GnuCOBOL Manual

for GnuCOBOL 2.0

Keisuke Nishida, Roger While, Brian Tiffin, Simon Sobisch

Edition 2.1
Updated for GnuCOBOL 2.0
22 December 2015

GnuCOBOL is a free and open-source COBOL compiler, which translates COBOL programs to C code and compiles it using GCC or other native operating system C compiler.

This manual corresponds to GnuCOBOL 2.0.

Written by Keisuke Nishida, Roger While, Brian Tiffin, Simon Sobisch.

Table of Contents

1 Getting Started

1.1 Hello World!

This is a sample program that displays "Hello World":

```
---- hello.cob -------------------------
      * Sample COBOL program
       IDENTIFICATION DIVISION.
       PROGRAM-ID. hello.
       PROCEDURE DIVISION.
       DISPLAY "Hello World!".
       STOP RUN.
----------------------------------------
```

The compiler is `cobc`, which is executed as follows:

```
$ cobc -x hello.cob
$ ./hello
Hello World!
```

The executable file name (i.e., `hello` in this case) is determined by removing the extension from the source file name.

You can specify the executable file name by specifying the compiler option `-o` as follows:

```
$ cobc -x -o hello-world hello.cob
$ ./hello-world
Hello World!
```

Using more modern sources.

```
---- hellonew.cob ----------------
*> Sample GnuCOBOL program
identification division.
program-id. hellonew.
procedure division.
display
   "Hello New World!"
end-display
goback.
---------------------------------
$ cobc -x -free hellonew.cob
$ ./hellonew
Hello New World!
```

Showing the use of free format, to end of line comments, the `goback` verb, and proper use of terminator with `end-display`.

2 Compile

This chapter describes how to compile COBOL programs using GnuCOBOL.

2.1 Compiler Options

The compiler `cobc` accepts the options described in this section.
General syntax -
`cobc` [options] file [file ..]
A complete list of options can be displayed by using the help option.

2.1.1 Help Options

The following switches can be used for informational displays:

`--help` Display help screen (see Appendix A [cobc –help], page 21), `-h` will also display the help.
No further actions will be taken.

`--version`
Display compiler version, author package date and executable build date. `-V` will also display version.
No further actions will be taken.

`--info` Display build information along with default and current compiler configuration.
No further actions will be taken except for further display options.

`-v` Verbosely displays the programs invoked during compilations.

`--list-reserved`
Display reserved words(see Appendix B [cobc –list-reserved], page 24). A Y/N field shows if the word is supported.[1] No further actions will be taken except for further display options.

`--list-intrinsics`
Display intrinsic functions (see Appendix C [cobc –list-intrinsics], page 36). A Y/N field shows if the function is implemented. No further actions will be taken except for further display options.

`--list-system`
Display system routines (see Appendix D [cobc –list-system], page 39). No further actions will be taken except for further display options.

`--list-mnemonics`
Display mnemonic names (see Appendix E [cobc –list-mnemonics], page 41). No further actions will be taken except for further display options.

2.1.2 Built Target

The `cobc` compiler can handle `*.cob`, `*.cbl` as COBOL source code, `*.c` for C source code, `*.o` for object code, `*.i` for preprocessed code and `*.so` for dynamic modules and will do the right thing in terms of generation, compilation, or link.

The special input name `-` takes input from `stdin` which is assumed to be COBOL source, and uses a default output name of `a.out` (or `a.so,c,o,i`) as appropriate for the build type.

The following options specify the target type produced by the compiler:

[1] Support may be partial or complete

`-E` Preprocess only. Compiler directives are executed. Comment lines are removed. COPY statements are expanded.
The output is saved in file `*.i`.

`-C` Translation only. COBOL source files are translated into C files.
The output is saved in file `*.c`.

`-S` Compile only. Translated C files are compiled by the C compiler to assembler code.
The output is saved in file `*.s`.

`-c` Compile and assemble. This is equivalent to `cc -c`.
The output is saved in file `*.o`.

`-m` Compile, assemble, and build a dynamically loadable module (i.e., a shared library).
The output is saved in file `*.so`.
This is the default behaviour if not other options are given.[2].

`-b` Compile, assemble, and combine all input files into a single dynamically loadable module. Unless `-o` is also used, the output is saved using the first filename as `*.so`.

`-x` Include the main function in the output, creating an executable image. The main entry point being the outermost `PROGRAM-ID`.

This option takes effect at the translation stage. If you give this option with `-C`, you will see the main function at the end of the generated C file.

`-j` Run job after compilation. Either from executable with -x, or with `cobcrun` when compiling a module.

`-I <directory>`
Add <directory> to copy/include search path

`-L <directory>`
Add <directory> to library search path

`-l <lib>` Link the library <lib>

`-D <define>`
Pass <define> to the COBOL compiler

`-o <file>` Place the output into <file>

Without any options above, the compiler builds a dynamically loadable module.

2.1.3 Source Format

GnuCOBOL supports both fixed and free source format.

The default format is the fixed format. This can be explicitly overwritten by one of the following options:

`-free` Free format. The program-text area starts in column 1 and continues till the end of line. Effectively 255 characters in GnuCOBOL. `-F` will also set free format, useful when using `cobc` as a shell interpreter directive to the program loader.

`-fixed` Fixed format. Source code is divided into a 1-6 column sequence number area, column 7 indicator area, columns 8-72 program-text area, with columns 72-80 as a reference area. Historically this format is based on 80 character punch cards. FIXED format is the default used by the compiler unless overridden by compiler switch or source code directive, `>>SOURCE [FORMAT] [IS] {FIXED|FREE}`.

[2] The extension varies depending on your host.

2.1.4 Warning Options

`-W` Enable every possible warning. This includes more information than -Wall would normally provide.

`-Wall` Enable all common warnings

`-Warchaic`
Warn if archaic features are used

`-Wcall-params`
Warn non 01/77 items for CALL params (NOT set with -Wall)

`-Wcolumn-overflow`
Warn if text after column 72 in FIXED format (NOT set with -Wall)

`-Wconstant`
Warn inconsistent constant

`-Wimplicit-define`
Warn implicitly defined data items

`-Wlinkage`
Warn dangling LINKAGE items (NOT set with -Wall)

`-Wobsolete`
Warn if obsolete features are used

`-Wparentheses`
Warn lack of parentheses around AND within OR

`-Wredefinition`
Warn incompatible redefinition of data items

`-Wstrict-typing`
Warn type mismatch strictly

`-Wterminator`
Warn lack of scope terminator END-XXX (NOT set with -Wall)

`-Wtruncate`
Warn possible field truncation (NOT set with -Wall)

`-Wunreachable`
Warn unreachable statements (NOT set with -Wall)

2.1.5 Configuration Options

`-std=<dialect>`
Compiler uses the given dialect to determine certain compiler features and warnings. See Appendix F [Appendix F], page 44, and `config/*.conf`.

`-std=cobol2002`
COBOL 2002

`-std=cobol2014`
COBOL 2014

`-std=cobol85`
COBOL 85

`-std=ibm` IBM Compatible

`-std=mvs` MVS Compatible

`-std=bs2000`
: BS2000 Compatible

`-std=mf` Micro Focus Compatible

`-std=acu` ACUCOBOL-GT Compatible

`-std=default`
: When not specified

`-conf=<file>`
: User defined dialect configuration. See `-std=` above.
See Appendix F [Appendix F], page 44, and `config/*.conf`.

`-cb_conf=<tag:value>`
: Override a single configuration entry. See Appendix F [Appendix F], page 44.

2.1.6 Debug Switches

`-debug` Enable all run-time error checks. `-d` will also enable all run-time error checks, useful when using `cobc` as a shell interpreter directive to the program loader.

`-g` Produce debugging information in the output.

`-O` Enable optimization of code size and execution speed. See `man gcc` for details.

`-O2` Optimize even more.

`-Os` Optimize for size. Optimizer will favour code size over execution speed.

`-ftrace` Generate trace code (Executed SECTION/PARAGRAPH)

`-ftraceall`
: Generate trace code (Executed SECTION/PARAGRAPH/STATEMENTS)

`-fsyntax-only`
: Syntax error checking only; don't emit any output

`-fdebugging-line`
: Enable debugging lines ('D' in indicator column)

`-fsource-location`
: Generate source location code (Turned on by -debug or -g)

`-fimplicit-init`
: Do automatic initialization of the COBOL runtime system

`-fstack-check`
: PERFORM stack checking (Turned on by -debug or -g)

`-fnotrunc`
: Do not truncate binary fields according to PICTURE

2.1.7 Miscellaneous

`-P` Generate and place a program listing into `*.lst`

`-ext <extension>`
: Add default file extension

`-fmfcomment`
: '*' or '/' in column 1 treated as comment (FIXED only)

`-acucomment`
: '|' is treated as inline comment marker

`-fsign=ASCII`
: Numeric display sign ASCII (Default on ASCII machines)

`-fsign=EBCDIC`
: Numeric display sign EBCDIC (Default on EBCDIC machines)

`-ffunctions-all`
: Allow use of intrinsic functions without FUNCTION keyword

`-ffold-copy=LOWER`
: Fold COPY subject to lower case (Default no transformation)

`-ffold-copy=UPPER`
: Fold COPY subject to upper case (Default no transformation)

`-save-temps(=<dir>)`
: Save intermediate files (default current directory)

2.2 Multiple Sources

A program often consists of multiple source files. This section describes how to compile multiple source files.

