COBOL Cheat Sheet

COBOL 2023 · GnuCOBOL 3.2 · Enterprise COBOL 6.5 · 2026-10-20

TOOLCHAIN

GnuCOBOL `cobc -x p.cob` exe · `-m` module (.so) · `-j` build+run · `-free` free format · `-std=ibm` IBM dialect · `-Wall` · `-g` · `--debug` run-time checks · `-I dir` copybooks · `cobcrun MOD` runs a module · env `COB_LIBRARY_PATH`, `COB_FILE_PATH`, `COB_SET_TRACE`, `COB_TRACE_FILE`, `COB_RUNTIME_CONFIG`

z/OS procs `IGYWCL` (compile+link) · `IGYWCLG` (+go). Options: `ARCH(n)` `OPT(0|1|2)` `TEST` `SSRANGE` `NUMCHECK` `TRUNC(STD|OPT|BIN)` `SQL` `CICS` `LIST` `MAP` `XREF` `OFFSET` · `cob2` on z/OS UNIX

FIXED-FORMAT RULER

1-6 seq	7	8-11 A	12-72 Area B	73-80

Col 7: `*` comment · `-` continuation · `/` page eject · `D` debug line. Area A: divisions, sections, paragraphs, `01`/`77`. Area B: statements. Free format: `*>` comment; switch with `>>SOURCE FORMAT FREE`. Statements end with a scope terminator (`END-IF`); period ends a sentence.

PROGRAM SKELETON

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PAYROLL.
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
  FILE-CONTROL. SELECT ...
DATA DIVISION.
 FILE SECTION.              FD, 01
 WORKING-STORAGE SECTION. per run unit
 LOCAL-STORAGE SECTION.   per call
 LINKAGE SECTION.         parameters
PROCEDURE DIVISION [USING ...].
 MAIN-PARA.
    PERFORM ...
    STOP RUN.   *> or GOBACK
```

`STOP RUN` ends the run unit; `GOBACK` returns to caller. Nested programs: `PROGRAM-ID. X` … `END PROGRAM X.` inside another; `COMMON`, `INITIAL`.

DATA DIVISION

Levels `01` record · `02`–`49` subordinate · `66` `RENAMES` · `77` stand-alone · `88` condition name. `FILLER` unnamed.

```
01 REC.
   05 ID    PIC 9(5) VALUE ZERO.
   05 NAME  PIC X(20).
   05 ST    PIC X VALUE 'A'.
      88 ACTIVE  VALUE 'A'.
01 RAW REDEFINES REC PIC X(26).
66 K RENAMES ID THRU NAME.
01 AMT IS TYPEDEF PIC S9(7)V99 COMP-3.
```

`INITIALIZE grp` (numeric→0, alnum→spaces) · `VALUE` sets initial value · `SPACE(S)` `ZERO(S)` `HIGH-VALUE` `LOW-VALUE` `QUOTE` `ALL "x"`

PICTURE SYMBOLS

`9` digit	`X` any char	`A` letter
`N` national	`U` UTF-8	`S` sign (no byte)
`V` implied point	`P` scaling	`(n)` repeat

Edited: `Z` zero→space · `*` check fill · `$` currency (float) · `,` `.` insert · `+` `-` sign · `CR` `DB` credit/debit · `B` blank · `0` zero · `/` slash

PIC	Value	Shown
`ZZ,ZZ9.99`	1234.5	`_1,234.50`
`$$$,$$9.99`	45.6	`____$45.60`
`***,**9.99`	12.3	`****12.30`
`-ZZ9`	-7	`__-7`
`ZZ9.99CR`	-5	`__5.00CR`
`99/99/9999`	12312026	`12/31/2026`

USAGE

Usage	Bytes	Notes
`DISPLAY`	1/digit	zoned; default; sign in zone
`COMP-3` packed	⌈(n+1)/2⌉	sign nibble C + D – F u
`COMP` / `BINARY`	2 / 4 / 8	1-4 / 5-9 / 10-18 digits; big-endian
`COMP-5`	2/4/8	native, full range
`COMP-1` / `-2`	4 / 8	float / double
`NATIONAL`	2/char	UTF-16
`UTF-8`	1-4/ch	2023 std
`INDEX`	4	table subscript
`POINTER`	4 / 8	`SET ADDRESS OF`

COMP-3 `PIC S9(4) COMP-3` = 3 bytes:

0 pad	1 d1	2 d2	3 d3	4 d4	C sign

`+1234` = `01 23 4C` `-1234` = `01 23 4D` `unsigned` `12345` = `12 34 5F`

TABLES

