MAINFRAME STUDY NOTES

Quick Reference Guide

COBOL • JCL • DB2 • CICS • VSAM

Concise syntax, key concepts, and must-know commands for every mainframe topic — all in one place.

TABLE OF CONTENTS

CHAPTER 01

COBOL

Common Business-Oriented Language — the backbone of enterprise computing

Program Structure

Every COBOL program is divided into four **Divisions**, each with a specific purpose:

IDENTIFICATION DIVISION	Identifies the program — PROGRAM-ID, AUTHOR, DATE-WRITTEN.
ENVIRONMENT DIVISION	Links logical files to physical datasets (FILE-CONTROL).
DATA DIVISION	Declares all variables, files, and working storage (FD, 01-level items).
PROCEDURE DIVISION	Contains all executable logic — paragraphs and sections.

Data Division — Level Numbers

01	Top-level record or group item.
02–49	Elementary or subordinate group items.
66	RENAMES clause — alternative name for a range of items.
77	Independent elementary item (no group).
88	Condition name (VALUE clause used for IF checks).

PICTURE Clause Symbols

9	Numeric digit (0–9).
A	Alphabetic character (A–Z, space).
X	Alphanumeric — any character.
V	Implied decimal point (no physical character).
S	Sign (+ or –), must be leftmost character.
Z	Numeric digit; prints as space when zero (leading zero suppression).
.	Actual decimal point in edited picture.

,	Comma insertion in edited numeric output.
$	Currency symbol — floats to left of first significant digit.

Common COBOL Verbs

MOVE A TO B	Copy value of A into B.
ADD A B GIVING C	C = A + B (also SUBTRACT, MULTIPLY, DIVIDE).
COMPUTE C = A * B + D	Arithmetic expression — preferred for complex calculations.
IF cond … ELSE … END-IF	Conditional logic; always use END-IF scope terminator.
EVALUATE var … END-EVALUATE	Multi-way branch (equivalent of switch/case).
PERFORM para THRU para-X	Execute a paragraph range; use UNTIL or TIMES for loops.
OPEN / CLOSE file	Open in INPUT, OUTPUT, I-O, or EXTEND mode.
READ file INTO ws-record	Read next record; check AT END for EOF.
WRITE record FROM ws	Write a record to an output file.
REWRITE record	Update current record in I-O file.
DELETE file	Logically delete current record (VSAM only).
STRING … INTO … DELIMITED	Concatenate multiple strings into one field.
UNSTRING … INTO … DELIMITED	Split a string into multiple fields.
INSPECT var TALLYING / REPLACING	Count or replace character occurrences.
CALL 'SUBPGM' USING …	Invoke a subprogram; pass parameters BY REFERENCE or VALUE.
STOP RUN	Terminate program; return control to OS.
GOBACK	Return from subprogram to caller.

Quick COBOL Snippet

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
DATA DIVISION.
WORKING-STORAGE SECTION.
  01 WS-NAME   PIC X(20) VALUE 'WORLD'.
  01 WS-COUNT  PIC 9(4)  VALUE ZEROS.
PROCEDURE DIVISION.
  PERFORM VARYING WS-COUNT FROM 1 BY 1
          UNTIL WS-COUNT > 5
    DISPLAY 'HELLO ' WS-NAME
  END-PERFORM.
  STOP RUN.
```

■ **TIP:** Use 88-level condition names to make IF logic self-documenting: 88 IS-EOF VALUE 'Y'. Then: IF IS-EOF PERFORM END-OF-FILE.

■■ **NOTE:** COBOL is column-sensitive in fixed format: Areas A (cols 8–11) and B (cols 12–72). In free format (COBOL 2002+) this restriction is lifted.

CHAPTER 02

JCL

Job Control Language — tells the OS how to run your batch jobs

The Three Core Statements

//jobname JOB …	Marks start of a job; specifies accounting, class, priority, MSGCLASS.
//stepname EXEC …	Names a step; PGM= runs a program, PROC= invokes a catalogued procedure.
//ddname DD …	Defines a data set (DSN=, DISP=, SPACE=, DCB= etc.).

JOB Statement Key Parameters

CLASS=A	Job class — controls scheduling queue.
MSGCLASS=X	Output class for JES job log (spool).
MSGLEVEL=(1,1)	Controls which JCL statements and messages appear in the log.
NOTIFY=&SYSUID;	Send TSO notification when job completes.
REGION=0M	Memory limit (0M = use installation maximum).
TIME=(mm,ss)	CPU time limit; CANCEL if exceeded.
TYPRUN=SCAN	Syntax-check JCL without executing.

DD Statement Key Parameters

DSN=MY.DATA.SET	Dataset name (also DSNAME=).
DISP=(NEW,CATLG,DELETE)	Status, normal end, abnormal end dispositions.
SPACE=(TRK,(10,5),RLSE)	Primary, secondary allocation + release unused space.
UNIT=SYSDA	Device type for new datasets.
DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)	Record format, logical record length, blocksize.
*** or DATA**	In-stream data follows immediately in the JCL stream.

DUMMY	Nullifies the DD; reads return EOF, writes are discarded.
SYSOUT=*	Route output to default MSGCLASS spool.

Sample JCL Job