This section also describes how to build a shared library that can be used by any COBOL programs and how to use external libraries from COBOL programs.

2.2.1 Static Linking

The easiest way of combining multiple files is to compile them into a single executable.

One way is to specify all files on the command line:

```
$ cobc -x -o prog main.cob subr1.cob subr2.cob
```

Another way is to compile each file with the option `-c`, and link them at the end. The top-level program must be compiled with the option `-x`:

```
$ cobc -c subr1.cob
$ cobc -c subr2.cob
$ cobc -c -x main.cob
$ cobc -x -o prog main.o subr1.o subr2.o
```

You can link C routines as well using either method:

Method 1:

```
$ cobc -o prog main.cob subrs.c
```

Method 2:

```
$ cobc -c subrs.c
$ cobc -c -x main.cob
$ cobc -x -o prog main.o subrs.o
```

Any number of functions can be contained in a single C file.

The linked programs will be called dynamically; that is, the symbol will be resolved at run time. For example, the following COBOL statement

```
CALL "subr" USING X.
```

will be converted into an equivalent C code like this:

```
int (*func)() = cob_resolve("subr");
if (func != NULL)
  func (X);
```

With the compiler options `-fstatic-call`, more efficient code will be generated like this:

```
subr(X);
```

Note that this option is effective only when the called program name is a literal (like `CALL "subr".`). With a data name (like `CALL SUBR.`), the program is still called dynamically.

2.2.2 Dynamic Linking

There are two methods to achieve this. Method 1 (Using driver program). Compile all programs with the option `-m`:

```
$ cobc -m main.cob subr.cob
```

This creates shared object files `main.so subr.so`[3].

Before running the main program, install the module files in your library directory:

```
$ cp subr.so /your/cobol/lib
```

Set the runtime variable `COB_LIBRARY_PATH` to your library directory, and run the main program:

```
$ export COB_LIBRARY_PATH=/your/cobol/lib
```

Note: You may set the variable via a runtime configuration file, See Appendix H [Appendix H], page 48. You may set the variable to directly point to the directory where you compiled the sources.

Now execute your program:

```
$ cobcrun main
```

Method 2. The main program and subprograms can be compiled separately.

The main program is compiled as usual:

```
$ cobc -x -o main main.cob
```

Subprograms are compiled with the option `-m`:

```
$ cobc -m subr.cob
```

This creates a module file `subr.so`[4].

Before running the main program, install the module files in your library directory:

```
$ cp subr.so /your/cobol/lib
```

Now, set the environment variable `COB_LIBRARY_PATH` to your library directory, and run the main program:

```
$ export COB_LIBRARY_PATH=/your/cobol/lib
$ ./main
```

2.2.3 Building Library

You can build a shared library by combining multiple COBOL programs and even C routines:

```
$ cobc -c subr1.cob
$ cobc -c subr2.cob
$ cc -c subr3.c
$ cc -shared -o libsubrs.so subr1.o subr2.o subr3.o
```

[3] The extension varies depending on your host.

[4] The extension varies depending on your host.

2.2.4 Using Library

You can use a shared library by linking it with your main program.

Before linking the library, install it in your system library directory:

```
$ cp libsubrs.so /usr/lib
```

or install it somewhere else and set `LD_LIBRARY_PATH`:

```
$ cp libsubrs.so /your/cobol/lib
$ export LD_LIBRARY_PATH=/your/cobol/lib
```

Then, compile the main program, linking the library as follows:

```
$ cobc -x main.cob -L/your/cobol/lib -lsubrs
```

2.3 C Interface

This chapter describes how to combine C programs with COBOL programs.

2.3.1 Writing Main Program in C

Include `libcob.h` in your C program. Call `cob_init` before using any COBOL module:

```
#include <libcob.h>

int
main (int argc, char **argv)
{
  /* initialize your program */
  ...

  /* initialize the COBOL run-time library */
  cob_init (argc, argv);

  /* rest of your program */
  ...

  /* Clean up and terminate - This does not return */
  cob_stop_run (return_status);

}
```

You can write `cobc_init(0, NULL);` if you do not want to pass command line arguments to COBOL.

You can compile your C program as follows:

```
cc -c `cob-config --cflags` main.c
```

The compiled object must be linked with libcob as follows:

```
cc -o main main.o `cob-config --libs`
```

2.3.2 Static linking with COBOL programs

Let's call the following COBOL module from a C program:

```
---- say.cob ---------------------------
       IDENTIFICATION DIVISION.
       PROGRAM-ID. say.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       LINKAGE SECTION.
```

```
        01 HELLO PIC X(6).
        01 WORLD PIC X(6).
        PROCEDURE DIVISION USING HELLO WORLD.
        DISPLAY HELLO WORLD.
        EXIT PROGRAM.
---------------------------------------
```

This program accepts two arguments, displays them, and exits.

From the viewpoint of C, this is equivalent to a function having the following prototype:

```
extern int say(char *hello, char *world);
```

So, your main program will look like as follows:

```
---- hello.c --------------------------
#include <libcob.h>

extern int say(char *hello, char *world);

int
main()
{
  int ret;
  char hello[7] = "Hello ";
  char world[7] = "World!";

  cob_init(0, NULL);

  ret = say(hello, world);

  return ret;
}
---------------------------------------
```

Compile these programs as follows:

```
$ cc -c `cob-config --cflags` hello.c
$ cobc -c -static say.cob
$ cobc -x -o hello hello.o say.o
$ ./hello
Hello World!
```

2.3.3 Dynamic linking with COBOL programs

You can find a COBOL module having a specific PROGRAM-ID by using a C function `cob_resolve`, which takes the module name as a string and returns a pointer to the module function.

`cob_resolve` returns NULL if there is no module. In this case, the function `cob_resolve_error` returns the error message.

Let's see an example:

```
---- hello-dynamic.c ------------------
#include <libcob.h>

static int (*say)(char *hello, char *world);

int
main()
```

```
{
  int ret;
  char hello[7] = "Hello ";
  char world[7] = "World!";

  cob_init(0, NULL);

  /* find the module with PROGRAM-ID "say". */
  say = cob_resolve("say");

  /* if there is no such module, show error and exit */
  if (say == NULL) {
    fprintf(stderr, "%s\n", cob_resolve_error ());
    exit(1);
  }

  /* call the module found and exit with the return code */
  ret = say(hello, world);

  return ret;
}
----------------------------------------
```

Compile these programs as follows:

```
$ cc -c `cob-config --cflags` hello-dynamic.c
$ cobc -x -o hello hello-dynamic.o
$ cobc -m say.cob
$ export COB_LIBRARY_PATH=.
$ ./hello
Hello World!
```

2.3.4 Static linking with C programs

Let's call the following C function from COBOL:

```
---- say.c ----------------------------
int
say(char *hello, char *world)
{
  int i;
  for (i = 0; i < 6; i++)
    putchar(hello[i]);
  for (i = 0; i < 6; i++)
    putchar(world[i]);
  putchar('\n');
  return 0;
}
----------------------------------------
```

This program is equivalent to the foregoing `say.cob`.

Note that, unlike C, the arguments passed from COBOL programs are not terminated by the null character (i.e., `\0`).

You can call this function in the same way you call COBOL programs:

```
---- hello.cob ------------------------
```

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. hello.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 HELLO PIC X(6) VALUE "Hello ".
       01 WORLD PIC X(6) VALUE "World!".
       PROCEDURE DIVISION.
       CALL "say" USING HELLO WORLD.
       STOP RUN.
----------------------------------------
```

Compile these programs as follows:

```
$ cc -c say.c
$ cobc -c -static -x hello.cob
$ cobc -x -o hello hello.o say.o
$ ./hello
Hello World!
```

2.3.5 Dynamic linking with C programs

You can create a dynamic-linking module from a C program by passing an option `-shared` to the C compiler:

```
$ cc -shared -o say.so say.c
$ cobc -x hello.cob
$ export COB_LIBRARY_PATH=.
$ ./hello
Hello World!
```

3 Customize

3.1 Customizing Compiler

These settings are effective at compile-time.

Environment variables (default value):

`COB_CC` C compiler ("gcc")

`COB_CFLAGS`
: Flags passed to the C compiler ("-I$(PREFIX)/include")

`COB_LDFLAGS`
: Flags passed to the C compiler ("")

`COB_LIBS` Standard libraries linked with the program ("-L$(PREFIX)/lib -lcob")

`COB_LDADD`
: Additional libraries linked with the program ("")

3.2 Customizing Library

These settings are effective at run-time. You can set them either via environment or by a runtime configuration file.

To set global runtime configuration file export `COB_RUNTIME_CONFIG` pointing to your configuration file. To set an explicit runtime configuration file for a single run via `cobcrun` you can use its option `-c <file>, -config=<file>`.

For diplaying the current runtime settings you can use the option `-r, -runtime-env` of `cobcrun`.

For a complete list of runtime variables, aliases, their default values and options to set them See Appendix H [Appendix H], page 48.

4 Optimize

4.1 Optimize Options

There are three compiler options for optimization: `-O`, `-Os` and `-O2`. These options enable optimization at both translation (from COBOL to C) and compilation (C to assembly) levels.

Currently, there is no difference between these optimization options at the translation level.

The option `-O`, `-Os` or `-O2` is passed to the C compiler as it is and used for C level optimization.

4.2 Optimize Call

When a CALL statement is executed, the called program is linked at run time. By specifying the compiler option `-fstatic-call`, you can statically link the program at compile time and call it efficiently. (see Section 2.2.1 [Static Linking], page 6)

4.3 Optimize Binary

By default, data items of usage binary or comp are stored in the big-endian form. On those machines whose native byte order is little-endian, this is not quite efficient.

If you prefer, you can store binary items in the native form of your machine. Set the config option `binary-byteorder` to `native` in your config file (see Chapter 3 [Customize], page 12).

In addition, setting the option `binary-size` to `2-4-8` or `1-2-4-8` is more efficient than others.

5 Debug

5.1 Debug Options

The compiler option `-debug` can be used during the development of your programs. It enables all run-time error checking, such as subscript boundary checks and numeric data checks, and displays run-time errors with source locations.

6 Extensions not ISO/ANSI standard

6.1 SELECT ASSIGN TO

<This section is in progress.>

6.2 Indexed file packages

<This section is in progress.>

6.3 Extended ACCEPT statement

Extended ACCEPT statements allow for full control of items accepted from the screen. Items accept by line and column positioning.