```
01 T.
  05 E OCCURS 1 TO 100 TIMES
       DEPENDING ON N
       ASCENDING KEY IS K
       INDEXED BY IX.
         10 K PIC 9(4).
SET IX TO 1   SET IX UP BY 1
SEARCH E AT END ... WHEN K(IX)=X ... END-SEARCH
SEARCH ALL E WHEN K(IX)=X ... END-SEARCH
```

Subscripts start at 1. `SEARCH ALL` is a binary search and needs a sorted table with `ASCENDING|DESCENDING KEY`. `SORT table ON ASCENDING KEY` sorts in memory. Subscripts: `T(I J)`; reference modification: `FLD(start:len)`.

MOVE & ARITHMETIC

MOVE numeric aligns on the decimal point (truncates high/low digits); alphanumeric left-justifies and pads with spaces; `CORRESPONDING` matches names; `MOVE X TO A B C.`

```
ADD A TO B GIVING C
SUBTRACT A FROM B
MULTIPLY A BY B GIVING C
DIVIDE A INTO B GIVING Q REMAINDER R
COMPUTE X ROUNDED = (A + B) ** 2 / C
   ON SIZE ERROR  DISPLAY "OVERFLOW"
   NOT ON SIZE ERROR ...
END-COMPUTE
```

`ROUNDED` (default nearest-away; `MODE IS ...`). Without `ON SIZE ERROR` results truncate silently.

CONDITIONS, IF, EVALUATE

Relation `= < > >= <= NOT` · class `IS NUMERIC|ALPHABETIC` · sign `IS POSITIVE|NEGATIVE|ZERO` · 88-level `IF ACTIVE` · compound `AND OR NOT`, abbreviated `A > 1 AND < 9`

```
IF A > B  ...  ELSE ...  END-IF
EVALUATE TRUE
  WHEN X < 0      ...
  WHEN 0 THRU 9   ...
  WHEN OTHER      ...
END-EVALUATE
EVALUATE A ALSO B  WHEN 1 ALSO ANY ...
```

PERFORM

```
PERFORM PARA    PERFORM A THRU B
PERFORM PARA 5 TIMES
PERFORM PARA UNTIL EOF
PERFORM WITH TEST AFTER UNTIL X > 9
PERFORM VARYING I FROM 1 BY 1
        UNTIL I > N
        AFTER J FROM 1 BY 1 UNTIL J > M ...
PERFORM UNTIL EXIT                *> 2002
   ... EXIT PERFORM / EXIT PERFORM CYCLE
END-PERFORM                        inline
```

Default test is **before**. Avoid `GO TO`.

STRINGS, FUNCTIONS, DATES, I/O

```
STRING A DELIMITED BY SIZE
       B DELIMITED BY SPACE
   INTO T WITH POINTER P
   ON OVERFLOW ... END-STRING
UNSTRING S DELIMITED BY "," INTO A B C
   TALLYING IN N
INSPECT S TALLYING N FOR ALL "x"
INSPECT S REPLACING ALL "a" BY "b"
INSPECT S CONVERTING lo TO UP
```

FUNCTION `LENGTH` `TRIM` `UPPER-CASE` `LOWER-CASE` `REVERSE` `NUMVAL` `NUMVAL-C` `MAX` `MIN` `MOD` `REM` `ABS` `SQRT` `INTEGER` `ORD` `CHAR` `RANDOM` `CURRENT-DATE` `INTEGER-OF-DATE` `DATE-OF-INTEGER`

Dates `ACCEPT d FROM DATE YYYYMMDD` · `DAY` · `TIME` · `CURRENT-DATE` = 21 chars `YYYYMMDDhhmmsshh+hhmm`. **I/O** `DISPLAY a b UPON CONSOLE` · `ACCEPT x` · `ACCEPT x FROM ARGUMENT-VALUE` · `ENVIRONMENT`.

ERRORS

`AT END` sequential read · `INVALID KEY` keyed I/O · `ON EXCEPTION` `CALL` / `ACCEPT` / `JSON` / `XML` · `ON OVERFLOW` `STRING` / `UNSTRING` · `ON SIZE ERROR` arithmetic. Check `FILE STATUS` after every I/O.

