GnuCOBOL 3.2 Build Guide
for MSYS2 64-bit – Minimalist GNU for Windows
GCC version (MinGW64) "13.1.0"

cobc (GnuCOBOL) 3.2 (2023-07-28)

Written by Keisuke Nishida, Roger While, Ron Norman, Simon Sobisch, Edward Hart.

This document was prepared by: Arnold J. Trembley (arnold.trembley@att.net),
based on extensive work by Chuck Haatvedt (chuck.haatvedt@gmail.com),
and last updated Wednesday, 6 December 2023, for GnuCOBOL 3.2 Final Release.

Special thanks go to **Chuck Haatvedt**, who developed the shell scripts and .cmd files for this approach. **Simon Sobisch**, the GnuCOBOL project leader, also provided Chuck with assistance and made changes to the GnuCOBOL source packages to facilitate this MSYS2 build process. I am deeply indebted to both Chuck and Simon for their work.

These scripts build GnuCOBOL with the following components at the time these scripts were created:

```
Sequential file handler          : built-in
Indexed file handler             : BDB, version 18.1.40  ==OR== 6.0.19 ==OR==
                                 : VBISAM 2.1.1 (patched by Chuck Haatvedt)  ==OR==
                                 : NO ISAM support
Mathematical library             : GMP, version 6.3.0
XML library                      : libxml2, version 2.12.1
JSON library                     : cJSON, version 1.7.15
Extended screen I/O              : pdcurses, version 4.4.0 (CHTYPE=64, WIDE=0, UTF8=0)
```

Note that if you would like to build with other configurations, please contact either Chuck or me and we will attempt to assist you. This process can also be adapted to build GnuCOBOL 4.x.

STEP01 – Disable Anti-Virus (optional)

Close all web browsers and disable real-time Anti-Virus scanning. This may make the build process go more smoothly, but you cannot disconnect from the internet because the scripts download components from multiple trusted sources. The Microsoft Windows default AV scanning generally does not cause delays in loading and compiling. So if you are very cautious about security, you may want to leave your non-Microsoft Anti-Virus enabled. Just be aware that it may slow down the process due to scanning very large download files.

STEP02 – Download and install MSYS2/MinGW64

If you had previously installed MSYS2/MinGW, then you can skip this step and go to **Step 03 "Update the MSYS2 system"**.

Otherwise, go to https://www.msys2.org/ and download "msys2-x86_64-*.exe"
Look for version "msys2-x86_64-20231026.exe" or newer.

Install "msys2-x86_64-*.exe" into the default "**C:\msys64"** folder, and accept the default start menu shortcut folder "MSYS2 64bit". Follow the installation instructions from the msys2.org web page through their step 4.

When you see the MSYS2 UCRT64 terminal shell environment, type "exit".

You can actually install MSYS2 to ANY folder on ANY drive letter, but I recommend using the suggested defaults. If you used a path different from "**C:\msys64"** then you will need to edit some of the scripts because several filenames depend on specific filename paths.

STEP03 – Update the MSYS2 system

Whether you have just installed MSYS2 or you have a previous installation, you should open the MSYS2 MSYS window and request the latest MSYS2 updates using the "pacman -Syuu" command.

Remember to run "pacman -Syuu" TWICE, or as many times as needed until there are no more updates to apply. Then type "exit" to leave MSYS2.