```
ACCEPT variable-1
   LINE <line> COLUMN <column>
   WITH
      AUTO-SKIP | AUTO
      [PROTECTED] SIZE [IS] variable-2 | literal-2
END-ACCEPT.
```

6.3.1 AUTO-SKIP

With this option the ACCEPT statement returns after the last character is typed at the end of the field. This is the same as if the Enter key were pressed.

Without this option the cursor remains at the end of the field and waits for the user to press Enter.

The word AUTO may be used for AUTO-SKIP.

The Right-Arrow key returns from the end of the field. The Left-Arrow key returns from the beginning. See Section 6.4 [ACCEPT Special], page 16.

The Alt-Right-Arrow and Alt-Left-Arrow keys never AUTO-SKIP.

6.3.2 PROTECTED

PROTECTED is ignored. It is optional.

6.3.3 SIZE

The size of variable-1 to accept from the screen. It is optional.

`SIZE <greater than zero>`
: If SIZE is less than the length of variable-1 then only the SIZE number of characters accept into the field. Variable-1 pads with spaces after SIZE to the end of the field.

  If SIZE is greater than variable-1, then the screen pads with spaces after variable-1 to the SIZE length.

`SIZE ZERO`
`<SIZE option not specified>`
: The variable-1 field accepts with its length.

6.4 ACCEPT special keys

Special keys are available for Extended ACCEPT statements.

The COB-CRT-STATUS values are in the screenio.cpy copy file.

6.4.1 Arrow keys

The Left-Arrow key moves the cursor to the left. Without AUTO-SKIP the cursor stops at the beginning of the field. With AUTO-SKIP it returns with the COB-SCR-KEY-LEFT value of 2009. See Section 6.3 [Extended ACCEPT], page 15.

The Alt-Left-Arrow key is the same as Left-Arrow except that it never returns, even for AUTO-SKIP.

The Right-Arrow key moves the cursor to the right. Without AUTO-SKIP the cursor stops at the end of the field. With AUTO-SKIP it returns with the COB-SCR-KEY-RIGHT value of 2010. See Section 6.3 [Extended ACCEPT], page 15.

The Alt-Right-Arrow key is the same as Right-Arrow except that it never returns, even for AUTO-SKIP.

6.4.2 Backspace key

The Backspace key moves the cursor, and the remainder of the text, to the left.

6.4.3 Delete keys

The Delete key deletes the cursor's character and moves the remainder of the text to the left. The cursor does not move.

The Alt-Delete key deletes all text from the cursor to the end of the field.

6.4.4 End keys

The End key moves the cursor after the last non-space character.

The Alt-End key moves the cursor to the end of the field.

6.4.5 Home keys

The Home key moves the cursor to the first non-space character.

The Alt-Home key moves the cursor to the beginning of the field.

6.4.6 Insert key

The Insert key changes the insert mode.

When the insert mode is on, typed characters move the existing characters to the right. When it is off, typed characters type over existing characters.

The default insert mode is set by the COB_INSERT_MODE variable, See Appendix H [Appendix H], page 48. This must be set before the first Extended ACCEPT, DISPLAY, or any routine that gets information from the screen.

The last press of the Insert key is used in all following ACCEPT statements while the program is running.

6.4.7 Tab keys

The Tab key returns from the ACCEPT with the COB-SCR-TAB value of 2007.

The Shift-Tab key returns with the COB-SCR-BACK-TAB value of 2008.

6.5 Extended DISPLAY statement

Extended DISPLAY statements allow for full control of items that display on the screen. Items display by line and column positioning.

```
DISPLAY variable-1 | literal-1 | figurative constant
   LINE <line> COLUMN <column>
   WITH BELL
        BLANK LINE | SCREEN
        ERASE EOL | EOS
        SIZE [IS] variable-2 | literal-2
END-DISPLAY.
```

6.5.1 BELL

Ring the bell. It is optional.

6.5.2 BLANK

Clear the whole line or screen. It is optional.

`BLANK LINE`
: Clear the line from the beginning of the line to the end of the line.

`BLANK SCREEN`
: Clear the whole screen.

6.5.3 ERASE

Clear the line or screen from LINE and COLUMN. It is optional.

`ERASE EOL`
: Clear the line from LINE and COLUMN to the end of the line.

`ERASE EOS`
: Clear the screen from LINE and COLUMN to the end of the screen.

6.5.4 SIZE

The size of variable-1, literal-1, or figurative constant to display onto the screen. It is optional.

`SIZE <greater than zero>`
: If SIZE is less than the length of variable-1 or literal-1 then only the SIZE number of characters display.

  If SIZE is greater than the length of variable-1 or literal-1, then the screen pads with spaces after the field to the SIZE length.

  Figurative constants display repeatedly the number of times in SIZE. Except that LOW-VALUES always positions the cursor (see SIZE ZERO below).

`SIZE ZERO`
`<SIZE option not specified>`
: Variable-1 or literal-1 displays with the field length.

  Certain figurative constants have special functions.

  SPACE Display spaces from LINE and COLUMN to the end of the screen. This is the same as WITH ERASE EOS.

  LOW-VALUE Position the cursor to LINE and COLUMN. The next DISPLAY statement does not need a LINE or COLUMN to display at that position.

ALL "1" Display spaces from LINE and COLUMN to the end of the line. This is the same as WITH ERASE EOL.

ALL "2" Clear the whole screen. This is the same as WITH BLANK SCREEN.

ALL "7" Ring the bell. This is the same as WITH BELL.

All other figurative constants display as a single character.

7 System routines

For a complete list of supported system routines See Appendix D [cobc –list-system], page 39.

7.1 CBL_OC_GETOPT

CBL_OC_GETOPT realises the quite well-known option parser getopt for GnuCOBOL. The usage of this system routine is described by the following example.

```
identification division.
program-id. prog.

data division.
working-storage section.
    78 shortoptions value "jkl".

    01 longoptions.
        05 optionrecord occurs 2 times.
            10 optionname   pic x(25).
            10 has-value    pic 9.
            10 valpoint     pointer value NULL.
            10 return-value pic x(4).

    01 longind     pic 99.
    01 long-only   pic 9 value 1.

    01 return-char pic x(4).
    01 opt-val     pic x(10).

    01 counter     pic 9 value 0.
```

We first need to define the necessary fields for getopt's shortoptions (so), longoptions (lo), longoption index (longind), long-only-option (long-only) and also the fields for return values return-char and opt-val (arbitrary size with trimming, see return codes).
The shortoptions are written down as an alphanumeric field (string with arbitrary size) as follows:

```
"ab:c::d"
```

This means we want getopt to look for shortoptions named a, b, c or d and we demand an option value for b and we are accepting an optional one for c.

The longoptions are defined as a table of records with oname, has-value, valpoint and val. The field oname defines the name of a longoption, has-value defines if an option value is demanded(has-val = 1), optional(2) or not required(0).
The longoption structure is immutable! You can vary the amount of records only. The pointer valpoint is used to specify an address to save getopt's return value to. The pointer is optional.

If it is NULL, getopt returns a value as usual. If you use the pointer it has to point to a PIC X(4) field.
The field val is a PIC X(4) character which is returned if the longoption was recognized.

Now we have the tools to run CBL_OC_GETOPT within the procedure division.

