dcl tcharacter char(10) init('ANUIQT');
    ......
    old = 'FIXTURES';
    new = translate(old,tcharacter,icharacter);

F   new = translate('FIXTURES','ANUIQT','FIRTUX'); = ANTIQUES
```

LENGTH (Built-in Function)

This determines the length of the string. value returned is binary fixed point number (15,0).

<u>form</u> :- length(x);

- e.g. :- dcl field char(90) varying;
  field = 'GOOD LUCK !';
  ilength = length(field);

- ilength receives the value 11.

Pseudovariables

If a function is on the left in an assignment, you have a pseudovariable. pseudovariables are functions which may not only manipulate values, but may also receive them.

- e.g. :- substr(tfield,2,5) = gfield;

- e.g. :- dcl x char(10);
  x = 'ABCDEFGHIJ';
  substr(x,6,5) = substr(x,1,5);

  /* soln is 'ABCDEABCDE' */

- **<u>note</u> :-** sqrt(sum) = 2; is not a pseudovariable.

Program Organization

Types of Blocks

- **<u>PL/1 knows two types of blocks</u> :-**

  1. Procedures
  2. Begin-blocks

- **<u>Procedures are of two types</u> :-**

  1. External Procedures
  2. Internal Procedures

An EXTERNAL procedure is a block which is not contained in another block.
They are compiled separately but they can joined to make one program by linkage editor, so that 2 programmers can work on it simultaneously.
Option main identifies the main procedure.
i.e. label:proc options(main);

An INTERNAL procedure is invoked by a call from a surrounding block and control returns to statement after the call.
Within an external procedure you can have upto 50 internal procedures.

```
      F    e.g. :- outer:proc options(main);
                   statement1
                   call inner; --------
  ----------> statement2          :
 :   ^        inner:proc; <-------
 :   :              statement3
 :   :              statement4
 V    ------- end inner;
  ----------> statement5
                   statement6
            end outer;
```

BEGIN-block :- They always lie within a procedure and are executed where they are. They finish with an end statement.

```
      F    form :- A:proc;
              ......
              begin;
                ......
                ......
              end;
              ......
            end A;
```

Scope of Declaration

Variables which are declared have two scope attributes :-

1. Internal
2. External

If a variable has an INTERNAL scope, then it is known in the block in which it was declared and in all the blocks which physically lie in that block until it is not declared in one of these subordinate blocks.

```
e.g. :- outer:proc options(main);
              dcl rate fixed dec(7,3);
              dcl minus char(5);
              dcl x char(100);
```

```
          ......
          inner:proc;
               dcl minus fixed dec(7,2);
               dcl x char(100);
          end inner;
     end outer;
```

rate field is addressed in both blocks.
character field minus is not known in block inner.
The variable 'X' has attribute internal and hence you are dealing with 2 variables that are known within their blocks.
If a variable is to be same in two external procedures, then it must be declared with same attributes EXTERNAL.

```
mpro:proc options(main);          subpro:proc;
     dcl 1 rec external,          dcl 1 rec external,
           2 a char(30),               2 a char(30),
           2 b fixed bin(31);           2 b fixed bin(31);
```

Structure rec which is declared in both external procedures, occupies the same data area.

Storage Class in a Variable

Using storage class attributes, the programmer defines, when storage space is to be allocated to a variable and when this storage space is freed.

<u>There are 4 different storage class</u> :-

1. Automatic (default)
2. Static
3. Based
4. Controlled

AUTOMATIC Storage Class

An automatic variable has internal attribute.
Storage is allocated when block containing storage declaration is activated and is freed again when the block is terminated.

```
e.g. :- x:proc options(main);
          ......
          begin;   <-------------------------
            ......                          :
            dcl tab(100,100,10,2) char(25);  : activated
```

```
      ......                   :
    end;   <------------------------  freed
    ......
   end x;
```

STATIC Storage Class

Static variable has external attribute.
Storage is allocated before the execution of the program and remains allocated until end of the program. This storage class cannot be influenced by the programmer during program execution.

```
e.g. :- dcl tab(10,10)  fixed decimal(7,2) static;
        dcl x(10,10)    fixed decimal(5,2) based(p);
        dcl x char(120) static external;
```

BASED Storage Class

Based storage class is used in connection with record i/o. Based variable should always have attribute internal. In locate mode, it was necessary to declare the storage class based for record to be read in / written out.

```
e.g. :- dcl in file record buffered input
           env(fb recsize(100) blksize(1000) buffers(2));
        dcl out file record buffered output
           env(fb recsize(100) blksize(1000) buffers(2));
        dcl iarea char(100) based(q);
        dcl oarea char(100) based(p);
        dcl (q,p) pointer;
        ......
        read file(in) set(q);
        ......
        locate oarea file(out);
```

CONTROLLED Storage Class

With controlled variable the programmer has complete control over storage space. Using ALLOCATE space is assigned and with FREE the storage space is freed again. they can have the attributes internal or external.