STEP04 – Install the GnuCOBOL MSYS2 build kit

Create a folder named "**C:\msys64prep"** and copy the "MSYS2-Build-Kit-v08.7z" file into it. That file should be downloaded from the following link:
https://www.arnoldtrembley.com/MSYS2-Build-Kit-v09.7z

Then, using the 7-Zip file manager, extract all the files into that folder, so that you end up with a folder that looks something like this:

```
 Directory of C:\msys64prep

12/06/2023  12:07 AM    <DIR>          .
12/04/2023  01:30 AM    <DIR>          ..
11/17/2023  12:46 AM    <DIR>          docs
11/16/2023  02:16 AM    <DIR>          Extra-files
12/05/2023  11:08 PM             1,028 AMOVE_SCRIPTS.CMD
12/05/2023  11:09 PM            23,137 build-prod-x32.sh
12/05/2023  11:10 PM            23,203 build-prod-x64.sh
12/05/2023  11:11 PM            23,200 build-test-x32.sh
12/05/2023  11:12 PM            23,299 build-test-x64.sh
11/16/2023  02:50 AM               454 build-toolchains.cmd
11/16/2023  02:50 AM             2,889 build-toolchains.sh
11/16/2023  02:51 AM             1,286 build-x32.cmd
11/16/2023  02:52 AM             1,347 build-x64.cmd
12/06/2023  12:23 AM            19,684 GnuCOBOL-3.2-MSYS2-X64-Build-Guide-v1.5.docx
12/06/2023  12:23 AM           116,272 GnuCOBOL-3.2-MSYS2-X64-Build-Guide-v1.5.pdf
              11 File(s)        235,799 bytes
```

The "docs" subfolder contains PDF manuals for GnuCOBOL, GCSORT, and the GNU Debugger. The "Extra-files" folder contains additional files to be packaged with the GnuCOBOL binary (assuming you are building the compiler for distribution).

STEP05 – Customize your build scripts (Optional)

If you are using path-names different from "**C:\msys64**" or "**C:\msys64prep**" or if you want to bypass using GnuCOBOL 3.3 patches, this step covers those changes.

Also, you can control the speed of your GnuCOBOL MSYS2 builds by adjusting the "cpu_count" in the build scripts. The "cpu_count" defaults to 4 and can be seen in the following four build scripts:

```
export cpu_count=4
```

```
11/14/2023  11:07 PM            22,285 build-prod-x32.sh
11/14/2023  11:09 PM            23,080 build-prod-x64.sh
11/14/2023  11:18 PM            22,896 build-test-x32.sh
11/14/2023  11:26 PM            23,003 build-test-x64.sh
```

You can speed up your GnuCOBOL builds by changing "export cpu_count=4" to a higher number, if you have enough processors in your CPU.

If you expect to build 32-bit and 64-bit compilers concurrently, and your Windows build machine has 16 threads, you could change those scripts to "export cpu_count=8". Your builds should run a bit faster.

If you expect to build either a 32-bit or a 64-bit compiler (one at a time) and your Windows build machine has 10 threads, you can change those scripts to "export cpu_count=10". Your builds should run faster than with "cpu_count=4".

Sometimes, there can be conflicts across CPU threads that may cause the build to fail. For that reason you might want to change to "export cpu_count=2" or even "cpu_count=1". That will reduce the risk of multitasking conflicts, but your builds will run slower.

Whatever you choose, you should avoid setting the cpu_count greater than the maximum number of threads that your CPU supports.

CORRECTING PATH-NAMES

The AMOVE_SCRIPTS.CMD file has code in it like this:

```
:PROD
echo Copying the Production scripts with performance optimizations
rem timeout /t 3
copy C:\msys64prep\build-toolchains.sh    C:\msys64\build-toolchains.sh
copy C:\msys64prep\build-prod-x32.sh      C:\msys64\mingw32\bin\build-x32.sh
copy C:\msys64prep\build-prod-x64.sh      C:\msys64\mingw64\bin\build-x64.sh
```

If you installed MSYS2 into a path with a different name from "**C:\msys64”** then you will need to change all occurrences to the correct path in AMOVE_SCRIPTS.CMD before running it.

Similarly, if you installed used a different folder name from “**C:\msys64prev”** then you will need to change all occurrences of that to the correct name before running AMOVE_SCRIPTS.CMD.

STEP06 – RUN AMOVE_SCRIPTS.CMD

Next, while still in the “**C:\msys64prep”** folder of your CMD.EXE or Windows Terminal CMD shell, run the AMOVE_SCRIPTS.CMD file.