```
//MYJOB    JOB (ACCT),'MY JOB',CLASS=A,MSGCLASS=X,
//         NOTIFY=&SYSUID;,REGION=0M
//*--------------------------------------
//STEP1   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=MY.INPUT.FILE,DISP=SHR
//SYSUT2   DD DSN=MY.OUTPUT.FILE,
//         DISP=(NEW,CATLG,DELETE),
//         SPACE=(TRK,(5,2),RLSE),
//         DCB=(RECFM=FB,LRECL=80)
```

Common IBM Utilities

IEBGENER	Copy sequential datasets; simple record-by-record copy.
IEBCOPY	Copy/compress PDS or PDSE members.
IEFBR14	Do-nothing program — allocate or delete datasets via DD statements.
IDCAMS	VSAM and catalog management utility (DEFINE, DELETE, REPRO, LISTCAT).
SORT / ICETOOL	Sort, merge, copy, select, and transform records.
IEBPTPCH	Print or punch sequential/PDS datasets in formatted output.
IKJEFT01	TSO batch — run TSO commands in batch mode.

Condition Codes & COND Parameter

RC = 0	Successful completion.
RC = 4	Warning — job completed with minor issues.
RC = 8	Error — review output.
RC = 12	Severe error.
RC = 16	Terminal error.

COND=(4,LT)	Skip step if any previous RC < 4 (i.e., run only if RC >= 4).
COND=(0,NE,STEP1)	Skip this step if STEP1 RC is not zero.

■ **TIP:** Use //JCLLIB ORDER=(MY.PROCLIB) to search a private procedure library before the system libraries.

CHAPTER 03

DB2

IBM's relational database — SQL with COBOL embedded processing

Embedded SQL Basics

All SQL statements in COBOL are wrapped in **EXEC SQL … END-EXEC** delimiters. Host variables (COBOL fields) are prefixed with a colon (:) inside SQL.

```
EXEC SQL
  SELECT EMP-NAME, SALARY
  INTO  :WS-NAME, :WS-SALARY
  FROM  EMPLOYEE
  WHERE EMP-ID = :WS-EMP-ID
END-EXEC.

IF SQLCODE = 0
  DISPLAY 'FOUND: ' WS-NAME
ELSE
  DISPLAY 'SQLCODE: ' SQLCODE
END-IF.
```

SQLCODE Reference

0	Success — statement executed without error.
+100	Row Not Found — no row matched WHERE clause (or end of cursor).
-811	SELECT returned more than one row — use cursor instead.
-803	Duplicate key — unique constraint violated on INSERT/UPDATE.
-904	Unavailable resource — table/index locked or resource limit hit.
-911	Deadlock or timeout — transaction rolled back automatically.
-922	Authorization failure — ID lacks privilege on object.
-805	Package not found — program not bound to DB2.
-818	Timestamp mismatch — re-precompile and rebind the program.

Cursor Processing

Use a cursor when SELECT may return multiple rows.