```
procedure division.
    move "version" to optionname   (1).
    move 0         to has-value    (1).
    move "v"       to return-value (1).

    move "verbose" to optionname   (2).
    move 0         to has-value    (2).
    move "V"       to return-value (2).

    perform with test after until return-code = -1
        call 'CBL_OC_GETOPT' using
           by reference shortoptions longoptions longind
           by value long-only
           by reference return-char opt-val
        end-call

        display return-char end-display
        display opt-val     end-display
    end-perform
    stop run.
```

The example shows how we initialize all parameters and call the routine until as CBL_OC_GETOPT doesn't find any option, returning '-1' in this case.

The return-char might contain the following:

- regular character if an option was recognized
- '?' if we have got an undefined or ambiguous option
- '1' if got a non-option (only if first byte of so is '-')
- '0' if valpoint != NULL and we are writing the return value to the specified address
- '-1' if we don't have any more options (or reach the first non-option if first byte of so is '+')

The return-codes of CBL_OC_GETOPT are:

- 1 if we've got a non-option (only if first byte of so is '-')
- 0 if valpoint != NULL and we are writing the return value to the specified address
- -1 if we don't have any more options (or reach the first non-option if first byte of so is '+')
- 2 if we have got an truncated option value in opt-val (because opt-val was too small)
- 3 if we got a regular answer from getopt

Appendix A cobc --help

```
Usage: /d/Programme/Entwicklung/GnuCOBOL/gnu-cobol-2.0_ultraclean2/cobc/.libs/cobc [options]

GnuCOBOL compiler for most COBOL dialects with lots of extensions

Options:
  -h, -help             Display this help and exit
  -V, -version          Display compiler version and exit
  -i, -info             Display compiler information (build/environment)
  -v, -verbose          Display the commands invoked by the compiler
  -vv                   Display compiler version and display the commands
                        invoked by the compiler
  -x                    Build an executable program
  -m                    Build a dynamically loadable module (default)
  -j                    Run job, after build
  -std=<dialect>        Warnings/features for a specific dialect:
                          cobol2002   COBOL 2002
                          cobol2014   COBOL 2014
                          cobol85     COBOL 85
                          ibm         IBM Compatible
                          mvs         MVS Compatible
                          bs2000      BS2000 Compatible
                          mf          Micro Focus Compatible
                          acu         ACUCOBOL-GT Compatible
                          default     When not specified
                        See config/default.conf and config/*.conf
  -free                 Use free source format
  -fixed                Use fixed source format (default)
  -F                    Alias (short option) for -free
  -O, -O2, -Os          Enable optimization
  -g                    Enable C compiler debug / stack check / trace
  -d, -debug            Enable all run-time error checking
  -o <file>             Place the output into <file>
  -b                    Combine all input files into a single
                        dynamically loadable module
  -E                    Preprocess only; do not compile or link
  -C                    Translation only; convert COBOL to C
  -S                    Compile only; output assembly file
  -c                    Compile and assemble, but do not link
  -P(=<dir or file>)    Generate preprocessed program listing (.lst)
  -Xref                 Generate cross reference through 'cobxref'
                        (V. Coen's 'cobxref' must be in path)
  -I <directory>        Add <directory> to copy/include search path
  -L <directory>        Add <directory> to library search path
  -l <lib>              Link the library <lib>
  -A <options>          Add <options> to the C compile phase
  -Q <options>          Add <options> to the C link phase
  -D <define>           DEFINE <define> to the COBOL compiler
  -K <entry>            Generate CALL to <entry> as static
  -conf=<file>          User defined dialect configuration - See -std=
  -cb_conf=<tag:value>  Override configuration entry
```

```
-list-reserved        Display reserved words
-list-intrinsics      Display intrinsic functions
-list-mnemonics       Display mnemonic names
-list-system          Display system routines
-save-temps(=<dir>)   Save intermediate files
                      - Default : current directory
-ext <extension>      Add default file extension

-W                    Enable ALL warnings
-Wall                 Enable all warnings except as noted below
-Wobsolete            Warn if obsolete features are used
-Warchaic             Warn if archaic features are used
-Wredefinition        Warn incompatible redefinition of data items
-Wconstant            Warn inconsistent constant
-Woverlap             Warn overlapping MOVE items
-Wparentheses         Warn lack of parentheses around AND within OR
-Wstrict-typing       Warn type mismatch strictly
-Wimplicit-define     Warn implicitly defined data items
-Wcorresponding       Warn CORRESPONDING with no matching items
-Wexternal-value      Warn EXTERNAL item with VALUE clause
-Wcall-params         Warn non 01/77 items for CALL params
                      - NOT set with -Wall
-Wcolumn-overflow     Warn text after column 72, FIXED format
                      - NOT set with -Wall
-Wterminator          Warn lack of scope terminator END-XXX
                      - NOT set with -Wall
-Wtruncate            Warn possible field truncation
                      - NOT set with -Wall
-Wlinkage             Warn dangling LINKAGE items
                      - NOT set with -Wall
-Wunreachable         Warn unreachable statements
                      - NOT set with -Wall

-fsign=<value>        Define display sign representation
                      - ASCII or EBCDIC (Default : machine native)
-ffold-copy=<value>   Fold COPY subject to value
                      - UPPER or LOWER (Default : no transformation)
-ffold-call=<value>   Fold PROGRAM-ID, CALL, CANCEL subject to value
                      - UPPER or LOWER (Default : no transformation)
-fdefaultbyte=<value> Initialize fields without VALUE to decimal value
                      - 0 to 255 (Default : initialize to picture)
-fintrinsics=<value>  Intrinsics to be used without FUNCTION keyword
                      - ALL or intrinsic function name (,name,...)
-ftrace               Generate trace code
                      - Executed SECTION/PARAGRAPH
-ftraceall            Generate trace code
                      - Executed SECTION/PARAGRAPH/STATEMENTS
                      - Turned on by -debug
-fsyntax-only         Syntax error checking only; don't emit any output
-fdebugging-line      Enable debugging lines
                      - 'D' in indicator column or floating >>D
-fsource-location     Generate source location code
```

```
                      - Turned on by -debug/-g/-ftraceall
  -fimplicit-init     Automatic initialization of the Cobol runtime system
  -fstack-check       PERFORM stack checking
                      - Turned on by -debug or -g
  -fsyntax-extension  Allow syntax extensions
                      - eg. Switch name SW1, etc.
  -fwrite-after       Use AFTER 1 for WRITE of LINE SEQUENTIAL
                      - Default : BEFORE 1
  -fmfcomment         '*' or '/' in column 1 treated as comment
                      - FIXED format only
  -facucomment        '$' in indicator area treated as '*',
                      '|' treated as floating comment
  -fnotrunc           Allow numeric field overflow
                      - Non-ANSI behaviour
  -fodoslide          Adjust items following OCCURS DEPENDING
                      - Requires implicit/explicit relaxed syntax
  -fsingle-quote      Use a single quote (apostrophe) for QUOTE
                      - Default : double quote
  -frecursive-check   Check recursive program call
  -frelax-syntax      Relax syntax checking
                      - eg. REDEFINES position
  -foptional-file     Treat all files as OPTIONAL
                      - unless NOT OPTIONAL specified
```

Appendix B cobc --list-reserved

Reserved Words	Implemented (Y/N)
ACCEPT	Y
ACCESS	Y
ACTIVE-CLASS	N
ADD	Y
ADDRESS	Y
ADVANCING	Y
AFTER	Y
ALIGNED	N
ALL	Y
ALLOCATE	Y
ALPHABET	Y
ALPHABETIC	Y
ALPHABETIC-LOWER	Y
ALPHABETIC-UPPER	Y
ALPHANUMERIC	Y
ALPHANUMERIC-EDITED	Y
ALSO	Y
ALTER	Y
ALTERNATE	Y
AND	Y
ANY	Y
ANYCASE	N
ARE	Y
AREA	Y
AREAS	Y
ARGUMENT-NUMBER	Y
ARGUMENT-VALUE	Y
ARITHMETIC	N (Context sensitive)
AS	Y
ASCENDING	Y
ASCII	Y (Context sensitive)
ASSIGN	Y
AT	Y
ATTRIBUTE	Y (Context sensitive)
AUTO	Y
AUTO-SKIP	Y
AUTOMATIC	Y
AUTOTERMINATE	Y
AWAY-FROM-ZERO	Y (Context sensitive)
B-AND	N
B-NOT	N
B-OR	N
B-XOR	N
BACKGROUND-COLOR	Y
BACKGROUND-COLOUR	Y
BASED	Y
BEEP	Y

```
BEFORE                          Y
BELL                            Y
BINARY                          Y
BINARY-C-LONG                   Y
BINARY-CHAR                     Y
BINARY-DOUBLE                   Y
BINARY-INT                      Y
BINARY-LONG                     Y
BINARY-LONG-LONG                Y
BINARY-SHORT                    Y
BIT                             N
BLANK                           Y
BLINK                           Y
BLOCK                           Y
BOOLEAN                         N
BOTTOM                          Y
BY                              Y
BYTE-LENGTH                     Y (Context sensitive)
CALL                            Y
CANCEL                          Y
CAPACITY                        Y (Context sensitive)
CD                              N (85 obsolete)
CENTER                          N (Context sensitive)
CF                              Y
CH                              Y
CHAIN                           N
CHAINING                        Y
CHARACTER                       Y
CHARACTERS                      Y
CLASS                           Y
CLASS-ID                        N
CLASSIFICATION                  Y (Context sensitive)
CLOSE                           Y
CODE                            Y
CODE-SET                        Y
COL                             Y
COLLATING                       Y
COLS                            Y
COLUMN                          Y
COLUMNS                         Y
COMMA                           Y
COMMAND-LINE                    Y
COMMIT                          Y
COMMON                          Y
COMMUNICATION                   N (85 obsolete)
COMP                            Y
COMP-1                          Y
COMP-2                          Y
COMP-3                          Y
COMP-4                          Y
COMP-5                          Y
COMP-6                          Y
```