You should see a screen that looks like this:

```
...............................................
PRESS 1 or  2
...............................................

1 – Copy BUILD PROD scripts
2 – Copy BUILD DEBUG scripts

Type 1, 2 press ENTER:
```

If you plan to test the generated GnuCOBOL compiler for new features or code defects, then you would select option 2 “DEBUG”, to build GnuCOBOL with extra compiler debugging support.

But if you are building a GnuCOBOL compiler for COBOL development you should select 1 “PROD”. The PROD compiler will still include extensive debugging tools for using the Gnu Debugger, COBOLWORX, and python for debugging your COBOL programs.

You should see the following results:

```
Type 1, 2 press ENTER:1
Copying the Production scripts with performance optimizations
        1 file(s) copied.
        1 file(s) copied.
        1 file(s) copied.
'All Done'

C:\msys64prep>
```

STEP07 – Run the build-toolchains.cmd script

Open a CMD.EXE or Windows Terminal CMD shell in the "**C:\msys64prep**" folder, and run the build-toolchains.cmd script.

This will run an MSYS2 shell script to update the toolchain, which include the components that will be used to build the GnuCOBOL compiler, such as GCC (The Gnu Compiler Collection, GMP (Math Package), Python, and the Gnu Debugger. This step takes about 10 minutes or so.

If you have recently installed or updated your MSYS2 environment you should always run the "build-toolchains.cmd" script. But if you are rebuilding a GnuCOBOL compiler in the same day, you may not need to update the toolchain again.

If there are problems with "build-toolchains", you can review the log file named "build-toolchains.txt" found in your "**C:\msys64prep**" folder.

STEP08 – Run the "build-x64.cmd" script

Next, while still in the "**C:\msys64prep**" folder, run the "build-x64.cmd" script. You should see a screen that looks like this:

```
...............................................
PRESS 1, or 2
...............................................

1 – DO a full build of GNUCOBOL
2 – Only CONFIGURE / MAKE / MAKE INSTALL of GNUCOBOL

Type 1, 2 then press ENTER:
```

Your first choice would be to "DO a full build of GNUCOBOL". If you have recently done a build but the GnuCOBOL source code has changed (or the GCSORT source code), then you can save time by selecting option 2, to only rebuild the GnuCOBOL component.

The next screen you see should look like this:

```
...............................................
PRESS 1, 2, 3 OR 4 to select your task, or 5 to EXIT.
...............................................

1 – Use ISAM type of "--without-db"
2 – Use ISAM type of "--with-db"
3 - Use ISAM type of "--with-vbisam"
4 - Use ISAM type of "--with-db-old"
5 - EXIT

Type 1, 2, 3, 4 or 5 then press ENTER:
```

Here you would select the ISAM option (Indexed Sequential Access Method) for your GnuCOBOL build. Note that option 4 would create a GnuCOBOL compiler with an older version of Berkeley DataBase software (6.0.19) which has the less-restrictive SleepyCat license. Going forward, the preferred choice will be "—with-vbisam", because VBISAM (or V-ISAM) has an open-source GPL license.

Depending on your cpu_count, the speed of your processor, and the type of build you selected, it can take anywhere from 20 minutes to an hour to complete the build.

If there are problems with "build-x64", you can review the log file named "build-x64.txt" found in your "**C:\msys64prep"** folder.

The generated compiler will be found in a file named something like **GNUCOBOL-X64-VBISAM.7z** (depending on the ISAM choice), and will be found in the following folder:
C:\msys64\home\(user)\x64\gnucobol-trunk\GNUCOBOL-X64-VBISAM.7z

The compiler file names that could be generated are:

GNUCOBOL-X64-VBISAM.7z
GNUCOBOL-X64-BDB.7z
GNUCOBOL-X64-NODB.7z
GNUCOBOL-X64-OLD-BDB.7z

The 7-Zip archive can be decompressed into the folder of your choice, and it contains a complete, ready-to-run GnuCOBOL compiler.