```
EXEC SQL DECLARE EMP-CUR CURSOR FOR
  SELECT EMP-ID, EMP-NAME FROM EMPLOYEE
  WHERE DEPT = :WS-DEPT
  FOR READ ONLY
END-EXEC.

EXEC SQL OPEN EMP-CUR END-EXEC.

PERFORM UNTIL SQLCODE = +100
  EXEC SQL
    FETCH EMP-CUR INTO :WS-ID, :WS-NAME
  END-EXEC
  IF SQLCODE = 0
    DISPLAY WS-ID ' ' WS-NAME
  END-IF
END-PERFORM.

EXEC SQL CLOSE EMP-CUR END-EXEC.
```

DB2 Key Commands & Concepts

DCLGEN	Declaration Generator — creates COBOL copybook for a DB2 table layout.
PRECOMPILE	Translates EXEC SQL into COBOL calls; produces DBRM.
BIND	Creates an access plan (package/plan) from the DBRM.
COMMIT	Makes all changes in current unit-of-work permanent.
ROLLBACK	Undoes all changes in current unit-of-work.
LOCK TABLE … IN EXCLUSIVE MODE	Prevent concurrent access during batch update.
EXPLAIN	Shows access path chosen by the optimizer (stored in PLAN_TABLE).
RUNSTATS	Updates catalog statistics so the optimizer chooses efficient paths.
REORG	Reorganizes tablespace to reclaim space and restore clustering.

■ **TIP:** Always check SQLCODE immediately after every EXEC SQL. Use WHENEVER SQLERROR for global error trapping.

■■ **NOTE:** NULL values need indicator variables: PIC S9(4) COMP. Append :HOST-VAR :IND-VAR in SQL.

CHAPTER 04

CICS

Customer Information Control System — online transaction processing

Program Execution Flow

EXEC CICS RETURN END-EXEC	End task; no TRANSID = terminal goes idle.
EXEC CICS RETURN TRANSID('ABCD') COMMAREA(…) LENGTH(…) END-EXEC	Pseudo-converse: return and restart on next input.
EXEC CICS LINK PROGRAM('PGM') COMMAREA(…) END-EXEC	Call subprogram; waits for return (like CALL in COBOL).
EXEC CICS XCTL PROGRAM('PGM') COMMAREA(…) END-EXEC	Transfer control; calling program terminates.
EXEC CICS HANDLE CONDITION cond(para) END-EXEC	Route exceptional conditions to a paragraph.
EXEC CICS IGNORE CONDITION cond END-EXEC	Suppress a condition — program continues.
EXEC CICS ABEND ABCODE('XXXX') END-EXEC	Force an abend with a 4-char user code.

BMS — Map Send & Receive

```
* Send a map to the terminal:
EXEC CICS SEND MAP('MAPNAME') MAPSET('MAPSET1')
          ERASE FROM(WS-MAP-AREA) LENGTH(200)
END-EXEC.
* Receive input from terminal:
EXEC CICS RECEIVE MAP('MAPNAME') MAPSET('MAPSET1')
          INTO(WS-MAP-AREA) LENGTH(200)
END-EXEC.
```

File Control Commands

READ FILE(…) INTO(…) RIDFLD(key)	Read a record by key.
READ FILE(…) INTO(…) RIDFLD(key) UPDATE	Lock record for update.
REWRITE FILE(…) FROM(…)	Update the previously read-for-update record.
WRITE FILE(…) FROM(…) RIDFLD(key)	Insert new record.
DELETE FILE(…) RIDFLD(key)	Delete record by key.
STARTBR FILE(…) RIDFLD(key)	Start sequential browse.
READNEXT FILE(…) INTO(…) RIDFLD(key)	Read next record in browse.
ENDBR FILE(…)	End browse.

Temporary Storage & Transient Data

WRITEQ TS QUEUE(name) FROM(…)	Write to temporary storage queue (main or aux).
READQ TS QUEUE(name) INTO(…) ITEM(n)	Read a specific item from TS queue.
DELETEQ TS QUEUE(name)	Delete entire TS queue.
WRITEQ TD QUEUE(name) FROM(…)	Write to transient data (intra-partition or extra).
READQ TD QUEUE(name) INTO(…)	Destructive read from TD queue.

Common CICS Abend Codes

ASRA	Program check (S0C1–S0C7) — data exception, addressing error, etc.
AICA	Runaway task — infinite loop exceeded MAXTASK time.
AEI0	AEIP — invalid TRANSID or terminal not defined.
AKEA	Task killed by operator or exceeded storage limit.
AFCB	File not defined in FCT (File Control Table).

■ **TIP:** Use EIBAID (attention identifier byte) to detect which PF key the user pressed: IF EIBAID = DFHPF3 PERFORM END-SESSION.

■■ **NOTE:** EIBCALEN = 0 on first invocation — always check before addressing COMMAREA.

CHAPTER 05

VSAM

Virtual Storage Access Method — the standard mainframe file organization

VSAM File Types

KSDS — Key-Sequenced Data Set	Records stored by key order. Supports sequential, random, and skip-sequential access. Most common type.
ESDS — Entry-Sequenced Data Set	Records in insertion order; accessed by RBA (Relative Byte Address). No key. Used for logs.
RRDS — Relative Record Data Set	Fixed-length slots; accessed by relative record number (RRN). Like an array on disk.
LDS — Linear Data Set	Byte-stream; no VSAM record structure. Used by DB2 tablespaces and coupling facility.

IDCAMS — Define a KSDS