```
COMP-X                          Y
COMPUTATIONAL                   Y
COMPUTATIONAL-1                 Y
COMPUTATIONAL-2                 Y
COMPUTATIONAL-3                 Y
COMPUTATIONAL-4                 Y
COMPUTATIONAL-5                 Y
COMPUTATIONAL-X                 Y
COMPUTE                         Y
CONDITION                       Y
CONFIGURATION                   Y
CONSTANT                        Y
CONTAINS                        Y
CONTENT                         Y
CONTINUE                        Y
CONTROL                         Y
CONTROLS                        Y
CONVERSION                      Y (Context sensitive)
CONVERTING                      Y
COPY                            Y
CORR                            Y
CORRESPONDING                   Y
COUNT                           Y
CRT                             Y
CRT-UNDER                       Y
CURRENCY                        Y
CURSOR                          Y
CYCLE                           Y (Context sensitive)
DATA                            Y
DATA-POINTER                    N
DATE                            Y
DAY                             Y
DAY-OF-WEEK                     Y
DE                              Y
DEBUGGING                       Y
DECIMAL-POINT                   Y
DECLARATIVES                    Y
DEFAULT                         Y
DELETE                          Y
DELIMITED                       Y
DELIMITER                       Y
DEPENDING                       Y
DESCENDING                      Y
DESTINATION                     N
DETAIL                          Y
DISABLE                         N
DISC                            Y (Context sensitive)
DISK                            Y (Context sensitive)
DISPLAY                         Y
DIVIDE                          Y
DIVISION                        Y
DOWN                            Y
```

```
DUPLICATES                      Y
DYNAMIC                         Y
EBCDIC                          Y (Context sensitive)
EC                              Y
EGI                             N (85 obsolete)
ELSE                            Y
EMI                             N (85 obsolete)
EMPTY-CHECK                     Y
ENABLE                          N (85 obsolete)
END                             Y
END-ACCEPT                      Y
END-ADD                         Y
END-CALL                        Y
END-CHAIN                       N
END-COMPUTE                     Y
END-DELETE                      Y
END-DISPLAY                     Y
END-DIVIDE                      Y
END-EVALUATE                    Y
END-IF                          Y
END-MULTIPLY                    Y
END-OF-PAGE                     Y
END-PERFORM                     Y
END-READ                        Y
END-RECEIVE                     N (85 obsolete)
END-RETURN                      Y
END-REWRITE                     Y
END-SEARCH                      Y
END-START                       Y
END-STRING                      Y
END-SUBTRACT                    Y
END-UNSTRING                    Y
END-WRITE                       Y
ENTRY                           Y
ENTRY-CONVENTION                N (Context sensitive)
ENVIRONMENT                     Y
ENVIRONMENT-NAME                Y
ENVIRONMENT-VALUE               Y
EO                              N
EOL                             Y (Context sensitive)
EOP                             Y
EOS                             Y (Context sensitive)
EQUAL                           Y
EQUALS                          Y
ERASE                           Y
ERROR                           Y
ESCAPE                          Y
ESI                             N (85 obsolete)
EVALUATE                        Y
EXCEPTION                       Y
EXCEPTION-OBJECT                N
EXCLUSIVE                       Y
```

```
EXIT                            Y
EXPANDS                         N (Context sensitive)
EXTEND                          Y
EXTERNAL                        Y
FACTORY                         N
FALSE                           Y
FD                              Y
FILE                            Y
FILE-CONTROL                    Y
FILE-ID                         Y
FILLER                          Y
FINAL                           Y
FIRST                           Y
FLOAT-BINARY-128                N
FLOAT-BINARY-32                 N
FLOAT-BINARY-64                 N
FLOAT-DECIMAL-16                Y
FLOAT-DECIMAL-34                Y
FLOAT-EXTENDED                  N
FLOAT-INFINITY                  N
FLOAT-LONG                      Y
FLOAT-NOT-A-NUMBER              N (Context sensitive)
FLOAT-SHORT                     Y
FOOTING                         Y
FOR                             Y
FOREGROUND-COLOR                Y
FOREGROUND-COLOUR               Y
FOREVER                         Y
FORMAT                          N
FREE                            Y
FROM                            Y
FULL                            Y
FUNCTION                        Y
FUNCTION-ID                     Y
FUNCTION-POINTER                N
GENERATE                        Y
GET                             N
GIVING                          Y
GLOBAL                          Y
GO                              Y
GOBACK                          Y
GREATER                         Y
GROUP                           Y
GROUP-USAGE                     N
HEADING                         Y
HIGH-VALUE                      Y
HIGH-VALUES                     Y
HIGHLIGHT                       Y
I-O                             Y
I-O-CONTROL                     Y
ID                              Y
IDENTIFICATION                  Y
```

```
IF                              Y
IGNORE                          Y
IGNORING                        Y
IMPLEMENTS                      N (Context sensitive)
IN                              Y
INDEX                           Y
INDEXED                         Y
INDICATE                        Y
INDIRECT                        N (Context sensitive)
INHERITS                        N
INITIAL                         Y
INITIALISE                      Y
INITIALISED                     Y
INITIALIZE                      Y
INITIALIZED                     Y
INITIATE                        Y
INPUT                           Y
INPUT-OUTPUT                    Y
INSPECT                         Y
INTERFACE                       N
INTERFACE-ID                    N
INTERMEDIATE                    N (Context sensitive)
INTO                            Y
INTRINSIC                       Y (Context sensitive)
INVALID                         Y
INVOKE                          N
IS                              Y
JUST                            Y
JUSTIFIED                       Y
KEPT                            Y
KEY                             Y
KEYBOARD                        Y (Context sensitive)
LABEL                           Y
LAST                            Y
LC_ALL                          N (Context sensitive)
LC_COLLATE                      N (Context sensitive)
LC_CTYPE                        N (Context sensitive)
LC_MESSAGES                     N (Context sensitive)
LC_MONETARY                     N (Context sensitive)
LC_NUMERIC                      N (Context sensitive)
LC_TIME                         N (Context sensitive)
LEADING                         Y
LEFT                            Y
LEFT-JUSTIFY                    N
LEFTLINE                        Y
LENGTH                          Y
LENGTH-CHECK                    Y
LESS                            Y
LIMIT                           Y
LIMITS                          Y
LINAGE                          Y
LINAGE-COUNTER                  Y
```

```
LINE                          Y
LINE-COUNTER                  Y
LINES                         Y
LINKAGE                       Y
LOCAL-STORAGE                 Y
LOCALE                        Y
LOCK                          Y
LOW-VALUE                     Y
LOW-VALUES                    Y
LOWER                         Y (Context sensitive)
LOWLIGHT                      Y
MANUAL                        Y
MEMORY                        Y
MERGE                         Y
MESSAGE                       N (85 obsolete)
METHOD                        N
METHOD-ID                     N
MINUS                         Y
MODE                          Y
MOVE                          Y
MULTIPLE                      Y
MULTIPLY                      Y
NAME                          Y (Context sensitive)
NATIONAL                      Y
NATIONAL-EDITED               Y
NATIVE                        Y
NEAREST-AWAY-FROM-ZERO        Y (Context sensitive)
NEAREST-EVEN                  Y (Context sensitive)
NEAREST-TOWARD-ZERO           Y (Context sensitive)
NEGATIVE                      Y
NESTED                        N
NEXT                          Y
NO                            Y
NO-ECHO                       Y
NONE                          N (Context sensitive)
NORMAL                        Y (Context sensitive)
NOT                           Y
NULL                          Y
NULLS                         Y
NUMBER                        Y
NUMBERS                       Y
NUMERIC                       Y
NUMERIC-EDITED                Y
OBJECT                        N
OBJECT-COMPUTER               Y
OBJECT-REFERENCE              N
OCCURS                        Y
OF                            Y
OFF                           Y
OMITTED                       Y
ON                            Y
ONLY                          Y
```

```
OPEN                          Y
OPTIONAL                      Y
OPTIONS                       N
OR                            Y
ORDER                         Y
ORGANISATION                  Y
ORGANIZATION                  Y
OTHER                         Y
OUTPUT                        Y
OVERFLOW                      Y
OVERLINE                      Y
OVERRIDE                      N
PACKED-DECIMAL                Y
PADDING                       Y
PAGE                          Y
PAGE-COUNTER                  Y
PARAGRAPH                     Y (Context sensitive)
PERFORM                       Y
PF                            Y
PH                            Y
PIC                           Y
PICTURE                       Y
PLUS                          Y
POINTER                       Y
POSITION                      Y
POSITIVE                      Y
PREFIXED                      N (Context sensitive)
PRESENT                       Y
PREVIOUS                      Y
PRINTER                       Y (Context sensitive)
PRINTING                      Y
PROCEDURE                     Y
PROCEDURE-POINTER             Y
PROCEDURES                    Y
PROCEED                       Y
PROGRAM                       Y
PROGRAM-ID                    Y
PROGRAM-POINTER               Y
PROHIBITED                    Y (Context sensitive)
PROMPT                        Y
PROPERTY                      N
PROTECTED                     Y
PROTOTYPE                     N
PURGE                         N (85 obsolete)
QUEUE                         N (85 obsolete)
QUOTE                         Y
QUOTES                        Y
RAISE                         N
RAISING                       N
RANDOM                        Y
RD                            Y
READ                          Y
```

```
RECEIVE                         N (85 obsolete)
RECORD                          Y
RECORDING                       Y
RECORDS                         Y
RECURSIVE                       Y (Context sensitive)
REDEFINES                       Y
REEL                            Y
REFERENCE                       Y
REFERENCES                      Y
RELATION                        N (Context sensitive)
RELATIVE                        Y
RELEASE                         Y
REMAINDER                       Y
REMOVAL                         Y
RENAMES                         Y
REPLACE                         Y
REPLACING                       Y
REPORT                          Y
REPORTING                       Y
REPORTS                         Y
REPOSITORY                      Y
REQUIRED                        Y
RESERVE                         Y
RESET                           Y
RESUME                          N
RETRY                           N
RETURN                          Y
RETURNING                       Y
REVERSE-VIDEO                   Y
REVERSED                        Y
REWIND                          Y
REWRITE                         Y
RF                              Y
RH                              Y
RIGHT                           Y
RIGHT-JUSTIFY                   N
ROLLBACK                        Y
ROUNDED                         Y
ROUNDING                        N (Context sensitive)
RUN                             Y
SAME                            Y
SCREEN                          Y
SCROLL                          Y (Context sensitive)
SD                              Y
SEARCH                          Y
SECONDS                         N (Context sensitive)
SECTION                         Y
SECURE                          Y
SEGMENT                         N (85 obsolete)
SEGMENT-LIMIT                   Y
SELECT                          Y
SELF                            N
```