```
DECLARATIVES.
 H SECTION. USE AFTER STANDARD ERROR
            PROCEDURE ON IN-FILE.
  ...
END DECLARATIVES.
```

`EC-*` exception conditions (`EC-SIZE`, `EC-BOUND`) · `RETURN-CODE` special register becomes the step condition code (`MOVE 8 TO RETURN-CODE`); `STOP RUN WITH ERROR STATUS` (GnuCOBOL 3.2).

FILES

```
SELECT F ASSIGN TO "x.dat"   *> DD name on z/OS
  ORGANIZATION IS SEQUENTIAL|LINE SEQUENTIAL|
     INDEXED|RELATIVE
  ACCESS MODE IS SEQUENTIAL|RANDOM|DYNAMIC
  RECORD KEY IS K
  ALTERNATE KEY IS A WITH DUPLICATES
  FILE STATUS IS FS.
FD F. 01 F-REC ...
```

OPEN	READ	WRITE	REWRITE	DELETE	START
INPUT	yes	-	-	-	idx
OUTPUT	-	yes	-	-	-
I-O	yes	idx/rel	yes	idx/rel	idx
EXTEND	-	yes	-	-	-

`READ F NEXT|PREVIOUS INTO w [KEY IS k] AT END|INVALID KEY` · `START F KEY >= k` · `CLOSE F`. VSAM KSDS = `INDEXED`; ESDS = sequential; RRDS = `RELATIVE`.

FILE STATUS CODES

`00`	OK	`35`	file not found
`02`	OK, duplicate alt key	`37`	no permission/mode
`04`	OK, record length differs	`39`	attribute conflict
`05`	OK, optional file absent	`41`	already open
`10`	end of file	`42`	not open (close)
`21`	key sequence error	`43`	no prior READ
`22`	duplicate key	`44`	record size boundary
`23`	record not found	`46`	no valid next record
`24`	boundary violation	`47`	READ, not open INPUT/I-O
`30`	permanent error	`9x`	implementor-defined (VSAM)

First digit: **0** success · **1** at end · **2** invalid key · **3** permanent · **4** logic · **9** vendor.

SORT, MERGE & PATTERNS

```
SD SW. 01 SR. 05 K PIC X(5). ...
SORT SW ON ASCENDING KEY K
   USING IN  GIVING OUT
SORT SW ON DESCENDING KEY K
   INPUT PROCEDURE IS P1   *> RELEASE rec
   OUTPUT PROCEDURE IS P2  *> RETURN SW AT END
MERGE MW ON ASCENDING KEY K
   USING F1 F2 GIVING F3
```

Control break: read sorted input; when key ≠ saved key, print subtotal, reset, save key; final total at EOF. **Balance line:** two files sorted by key, compare keys: low key advances; equal = match/update.

SUBPROGRAMS

```
CALL "SUB" USING BY REFERENCE A
                 BY CONTENT   B
                 BY VALUE     N
     RETURNING R
  ON EXCEPTION ... END-CALL
CANCEL "SUB"
PROCEDURE DIVISION USING A B.
FUNCTION-ID. F. ... RETURNING R.
```

Static `CALL "X"` literal bound at link (z/OS `NODYNAM`), **dynamic** loaded at run time (`DYNAM` option, `CALL ident`). Default is by reference; `BY CONTENT` passes a copy; `BY VALUE` is for C. `LOCAL-STORAGE` is fresh per call; `INITIAL` program resets each call.

JSON & XML

```
JSON GENERATE J FROM REC COUNT IN N
  NAME OF fld IS "x" SUPPRESS fld
  ON EXCEPTION ... END-JSON
JSON PARSE J INTO REC WITH DETAIL
  NAME OF fld IS "x"
XML GENERATE X FROM REC COUNT IN N
XML PARSE X PROCESSING PROCEDURE P
  ON EXCEPTION ... END-XML
```

XML PARSE event handler reads `XML-EVENT`, `XML-TEXT`. `JSON-CODE` / `XML-CODE` hold the error code.

JCL