```
//DEFKSDS EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE CLUSTER                    -
    (NAME(MY.VSAM.FILE)             -
     CYL(5 1)                       -
     RECSZ(100 100)                 -
     KEYS(10 0)                     -
     INDEXED                        -
     SHAREOPTIONS(2 3))             -
  DATA                              -
    (NAME(MY.VSAM.FILE.DATA))       -
  INDEX                             -
    (NAME(MY.VSAM.FILE.INDEX)
/*
```

IDCAMS Utility Commands

DEFINE CLUSTER	Create a new VSAM dataset with data and index components.

DELETE cluster	Delete VSAM cluster and all components.
REPRO INFILE(…) OUTFILE(…)	Copy records between VSAM files or to/from sequential.
LISTCAT ENTRIES(name) ALL	Display catalog information for a dataset.
PRINT INFILE(…) COUNT(10)	Print records in HEX, CHAR, or DUMP format.
ALTER cluster NEWNAME(…)	Rename a dataset in the catalog.
VERIFY FILE(…)	Check and correct the end-of-file marker after a crash.
EXPORT / IMPORT	Backup and restore a VSAM cluster portably.

COBOL VSAM File Access

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
  SELECT EMPFILE ASSIGN TO EMPVSAM
    ORGANIZATION IS INDEXED
    ACCESS MODE  IS DYNAMIC
    RECORD KEY   IS EMP-ID
    FILE STATUS  IS WS-STATUS.

PROCEDURE DIVISION.
  OPEN I-O EMPFILE.
  MOVE '00100' TO EMP-ID.
  READ EMPFILE KEY IS EMP-ID
    INVALID KEY DISPLAY 'NOT FOUND'
  END-READ.
  IF WS-STATUS = '00'
    MOVE 'UPDATED NAME' TO EMP-NAME
    REWRITE EMPFILE-RECORD
  END-IF.
  CLOSE EMPFILE.
```

VSAM File Status Codes

00	Successful completion.
02	Duplicate alternate key (non-unique AIX) — not an error, record written.
10	End of file reached on sequential READ.

22	Duplicate prime key — record with this key already exists (INSERT rejected).
23	Record not found — key does not exist (READ / DELETE / REWRITE).
24	Boundary violation — WRITE beyond end of file extent.
35	File not found when OPEN attempted.
39	Conflicting DCB/OPEN attributes — LRECL or RECFM mismatch.
97	File already OPEN or open failed — check VERIFY then retry.

■ **TIP:** Use ACCESS MODE IS DYNAMIC to mix random keyed READs with sequential START/READNEXT browsing in the same program.

■■ **NOTE:** Always CLOSE VSAM files cleanly; an abnormal end leaves the end-of-file indicator dirty — run VERIFY before the next open.

END OF MAINFRAME STUDY NOTES

COBOL • JCL • DB2 • CICS • VSAM