```
SEND                          N (85 obsolete)
SENTENCE                      Y
SEPARATE                      Y
SEQUENCE                      Y
SEQUENTIAL                    Y
SET                           Y
SHARING                       Y
SIGN                          Y
SIGNED                        Y
SIGNED-INT                    Y
SIGNED-LONG                   Y
SIGNED-SHORT                  Y
SIZE                          Y
SORT                          Y
SORT-MERGE                    Y
SOURCE                        Y
SOURCE-COMPUTER               Y
SOURCES                       N
SPACE                         Y
SPACE-FILL                    N
SPACES                        Y
SPECIAL-NAMES                 Y
STANDARD                      Y
STANDARD-1                    Y
STANDARD-2                    Y
STANDARD-BINARY               N (Context sensitive)
STANDARD-DECIMAL              N (Context sensitive)
START                         Y
STATEMENT                     N (Context sensitive)
STATIC                        Y (Context sensitive)
STATUS                        Y
STDCALL                       Y (Context sensitive)
STEP                          Y
STOP                          Y
STRING                        Y
STRONG                        N (Context sensitive)
SUB-QUEUE-1                   N (85 obsolete)
SUB-QUEUE-2                   N (85 obsolete)
SUB-QUEUE-3                   N (85 obsolete)
SUBTRACT                      Y
SUM                           Y
SUPER                         N
SUPPRESS                      Y
SYMBOL                        N (Context sensitive)
SYMBOLIC                      Y
SYNC                          Y
SYNCHRONISED                  Y
SYNCHRONIZED                  Y
SYSTEM-DEFAULT                Y
TAB                           Y (Context sensitive)
TABLE                         N
TALLYING                      Y
```

```
TAPE                            Y (Context sensitive)
TERMINAL                        N (85 obsolete)
TERMINATE                       Y
TEST                            Y
TEXT                            N (85 obsolete)
THAN                            Y
THEN                            Y
THROUGH                         Y
THRU                            Y
TIME                            Y
TIME-OUT                        Y (Context sensitive)
TIMEOUT                         Y (Context sensitive)
TIMES                           Y
TO                              Y
TOP                             Y
TOWARD-GREATER                  Y (Context sensitive)
TOWARD-LESSER                   Y (Context sensitive)
TRAILING                        Y
TRAILING-SIGN                   N
TRANSFORM                       Y
TRUE                            Y
TRUNCATION                      Y (Context sensitive)
TYPE                            Y
TYPEDEF                         N
UCS-4                           N (Context sensitive)
UNDERLINE                       Y
UNIT                            Y
UNIVERSAL                       N
UNLOCK                          Y
UNSIGNED                        Y
UNSIGNED-INT                    Y
UNSIGNED-LONG                   Y
UNSIGNED-SHORT                  Y
UNSTRING                        Y
UNTIL                           Y
UP                              Y
UPDATE                          Y
UPON                            Y
UPPER                           Y (Context sensitive)
USAGE                           Y
USE                             Y
USER                            Y (Context sensitive)
USER-DEFAULT                    Y
USING                           Y
UTF-16                          N (Context sensitive)
UTF-8                           N (Context sensitive)
VAL-STATUS                      N
VALID                           N
VALIDATE                        N
VALIDATE-STATUS                 N
VALUE                           Y
VALUES                          Y
```

```
VARYING                         Y
WAIT                            Y
WHEN                            Y
WITH                            Y
WORDS                           Y
WORKING-STORAGE                 Y
WRITE                           Y
YYYYDDD                         Y (Context sensitive)
YYYYMMDD                        Y (Context sensitive)
ZERO                            Y
ZERO-FILL                       N
ZEROES                          Y
ZEROS                           Y

Extra (obsolete) context sensitive words
AUTHOR
DATE-COMPILED
DATE-MODIFIED
DATE-WRITTEN
INSTALLATION
REMARKS
SECURITY

Extra internal registers        Definition
RETURN-CODE                     USAGE BINARY-LONG
SORT-RETURN                     USAGE BINARY-LONG
NUMBER-OF-CALL-PARAMETERS       USAGE BINARY-LONG
COB-CRT-STATUS                  PIC 9(4)
TALLY                           GLOBAL PIC 9(5) USAGE BINARY VALUE ZERO
'LENGTH OF' phrase              USAGE BINARY-LONG
```

Appendix C cobc --list-intrinsics

Intrinsic Function	Implemented	Parameters
ABS	Y	1
ACOS	Y	1
ANNUITY	Y	2
ASIN	Y	1
ATAN	Y	1
BOOLEAN-OF-INTEGER	N	2
BYTE-LENGTH	Y	1
CHAR	Y	1
CHAR-NATIONAL	N	1
COMBINED-DATETIME	Y	2
CONCATENATE	Y	Unlimited
COS	Y	1
CURRENCY-SYMBOL	Y	0
CURRENT-DATE	Y	0
DATE-OF-INTEGER	Y	1
DATE-TO-YYYYMMDD	Y	1 - 3
DAY-OF-INTEGER	Y	1
DAY-TO-YYYYDDD	Y	1 - 3
DISPLAY-OF	N	1 - 2
E	Y	0
EXCEPTION-FILE	Y	0
EXCEPTION-FILE-N	N	0
EXCEPTION-LOCATION	Y	0
EXCEPTION-LOCATION-N	N	0
EXCEPTION-STATEMENT	Y	0
EXCEPTION-STATUS	Y	0
EXP	Y	1
EXP10	Y	1
FACTORIAL	Y	1
FORMATTED-CURRENT-DATE	Y	1
FORMATTED-DATE	Y	2
FORMATTED-DATETIME	Y	3 - 4
FORMATTED-TIME	Y	2 - 3
FRACTION-PART	Y	1
HIGHEST-ALGEBRAIC	Y	1
INTEGER	Y	1
INTEGER-OF-BOOLEAN	N	1
INTEGER-OF-DATE	Y	1
INTEGER-OF-DAY	Y	1
INTEGER-OF-FORMATTED-DATE	Y	2
INTEGER-PART	Y	1
LENGTH	Y	1
LENGTH-AN	Y	1
LOCALE-COMPARE	Y	2 - 3
LOCALE-DATE	Y	1 - 2
LOCALE-TIME	Y	1 - 2
LOCALE-TIME-FROM-SECONDS	Y	1 - 2
LOG	Y	1

LOG10	Y	1
LOWER-CASE	Y	1
LOWEST-ALGEBRAIC	Y	1
MAX	Y	Unlimited
MEAN	Y	Unlimited
MEDIAN	Y	Unlimited
MIDRANGE	Y	Unlimited
MIN	Y	Unlimited
MOD	Y	2
MODULE-CALLER-ID	Y	0
MODULE-DATE	Y	0
MODULE-FORMATTED-DATE	Y	0
MODULE-ID	Y	0
MODULE-PATH	Y	0
MODULE-SOURCE	Y	0
MODULE-TIME	Y	0
MONETARY-DECIMAL-POINT	Y	0
MONETARY-THOUSANDS-SEPARATOR	Y	0
NATIONAL-OF	N	1 - 2
NUMERIC-DECIMAL-POINT	Y	0
NUMERIC-THOUSANDS-SEPARATOR	Y	0
NUMVAL	Y	1
NUMVAL-C	Y	2
NUMVAL-F	Y	1
ORD	Y	1
ORD-MAX	Y	Unlimited
ORD-MIN	Y	Unlimited
PI	Y	0
PRESENT-VALUE	Y	Unlimited
RANDOM	Y	Unlimited
RANGE	Y	Unlimited
REM	Y	2
REVERSE	Y	1
SECONDS-FROM-FORMATTED-TIME	Y	2
SECONDS-PAST-MIDNIGHT	Y	0
SIGN	Y	1
SIN	Y	1
SQRT	Y	1
STANDARD-COMPARE	N	2 - 4
STANDARD-DEVIATION	Y	Unlimited
STORED-CHAR-LENGTH	Y	1
SUBSTITUTE	Y	Unlimited
SUBSTITUTE-CASE	Y	Unlimited
SUM	Y	Unlimited
TAN	Y	1
TEST-DATE-YYYYMMDD	Y	1
TEST-DAY-YYYYDDD	Y	1
TEST-FORMATTED-DATETIME	Y	2
TEST-NUMVAL	Y	1
TEST-NUMVAL-C	Y	2
TEST-NUMVAL-F	Y	1
TRIM	Y	1 - 2