STEP09 – Run the “build-x32.cmd” script

Next, while still in the “**C:\msys64prep”** folder, run the “build-x32.cmd” script.

The screens look very similar to the “build-x64” screens from the previous step, and you have the same options for doing a full build of the compiler, rebuilding just the COBOL component, and your ISAM support option.

You should see a screen that looks like this:

```
...............................................
PRESS 1, or 2
...............................................

1 – DO a full build of GNUCOBOL
2 – Only CONFIGURE / MAKE / MAKE INSTALL of GNUCOBOL

Type 1, 2 then press ENTER:
```

Your first choice would be to “DO a full build of GNUCOBOL”. If you have recently done a build but the GnuCOBOL source code has changed (or the GCSORT source code), then you can save time by selecting option 2, to only rebuild the GnuCOBOL component.

The next screen you see should look like this:

```
...............................................
PRESS 1, 2, 3 OR 4 to select your task, or 5 to EXIT.
...............................................

1 – Use ISAM type of "--without-db"
2 – Use ISAM type of "--with-db"
3 - Use ISAM type of "--with-vbisam"
4 - Use ISAM type of "--with-db-old"
5 - EXIT

Type 1, 2, 3, 4 or 5 then press ENTER:
```

Here you would select the ISAM option (Indexed Sequential Access Method) for your GnuCOBOL build. Note that option 4 would create a GnuCOBOL compiler with an older version of Berkeley DataBase software (6.0.19) which has the less-restrictive SleepyCat license. Going forward, the preferred choice will be “—with-vbisam”, because VBISAM (or V-ISAM) has an open-source GPL license.

Depending on your cpu_count, the speed of your processor, and the type of build you selected, it can take anywhere from 20 minutes to an hour to complete the build.

If there are problems with “build-x32”, you can review the log file named “build-x32.txt” found in your “**C:\msys64prep”** folder.

The generated compiler will be found in a file named something like **GNUCOBOL-X32-VBISAM.7z** (depending on the ISAM choice), and will be found in the following folder:
C:\msys64\home\(user)\x32\gnucobol-trunk\GNUCOBOL-X32-VBISAM.7z

The compiler file names that could be generated are:

GNUCOBOL-X32-VBISAM.7z
GNUCOBOL-X32-BDB.7z
GNUCOBOL-X32-NODB.7z
GNUCOBOL-X32-OLD-BDB.7z

The 7-Zip archive can be decompressed into the folder of your choice, and it contains a complete, ready-to-run GnuCOBOL compiler.

STEP10 – Preparing the Distribution Binary

Note that the generated GnuCOBOL compiler (whether 32-bit or 64-bit) is contained in a 7-Zip compressed archive, so you will need to use 7-Zip to decompress it and create a GnuCOBOL folder.

Once you have a folder with a working compiler, you can make some enhancements using the additional contents of the “msys64prep” folder:

1. Delete the “BUGS.txt” file, unless it contains useful information.
2. Delete the “gnucobolpg.pdf” file (but NOT the “gnucobol.pdf” file).
3. Copy the entire “\msys64prep\docs” folder into a new “\docs” subfolder. It should be at the same hierarchy level as the “\bin” folder.
4. Copy all the files in the “\msys64prep\extra-files” folder into the compiler folder. They should be at the hierarchy level just above the “\bin” subfolder.

After you have finished that, you can compress the compiler folder using open source 7-Zip, available from: https://7-zip.org/

As of 16 November 2023, the current stable version of 7-Zip and 7-Zip file Manager is 23.01.