```
e.g. :- dcl fieldx char(12) controlled;
        allocate fieldx;      /* storage space is assigned */
        fieldx = 'all clear ?';
        put file(printer) edit(fieldx) (x(5),A);
        free fieldx;          /* storage space is freed */
```

Subroutines & Functions

Subroutines

It is always an internal or external procedure. A subroutine is not replaced by a result value after its execution. A subroutine is always invoked with a call statement. Subroutines are often added as external procedures.

```
F   e.g. :- mrou:proc options(main);
    dcl ......;
    call srou1;
    ......
    call srou2;
 end mrou;

 srou1:proc;         srou2:proc
      dcl ......;     dcl ......;
      ......          ......
 end srou1;          end srou2;
```

Arguments

When a procedure is invoked using a call, it is possible to pass variables which are called as arguments.

```
F   e.g. :- dcl rec1 char(45);
 dcl 1 inrec,
      2 part1 char(10),
      2 amount fixed dec(7,2);

 call srou(rec1,inrec);   /* arguments */
```

Parameters

A program to which arguments are passed when it is invoked, are called as parameters.

```
F   e.g.:- srou:proc(a,b);
    dcl a char(100);

    dcl 1 b,
         2 b1 char(10),
         2 b2 fixed dec(7,2);
```

Here a & b are the parameters where a = rec1 and b = inrec.
The sequence and the attributes of the arguments and parameters must be identical. names may be different, but they use same area.
<u>**Note**</u> **:-** null arguments may be indicated by two commas and unnecessary arguments of the end of the list may be omitted.

Multiple Entry Points

Subroutine may have several entry points.
- primary entry points can be identified by PROC.
- secondary entry points can be identified by ENTRY statements.

Each entry points may have different parameter list.

```
e.g. :- prog1:proc options(main);
             call prog5(a,b);
             call ent2(a,b,c);
             call ent1(c);
        end prog1;

           proc5:proc(x,y);
             ......
             ent1:entry(m);
             ......
             ent2:entry(x,y,m);
             ......
        end prog5;
```

Exit from Procedures

Normal exit for a procedure is END.

Furthur exit can be provided using RETURN.

```
F    e.g. :- srou:proc(x,y);
          ......
      ent2:entry(m,n,o);
          ......
          if x = 0 then return;
          ......
      end srou;
```

Functions

Functions are always a part of an expression. A function is replaced by its result after execution.

F e.g. :- /* built-in functions */

```
x = sum(tab3);        /* with an argument */
```

adds up all elements of array tab.

```
v = date();           /* with empty arguments */
```

F A function call is replaced by value returned.

F e.g. :- calc:proc(a,b,c);

```
          ......
          if x > y then return(x*y);
          else return(z);
          ......
      end calc;
```

(ON-UNIT)

On ENDPAGE and on ENDFILE are some important conditions which can be forseen.

```
          e.g. :- on endpage(sysprint) begin;
        n = n + 1;

        put edit('page', n) (page, x(70), a, p'z9');
        put skip(3);
      end;
```

This writes the current page number when the preceding page is full.

SIGNAL & REVERT

<u>SIGNAL option</u> :-With signal statement, a condition can be caused artifically. The program continues with the statement which comes immediately after signal statement.
<u>form</u> :- SIGNAL condition;
Signal is used to raise conditions deliberately.

```
e.g. :- on endpage(print3) call overs;
      ......
      signal endpage(print3);
```

<u>REVERT option</u> :- Cancels on-unit for the condition specified.

```
e.g. :- on error call error1;
      /* error1 will be called */
      on error call error2;
      /* error2 will be called */
      revert error;
      /* cancels error2, error1 will be called */
```

Sample Programs

Sample Program 1

```
run221a:proc options(main);

      dcl alpha fixed decimal(7,0),
          beta  fixed decimal(6,0);
      dcl (a,b,c) fixed decimal(5,2),
          answer  fixed decimal(7,2);

      alpha = 100;
      beta  = alpha + 200;

      on endfile(sysin) goto fini;

      read:get file(sysin) list(a,b,c);
           answer=a+b-c;
           put file(sysprint) list(a,b,c,answer)
           goto read;

      fini:end run221a;
```

Sample Program 2

```
P1M0:proc options(main);
      dcl counter fixed dec(1);
      dcl total   fixed dec(7,2);
      dcl number  fixed dec(5,2);

      on endfile(sysin) goto finish;

      counter=0;
      total=0;

      read:
          get file(sysin) list(number);
          total   = total   + number;
          counter = counter + 1;
          goto read;

      finish:
          put file(sysprint) list(counter,total);

        end p1m0;
```

Sample Program 3

```
P3M0:proc options(main);
     dcl (countin, countout) fixed dec(3) init(0);
     dcl (totalin, totalout) fixed dec(7,2) init (0);
     dcl (averagein, averageout) fixed dec(5,2);
     dcl number fixed dec(5,2);

     on endfile(sysin) goto finish;

     read:get file(sysin) list(number);
```

```
          if number > 100 & number <= 60; then goto in;
          totalout = totalout + number;
          countout = countout + 1;
          goto read;