```
UPPER-CASE                    Y             1
VARIANCE                      Y             Unlimited
WHEN-COMPILED                 Y             0
YEAR-TO-YYYY                  Y             1 - 3
```

Appendix D cobc --list-system

System routine	Parameters
SYSTEM	1
CBL_AND	3
CBL_CHANGE_DIR	1
CBL_CHECK_FILE_EXIST	2
CBL_CLOSE_FILE	1
CBL_COPY_FILE	2
CBL_CREATE_DIR	1
CBL_CREATE_FILE	5
CBL_DELETE_DIR	1
CBL_DELETE_FILE	1
CBL_EQ	3
CBL_ERROR_PROC	2
CBL_EXIT_PROC	2
CBL_FLUSH_FILE	1
CBL_GET_CSR_POS	1
CBL_GET_CURRENT_DIR	3
CBL_GET_SCR_SIZE	2
CBL_IMP	3
CBL_NIMP	3
CBL_NOR	3
CBL_NOT	2
CBL_OC_GETOPT	6
CBL_OC_NANOSLEEP	1
CBL_OPEN_FILE	5
CBL_OR	3
CBL_READ_FILE	5
CBL_RENAME_FILE	2
CBL_TOLOWER	2
CBL_TOUPPER	2
CBL_WRITE_FILE	5
CBL_XOR	3
C$CALLEDBY	1
C$CHDIR	2
C$COPY	3
C$DELETE	2
C$FILEINFO	2
C$GETPID	0
C$JUSTIFY	1
C$MAKEDIR	1
C$NARG	1
C$PARAMSIZE	1
C$PRINTABLE	1
C$SLEEP	1
C$TOLOWER	2
C$TOUPPER	2
X"91"	2
X"E4"	0

```
X"E5"                           0
X"F4"                           2
X"F5"                           2
```

Appendix E cobc --list-mnemonics

```
Mnemonic names
SYSIN           Device name
SYSIPT          Device name
STDIN           Device name
SYSOUT          Device name
SYSLIST         Device name
SYSLST          Device name
STDOUT          Device name
PRINTER         Device name
SYSERR          Device name
STDERR          Device name
CONSOLE         Device name
C01             Feature name
C02             Feature name
C03             Feature name
C04             Feature name
C05             Feature name
C06             Feature name
C07             Feature name
C08             Feature name
C09             Feature name
C10             Feature name
C11             Feature name
C12             Feature name
CSP             Feature name
FORMFEED        Feature name
CALL-CONVENTION Feature name
SWITCH-0        Switch name
SWITCH-1        Switch name
SWITCH-2        Switch name
SWITCH-3        Switch name
SWITCH-4        Switch name
SWITCH-5        Switch name
SWITCH-6        Switch name
SWITCH-7        Switch name
SWITCH-8        Switch name
SWITCH-9        Switch name
SWITCH-10       Switch name
SWITCH-11       Switch name
SWITCH-12       Switch name
SWITCH-13       Switch name
SWITCH-14       Switch name
SWITCH-15       Switch name
SWITCH-16       Switch name
SWITCH-17       Switch name
SWITCH-18       Switch name
SWITCH-19       Switch name
SWITCH-20       Switch name
SWITCH-21       Switch name
```

```
SWITCH-22       Switch name
SWITCH-23       Switch name
SWITCH-24       Switch name
SWITCH-25       Switch name
SWITCH-26       Switch name
SWITCH-27       Switch name
SWITCH-28       Switch name
SWITCH-29       Switch name
SWITCH-30       Switch name
SWITCH-31       Switch name
SWITCH-32       Switch name
SWITCH-33       Switch name
SWITCH-34       Switch name
SWITCH-35       Switch name
SWITCH-36       Switch name

Extended mnemonic names (with -fsyntax-extension)
SW0             Switch name
SW1             Switch name
SW2             Switch name
SW3             Switch name
SW4             Switch name
SW5             Switch name
SW6             Switch name
SW7             Switch name
SW8             Switch name
SW9             Switch name
SW10            Switch name
SW11            Switch name
SW12            Switch name
SW13            Switch name
SW14            Switch name
SW15            Switch name
SWITCH 0        Switch name
SWITCH 1        Switch name
SWITCH 2        Switch name
SWITCH 3        Switch name
SWITCH 4        Switch name
SWITCH 5        Switch name
SWITCH 6        Switch name
SWITCH 7        Switch name
SWITCH 8        Switch name
SWITCH 9        Switch name
SWITCH 10       Switch name
SWITCH 11       Switch name
SWITCH 12       Switch name
SWITCH 13       Switch name
SWITCH 14       Switch name
SWITCH 15       Switch name
SWITCH 16       Switch name
SWITCH 17       Switch name
SWITCH 18       Switch name
```

```
SWITCH 19       Switch name
SWITCH 20       Switch name
SWITCH 21       Switch name
SWITCH 22       Switch name
SWITCH 23       Switch name
SWITCH 24       Switch name
SWITCH 25       Switch name
SWITCH 26       Switch name
SWITCH A        Switch name
SWITCH B        Switch name
SWITCH C        Switch name
SWITCH D        Switch name
SWITCH E        Switch name
SWITCH F        Switch name
SWITCH G        Switch name
SWITCH H        Switch name
SWITCH I        Switch name
SWITCH J        Switch name
SWITCH K        Switch name
SWITCH L        Switch name
SWITCH M        Switch name
SWITCH N        Switch name
SWITCH O        Switch name
SWITCH P        Switch name
SWITCH Q        Switch name
SWITCH R        Switch name
SWITCH S        Switch name
SWITCH T        Switch name
SWITCH U        Switch name
SWITCH V        Switch name
SWITCH W        Switch name
SWITCH X        Switch name
SWITCH Y        Switch name
SWITCH Z        Switch name
```

Appendix F Compiler Configuration

The following list was extracted from `config/default.conf`.

```
# Value: any string
name: "GnuCOBOL"

# Value: enum
standard-define                 0
#        CB_STD_OC = 0,
#        CB_STD_MF,
#        CB_STD_IBM,
#        CB_STD_MVS,
#        CB_STD_BS2000,
#        CB_STD_ACU,
#        CB_STD_85,
#        CB_STD_2002,
#        CB_STD_2014

# Value: int
tab-width:                      8
text-column:                    72
# Maximum word-length for COBOL words / Programmer defined words
# Be aware that GC checks the word length against COB_MAX_WORDLEN
# first (currently 61)
word-length:                    31

# Value: 'mf', 'ibm'
#
assign-clause:                  mf

# If yes, file names are resolved at run time using
# environment variables.
# For example, given ASSIGN TO "DATAFILE", the file name will be
#  1. the value of environment variable 'DD_DATAFILE' or
#  2. the value of environment variable 'dd_DATAFILE' or
#  3. the value of environment variable 'DATAFILE' or
#  4. the literal "DATAFILE"
# If no, the value of the assign clause is the file name.
#
filename-mapping:               yes

# Alternate formatting of numeric fields
pretty-display:                 yes

# Allow complex OCCURS DEPENDING ON
complex-odo:                    no

# Allow REDEFINES to other than last equal level number
indirect-redefines:             no
```

```
# Binary byte size - defines the allocated bytes according to PIC
# Value:         signed  unsigned  bytes
#                ------  --------  -----
# '2-4-8'         1 -  4    same        2
#                 5 -  9    same        4
#                10 - 18    same        8
#
# '1-2-4-8'       1 -  2    same        1
#                 3 -  4    same        2
#                 5 -  9    same        4
#                10 - 18    same        8
#
# '1--8'          1 -  2    1 -  2      1
#                 3 -  4    3 -  4      2
#                 5 -  6    5 -  7      3
#                 7 -  9    8 -  9      4
#                10 - 11   10 - 12      5
#                12 - 14   13 - 14      6
#                15 - 16   15 - 16      7
#                17 - 18   17 - 18      8
#
binary-size:                    1-2-4-8

# Numeric truncation according to ANSI
binary-truncate:                yes

# Binary byte order
# Value: 'native', 'big-endian'
binary-byteorder:               big-endian

# Allow larger REDEFINES items
larger-redefines-ok:            no

# Allow certain syntax variations (eg. REDEFINES position)
relaxed-syntax-check:           no

# Perform type OSVS - If yes, the exit point of any currently
# executing perform is recognized if reached.
perform-osvs:                   no

# If yes, linkage-section items remain allocated
# between invocations.
sticky-linkage:                 no

# If yes, allow non-matching level numbers
relax-level-hierarchy:          no

# If yes, allow reserved words from the 85 standard
cobol85-reserved:               no

# Allow Hex 'F' for NUMERIC test of signed PACKED DECIMAL field
hostsign:                       no
```

```
# If yes, set WITH UPDATE clause as default for ACCEPT dest-item,
# except if WITH NO UPDATE clause is used
accept-update:          no

# If yes, set WITH AUTO clause as default for ACCEPT dest-item,
# except if WITH TAB clause is used
accept-auto:            no

# not-reserved:
# Value: Word to be taken out of the reserved words list
# (case independent)
# Words that are in the (proposed) standard but may conflict

# Dialect features
# Value: 'ok', 'warning', 'archaic', 'obsolete', 'skip', 'ignore', 'error',
#        'unconformable'

alter-statement:                        obsolete
author-paragraph:                       obsolete
data-records-clause:                    obsolete
debugging-line:                         ok
eject-statement:                        skip
entry-statement:                        obsolete
goto-statement-without-name:            obsolete
label-records-clause:                   obsolete
memory-size-clause:                     obsolete
move-noninteger-to-alphanumeric:        error
multiple-file-tape-clause:              obsolete
next-sentence-phrase:                   archaic
odo-without-to:                         warning
padding-character-clause:               obsolete
section-segments:                       ignore
stop-literal-statement:                 obsolete
synchronized-clause:                    ok
top-level-occurs-clause:                ok
value-of-clause:                        obsolete
numeric-boolean:                        unconformable
acucobol-literals:                      unconformable
```

Appendix G cobcrun --help