```
//PAYJOB  JOB (ACCT),'NAME',CLASS=A,
//   MSGCLASS=X
//STEP1   EXEC PGM=PAYROLL,PARM='X'
//STEPLIB DD DSN=MY.LOADLIB,DISP=SHR
//INFILE  DD DSN=MY.IN,DISP=SHR
//OUTFILE DD DSN=MY.OUT,DISP=(NEW,CATLG,DELETE),
//   SPACE=(CYL,(5,1),RLSE),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//GDG     DD DSN=MY.GDG(+1),DISP=(NEW,CATLG)
//SYSOUT  DD SYSOUT=*
//SYSIN   DD *
 data
/*
// IF STEP1.RC <= 4 THEN
// ENDIF
```

`DISP=(status,normal,abnormal)` : NEW OLD SHR MOD / CATLG KEEP DELETE PASS. `RC` 0 ok, 4 warning, 8 error, 12 severe, 16 terminal.

DB2 (EMBEDDED SQL)

```
EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC SQL SELECT NAME INTO :H-NAME :IND
   FROM T WHERE ID = :H-ID END-EXEC.
EXEC SQL DECLARE C1 CURSOR FOR
   SELECT ... FOR UPDATE OF X END-EXEC.
EXEC SQL OPEN C1  END-EXEC.
EXEC SQL FETCH C1 INTO :A, :B END-EXEC.
EXEC SQL CLOSE C1 END-EXEC.
EXEC SQL COMMIT END-EXEC.
```

`SQLCODE` **0** success · **+100** not found / end of cursor · **<0** error (–305 null w/o indicator, –803 duplicate key, –904 resource). Prepare: precompile or **coprocessor** (`SQL` option) → DBRM → `BIND PACKAGE` / `PLAN` → compile → link → run under `DSN`.

CICS

```
EXEC CICS RECEIVE MAP('M1') MAPSET('S1')
     INTO(MAPI) RESP(RC) END-EXEC.
EXEC CICS SEND MAP('M1') FROM(MAPO)
     ERASE END-EXEC.
EXEC CICS READ FILE('F') INTO(R)
     RIDFLD(K) END-EXEC.
EXEC CICS LINK PROGRAM('P')
     COMMAREA(C) END-EXEC.
EXEC CICS XCTL PROGRAM('Q') END-EXEC.
EXEC CICS RETURN TRANSID('TRN1')
     COMMAREA(C) END-EXEC.
```

Pseudo-conversational: task sends a map, returns with `TRANSID` + `COMMAREA`, ends; next input starts a new task (`EIBCALEN=0` first time). Test `RESP` against `DFHRESP(NORMAL|NOTFND|...)`. `LINK` returns; `XCTL` does not.

COMMON ABENDS (z/OS)

`S0C7`	data exception: invalid packed/zoned data (bad file layout; `NUMCHECK`, `IS NUMERIC`)
`S0C4`	protection: subscript out of range, bad `CALL` parameters, unset pointer (`SSRANGE`)
`S0CB`	divide by zero (`ON SIZE ERROR`)
`S322`	time limit: endless loop or volume growth
`S806`	load module not found (`STEPLIB`)
`S013`	OPEN failed: `RECFM` / `LRECL` mismatch
`U4038`	Language Environment abnormal end; read CEEDUMP

TOOLING & TESTING

VS Code: IBM Z Open Editor or Broadcom COBOL Language Support, plus Zowe Explorer (data sets, USS, jobs). **Zowe CLI**: `zowe zos-files list data-set "HLQ.*"` · `zowe zos-files download data-set` · `zowe zos-files upload file-to-data-set` · `zowe zos-jobs submit data-set` · `zowe zos-jobs list jobs`.

Debug: GnuCOBOL `-g --debug`, `COB_SET_TRACE`, `--coverage` (gcov); z/OS `TEST` option, z/OS Debugger, CEEDUMP. **Test**: COBOL Check, zUnit, Test4z, Galasa, GnuCOBOL + shell scripts in CI.

LEGACY → MODERN

`GO TO` / `ALTER` → `PERFORM` + `EVALUATE`	periods everywhere → scope terminators (1985)
`NEXT SENTENCE` → `CONTINUE`	`EXAMINE` → `INSPECT`
2-digit years → `CURRENT-DATE` / `YYYYMMDD`	OO COBOL → Java interop (6.4)
fixed format → free format (2002)	Micro Focus → Rocket
Zowe v1 → Zowe v3	COBOL Check → zUnit / Test4z / Galasa

Standards · 1985 scope terminators · 2002 free format, OO, UTF · 2014 JSON, floats · 2023 TYPEDEF, UTF-8 · Verify against the official documentation