     in:  totalin = totalin + number;
          countin = countin + 1;
          goto read;
finish:
  averagein = totalin / countin;
  averageout = totalout / countout;

  put file(sysprint) list(countin, totalin, averagein);
  put file(sysprint) skip list(countout, totalout, averageout);

end;
P6M0:proc options(main);
     dcl temp(7,2) fixed dec(3), da(7) fixed dec(3);
     dcl wa fixed dec(3), i fixed bin(15), eof bit(1) init('0'B);

     on endfile(sysin) eof='1'B;

     get file(sysin) list(temp); /* seven pairs of temparature */
     do while (eof = '0'B);
        do i = 1 to 7;
           da(i) = (temp(i,1) + temp(i,2)) / 2;
        end;

        put file(sysprint) skip list(da); /* write to sysprint */
        put file(sysprint) skip list(wa); /* 7 values of da,wa */
        get file(sysin) list(temp);       /* new input of temp */
     end;
end p6m0;
```

Sample Program 5

```
P9M0:PROC OPTIONS(MAIN);

     dcl E1 file record env(F recsize(50), blksize(50));

     dcl A CHAR(50) init('THIS IS A TEST');
     write file(E1) from(A); /* output is written to file E1 */

     close file(E1);

     read file(E1) into(A);  /* reads data from file E1 */

     put data(A);            /* write  data item 'A' */
     put skip list('END of JOB'); /* write to file sysprint */

end P9M0;
```

Sample Program 6

```
p11m0:
proc options(main);
  dcl 1 indet,
        2 intype char(1),
        2 itemno char(5),
        2 uprice fixed dec(3,2),
        2 qty fixed dec(3),
        2 deptno char(3),
        2 code1 fixed dec(3),
        2 code2 fixed dec(3),
        2 code3 fixed dec(3),
        2 iname char(23),
        2 dname char(27);
  dcl ab char(40);
  dcl 1 sta based(addr(ab)),
        2 rectype char(4),
        2 itemno char(4),
        2 uprice fixed dec(3,2),
        2 qty fixed dec(3),
        2 totval fixed dec(7,2),
        2 iname char(23);

  dcl 1 stb based(addr(ab)),
        2 rectype char(4),
        2 deptno char(3),
        2 code1 fixed dec(3),
        2 code2 fixed dec(3),
        2 code3 fixed dec(3),
        2 dname char(27);
  dcl a1 file record env(fb recsize(40) blksize(400));
  dcl (eofi, eofa) bit(1) init('0'b);
  on endfile(sysin) eofi = '1'b;
  open file(a1) output;
  do until (eofi = '1'b);
  read file(sysin) into(indet);
  if ªeofi then do;
     if intype = 'A' then do;
        sta=indet, by name; sta.rectype = ‘fap7’;
        sta.totval = indet.uprice * indet.qty;
  end;
     else do;
```

```
      stb=indet, by name; stb.rectype = 'fbp7';
    end;
    write file(a1) from(ab);
  end;
  else close file(ab);
  on endfile(a1) eofa = '1'b;
open file(a1) input;
  do until (eofa);
    read file(a1) into (ab);
    if ªeofa then
      if sta.rectype = 'fap7' then
        put skip list(sta);
      else put skip list(stb);
    else close file(a1);
  end;
end p11m0;
```

Sample Program 7

```
P14M0:
proc options(main);
dcl ca char(78);
dcl (name,street,town) char(25);
dcl 1 prline,
    2 cc char(1) init(' '),
    2 text char(25);
dcl (del1,del2,del3) fixed bin(15);
dcl (sn,ss,st) fixed bin(15);
dcl (ln,ls,lt) fixed bin(15);

do until (eof);
  read file(sysin) into(ca);
  if ªeof then do;
    del1=index(ca,',');
    del2=index(substr(ca,del1+1),',') + del1;
    del3=index(substr(ca,del2+1),',') + del2;

    sn=1;
    ss=del1+1;
    st=del2+1;

    ln = del1 - sn;
    ls = del2 - ss;
    lt = del3 - st;

    name   = substr(ca,sn,ln);
    street = substr(ca,ss,ls);
    town   = substr(ca,st,lt);
    prline.text = name;
    write file(a1) from(prline);

    prline.text = street;
    write file(a1) from(prline);

    if verify(town,'.-ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789') ª= 0
    then prline.text = '*****';
    else prline.text = town;

    write file(a1) from(prline);
  end;
end;
end P14M0;
```

Sample Program 8

```
P16M0:proc P16M0 */
     proc options(main);
        dcl p16m2  entry;
        dcl (number,pvalue) fixed decimal(5,2);
        on eof bit(1) init('0'B);
        on endfile(sysin) eof = '1'B;
        do until(eof);
          get list(number);
          if ª eof then do;
            call p16m2(number,pvalue);
            put skip list(number,pvalue);
          end;
        end;
     end;

     P16M2:proc(number,pvalue);
     dcl (number,pvalue) fixed decimal(5,2);
     if number >= 100
       then pvalue = number * 0.05;
       else pvalue = number * 0.03;
     end
```