```
Usage: /d/Programme/Entwicklung/GnuCOBOL/gnu-cobol-2.0_ultraclean2/bin/.libs/cobcrun [optio
  or:  /d/Programme/Entwicklung/GnuCOBOL/gnu-cobol-2.0_ultraclean2/bin/.libs/cobcrun option

COBOL driver program for GnuCOBOL modules

Options:
  -h, -help                   Display this help and exit
  -V, -version                Display cobcrun and runtime version and exit
  -i, -info                   Display runtime information (build/environment)
  -c <file>, -config=<file>   Set runtime configuration from <file>
  -r, -runtime-env            Display current runtime configuration
                              This will show all settings and how they were set.
                              Possible options are runtime configuration file,
                              environment (marked by 'env' when only set this way or
                              by 'Ovr' if this overrides the runtime setting) or default
```

Appendix H Runtime configuration

The following list was extracted from `config/runtime.cfg`.

H.1 General instructions

The initial runtime.cfg file is found in the `$COB_CONFIG_DIR/config` (`COB_CONFIG_DIR defaults to installdir/gnu-cobol`). The environment variable `COB_RUNTIME_CONFIG` may define a different runtime configuration file to read.
If settings are included in the runtime environment file multiple times then the last setting value is used, no warning occurs.
Settings via environment variables always take precedence over settings that are given in runtime configuration files. And the environment is checked after completing processing of the runtime configuration file(s)
All values set to string variables or environment variables are checked for `$envvar` and replacement is done at the time of the setting.
Any environment variable may be set with the directive `setenv` . Example: `setenv COB_LIBARAY_PATH $LD_LIBRARY_PATH`
Any environment variable may be unset with the directive `unsetenv` (one var per line). Example: `unsetenv COB_LIBRARY_PATH`
Runtime configuration files can include other files with the directive include. Example: `include my-runtime-configuration-file`
To include another configuration file only if it is present use the directive includeif. You can also use `$envvar` inside this. Example: `includeif $HOME/mygc.cfg`
If you want to reset a parameter to its default value use: `reset parametername`
Most runtime variables have boolean values, some are switches, some have string values, integer values and some are size values. The boolean values will be evaluated as following: to true: `1, Y, ON, YES, TRUE` (no matter of case) to false: `0, N, OFF` A 'size' value is an integer optionally followed by K, M, or G for kilo, mega or giga.
For convenience a parameter in the runtime.cfg file may be defined by using either the environment variable name or the parameter name. In most cases the environment variable name is the parameter name (in upper case) with the prefix `COB_` .

H.2 General environment

```
Environment name:  COB_DISABLE_WARNINGS
  Parameter name:  disable_warnings
         Purpose:  turn off runtime warning messages
            Type:  boolean
         Default:  false
         Example:  DISABLE_WARNINGS  TRUE

Environment name:  COB_ENV_MANGLE
  Parameter name:  env_mangle
```

```
         Purpose:  names checked in the environment would get non alphanumeric
                   change to '_'
            Type:  boolean
         Default:  false
         Example:  ENV_MANGLE  TRUE

Environment name:  COB_SET_TRACE
  Parameter name:  set_trace
         Purpose:  to enable to COBOL trace feature
            Type:  boolean
         Default:  false
         Example:  SET_TRACE  TRUE

Environment name:  COB_TRACE_FILE
  Parameter name:  trace_file
         Purpose:  to define where COBOL trace output should go
            Type:  string
         Default:  stderr
         Example:  TRACE_FILE  ${HOME}/mytrace.log
```

H.3 Call environment

```
Environment name:  COB_LIBRARY_PATH
  Parameter name:  library_path
         Purpose:  paths for dynamically-loadable modules
            Type:  string
            Note:  the default paths .:/installpath/extras are always
                   added to the given paths
         Example:  LIBRARY_PATH    /opt/myapp/test:/opt/myapp/production

Environment name:  COB_PRE_LOAD
  Parameter name:  pre_load
         Purpose:  modules that are loaded during startup, can be used
                   to CALL COBOL programs or C functions that are part
                   of a module library
            Type:  string
            Note:  the modules listed should NOT include extensions, the
                   runtime will use the right ones on the various platforms,
                   COB_LIBRARY_PATH is used to locate the modules
         Example:  PRE_LOAD      COBOL_function_library:external_c_library

Environment name:  COB_LOAD_CASE
  Parameter name:  load_case
         Purpose:  resolve ALL called program names to UPPER or LOWER case
            Type:  Only use  UPPER  or  LOWER
         Default:  if not set program names in CALL are case sensitive
         Example:  LOAD_CASE  UPPER
```

```
Environment name:  COB_PHYSICAL_CANCEL
  Parameter name:  physical_cancel
         Purpose:  physically unload a dynamically-loadable module on CANCEL,
                   this frees some RAM and allows the change of modules during
                   run-time but needs more time to resolve CALLs (both to
                   active and not-active programs)
           Alias:  default_cancel_mode, LOGICAL_CANCELS (0 = yes)
            Type:  boolean (evaluated for true only)
         Default:  false
         Example:  PHYSICAL_CANCEL  TRUE
```

H.4 File I/O

```
Environment name:  COB_VARSEQ_FORMAT
  Parameter name:  varseq_format
         Purpose:  declare format to be used for variable length sequential
                   files (different types and lengths preceding each record)
            Type:  0   means 2 byte record length (big-endian)
                   1   means 4 byte record length (big-endian)
                   2   means 4 byte record length (local machine int)
                   3   means 2 byte record length (local machine short)
         Default:  0
         Example:  VARSEQ_FORMAT 1

Environment name:  COB_FILE_PATH
  Parameter name:  file_path
         Purpose:  define default location where data files are stored
            Type:  file path directory
         Default:  .  (current directory)
         Example:  FILE_PATH ${HOME}/mydata

Environment name:  COB_LS_FIXED
  Parameter name:  ls_fixed
         Purpose:  Defines if LINE SEQUENTIAL files should be fixed length
                   (or variable, by removing trailing spaces)
           Alias:  STRIP_TRAILING_SPACES  (0 = yes)
            Type:  boolean
         Default:  false
         Example:  LS_FIXED TRUE

Environment name:  COB_LS_NULLS
  Parameter name:  ls_nulls
         Purpose:  Defines for LINE SEQUENTIAL files what to do with data
                   which is not DISPLAY type.  This could happen if a LINE
                   SEQUENTIAL record has COMP data fields in it.
            Type:  boolean
         Default:  false
            Note:  The TRUE setting will handle files that contain COMP data
```

```
                in a similar manner to the method used by Micro Focus COBOL
        Example:  LS_NULL = TRUE

Environment name:  COB_SYNC
  Parameter name:  sync
         Purpose:  Should the file be synced to disk after each write/update
            Type:  boolean
         Default:  false
         Example:  SYNC: TRUE

Environment name:  COB_SORT_MEMORY
  Parameter name:  sort_memory
         Purpose:  Defines how much RAM to assign for sorting data
            Type:  size  but must be more than 1M
         Default:  128M
         Example:  SORT_MEMORY 64M

Environment name:  COB_SORT_CHUNK
  Parameter name:  sort_chunk
         Purpose:  Defines how much RAM to assign for sorting data in chunks
            Type:  size  but must be within 128K and 16M
         Default:  256K
         Example:  SORT_CHUNK 1M
```

H.5 Screen I/O

```
Environment name:  COB_BELL
  Parameter name:  bell
         Purpose:  Defines how a request for the screen to beep is handled
            Type:  FLASH, SPEAKER, FALSE, BEEP
         Default:  BEEP
         Example:  BELL SPEAKER

Environment name:  COB_REDIRECT_DISPLAY
  Parameter name:  redirect_display
         Purpose:  Defines if DISPLAY output should be sent to 'stderr'
            Type:  boolean
         Default:  false
         Example:  redirect_display Yes

Environment name:  COB_SCREEN_ESC
  Parameter name:  screen_esc
         Purpose:  Enable handling of ESC key during ACCEPT
            Type:  boolean
         Default:  false
            Note:  is only evaluated if COB_SCREEN_EXCEPTIONS is active
         Example:  screen_esc Yes
```

```
Environment name:  COB_SCREEN_EXCEPTIONS
  Parameter name:  screen_exceptions
         Purpose:  enable exceptions for function keys during ACCEPT
            Type:  boolean
         Default:  false
         Example:  screen_exceptions Yes

Environment name:  COB_TIMEOUT_SCALE
  Parameter name:  timeout_scale
         Purpose:  specify translation in milliseconds for ACCEPT clauses
                   BEFORE TIME value / AFTER TIMEOUT
            Type:  integer
                   0 means 1000 (Micro Focus COBOL compatible), 1 means 100
                   (ACUCOBOL compatible), 2 means 10, 3 means 1
         Default:  0
         Example:  timeout_scale 3

Environment name:  COB_INSERT_MODE
  Parameter name:  insert_mode
         Purpose:  specify default insert mode for ACCEPT; 0=off, 1=on
         Default:  false
         Example:  insert_mode Y

Environment name:  COB_LEGACY
  Parameter name:  legacy
         Purpose:  keep behaviour of former runtime versions, currently only
                   for setting screen attributes for non input fields
            Type:  boolean
         Default:  not set
         Example:  legacy true

Note: If you want to slightly speed up a program's startup time, remove all
      of the comments from the actual real file that is processed
```

Index