CONTENTS OF THE \MSYS64PREP\DOCS FOLDER

```
 Directory of C:\msys64prep\docs

11/17/2023  12:46 AM    <DIR>          .
12/05/2023  11:14 PM    <DIR>          ..
11/15/2023  06:04 AM         1,087,624 GCSORT_Manual.pdf
05/22/2023  09:01 PM         2,997,619 gdb.pdf
05/22/2023  09:29 PM             2,995 gdb-help.txt
05/22/2023  09:02 PM           118,821 gdb-refcard.pdf
12/05/2023  06:09 PM         2,053,217 GnuCOBOL 3.2 Programmer's Guide.pdf
12/05/2023  06:09 PM         2,050,326 GnuCOBOL 3.2 Programmer's Reference.pdf
12/05/2023  06:10 PM           265,372 GnuCOBOL 3.2 Quick Reference.pdf
12/05/2023  06:10 PM           436,621 GnuCOBOL 3.2 Sample Programs.pdf
               8 File(s)      9,012,595 bytes
```

CONTENTS OF THE \MSYS64PREP\EXTRA-FILES FOLDER

```
 Directory of C:\msys64prep\Extra-files

11/16/2023  02:16 AM    <DIR>          .
12/05/2023  11:14 PM    <DIR>          ..
11/15/2023  11:42 PM           315,335 cobc-help.txt
02/11/2023  01:07 AM            22,810 gcsort-help.txt
12/28/2022  10:06 PM             3,077 gcx.cmd
11/16/2023  12:03 AM             4,718 STARTHERE.txt
12/28/2022  10:01 PM             3,699 testfunc.cob
11/17/2023  03:06 AM             2,611 TestGC.cmd
10/24/2022  07:53 PM               871 VScobc.cmd
10/24/2022  07:53 PM               868 VScobcrun.cmd
               8 File(s)        353,989 bytes
```

The “cobc-help.txt” file can be generated with the following command (after running set_env.cmd to set GnuCOBOL Environment Variables):

```
cobc -vvh > cobc-help.txt
```

GNUCOBOL BUILD TEST RESULTS

Your results may vary depending on ISAM options, but should look something like this:

From build-x32.txt or build-x64.txt:

```
## ------------- ##
## Test results. ##
## ------------- ##

ERROR: 1269 tests were run,
5 passed unexpectedly,
33 failed (26 expected failures).
13 tests were skipped.

## ------------------------ ##
## Summary of the failures. ##
## ------------------------ ##
Failed tests:
GnuCOBOL 3.2 test suite: GnuCOBOL Tests test groups:

 NUM: FILE-NAME:LINE     TEST-GROUP-NAME
      KEYWORDS

   8: used_binaries.at:374 temporary path invalid
      cobc runmisc
 808: run_misc.at:10199  stack and dump feature
      stacktrace configuration cob_stacktrace cob_dump_file call
 809: run_misc.at:11016  dump feature with NULL address
      stack stacktrace
 854: run_file.at:406    OUTPUT on INDEXED file to missing directory
      runfile open assign
 921: run_file.at:6957   INDEXED partial keys
      runfile
 930: run_file.at:9665   EXTFH: operation OP_GETINFO / QUERY-FILE
      runfile extfh getinfo query
 931: run_file.at:9847   EXTFH: changing record address
      runfile extfh

Skipped tests:
GnuCOBOL 3.2 test suite: GnuCOBOL Tests test groups:

 NUM: FILE-NAME:LINE     TEST-GROUP-NAME
      KEYWORDS

 470: syn_literals.at:1379 ACUCOBOL 32bit literal size
      extensions literals
 535: run_fundamental.at:1417 function with variable-length RETURNING item
      fundamental udf
 654: run_accept.at:367  ACCEPT OMITTED (SCREEN)
```

```
      extensions
 797: run_misc.at:7169   ON EXCEPTION clause of DISPLAY
      runmisc exceptions screen
 798: run_misc.at:7194   EC-SCREEN-LINE-NUMBER and -STARTING-COLUMN
      runmisc exceptions screen
 799: run_misc.at:7235   LINE/COLUMN 0 exceptions
      line column runmisc exceptions extensions screen
 844: run_misc.at:14197  runtime check: write to internal storage (2)
      runmisc call bounds exceptions
 922: run_file.at:7104   INDEXED undeclared keys
      runfile
 935: run_file.at:10869  EXTFH: auto-conversion FCD2 <-> FCD3 on 32bit
      runfile extfh
 1110: run_functions.at:4594 UDF with recursion
      functions local-storage
 1181: run_extensions.at:4137 System routine CBL_GC_FORK
      extensions c$getpid
 1182: run_extensions.at:4205 System routine CBL_GC_WAITPID
      extensions cbl_gc_fork
 1188: run_extensions.at:4788 System routine x'91' function NN
      extensions

Unexpected passes:
GnuCOBOL 3.2 test suite: GnuCOBOL Tests test groups:

 NUM: FILE-NAME:LINE     TEST-GROUP-NAME
      KEYWORDS

 912: run_file.at:6201   INDEXED file with LOCK MODE EXCLUSIVE
      runfile
 913: run_file.at:6280   INDEXED file with OPEN WITH LOCK
      runfile
 914: run_file.at:6358   INDEXED file with SHARING NO
      runfile
 915: run_file.at:6437   INDEXED file with SHARING READ ONLY
      runfile
 916: run_file.at:6523   INDEXED file with blocked lock
      runfile
```

NOTE: VBISAM typically has the “unexpected pass” tests.

From “NIST-summary.log” in the generated compiler folder:

(These are the NIST COBOL-85 Test Suite results)

```
------ Directory Information -------   --- Total Tests Information ---
Module Programs Executed Error Crash   Pass Fail Deleted Inspect Total
------ -------- -------- ----- -----  ----- ---- ------- ------- -----
NC           95       95     0     0   4393    0       4       1  4398
SM           17       17     0     0    291    0       2       1   294
IC           25       25     0     0    246    0       4       0   250
SQ           85       85     0     0    518    0       0      89   607
RL           35       35     0     0   1827    0       5       0  1832
ST           40       40     0     0    288    0       0       0   288
SG           13       13     0     0    310    0       0       0   310
OB            7        7     0     0     39    0       0       0    39
IF           45       45     0     0    733    0       0       0   733
RW            6        6     0     0     40    0       0       0    40
DB           16       16     0     0    418    0       4      27   449
IX           42       42     0     0    507    0       1       0   508
------ -------- -------- ----- -----  ----- ---- ------- ------- -----
Total       426      426     0     0   9610    0      20     118  9748
```

```
C:\GNUCOBOL-X64-VBISAM> cobc --version
cobc (GnuCOBOL) 3.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Keisuke Nishida, Roger While, Ron Norman, Simon Sobisch, Edward Hart
Built     Nov 15 2023 18:59:56
Packaged  Jul 28 2023 16:58:47 UTC
C version (MinGW) "13.2.0"
```

As of 11 August 2023, the MSYS2 64-bit and 32-bit GnuCOBOL 3.2 Final binaries can be downloaded from the following addresses:

https://www.arnoldtrembley.com/GC32M-BDB-x64.7z

https://www.arnoldtrembley.com/GC32M-BDB-x32.7z

Due to a security restriction from my web hosting service I cannot directly host ".exe" files. So the new files have been renamed with ".7z" as their file extension. After downloading they can be opened and extracted using 7-Zip, or the windows file extensions can be renamed from ".7z" to ".exe", allowing the files to be used as self-extracting archives. The self-extracting file will ask you to supply a path and folder name for the compiler. It can also be installed to a drive other than your C: drive.

7-Zip is open source software available from http://www.7-zip.org/

In the future, new binaries will be added to the following page:

http://www.arnoldtrembley.com/GnuCOBOL.htm

For additional assistance, please feel free to join the GnuCOBOL Forums on SourceForge:

https://sourceforge.net/p/open-cobol/discussion/

=====end=====