zCOBOL User Guide

v1.5.06

Automated Software Tools Corporation.

Getting Started

So you want to try compiling and executing some COBOL programs on Windows using zCOBOL. First go to www.z390.org and download and install z390 version v1.5.06 or later. Also install J2SE java runtime version 6 update 31 which is the latest version which zCOBOL was regression tested on. Both installs use InstallShield for Windows. There are also file image install options available for Linux users. Be sure you uninstall any old versions of J2SE such as 1.4.2 or
5.0 as they may conflict with the later 6.0+ required version.

Now you should have a z390 desktop icon which you can double click to start the z390 GUI Interface. And now you can compile, link, and execute your first zCOBOL program by entering the following zCOBOL command:

```
zc390clg zcobol\demo\HELLO
```

The above command uses the zCOBOL to HLASM compile, link, and execute command to compile the COBOL hello world program zcobol\demo\HELLO.CBL into an executable HLASM compatible assembler program zcobol\demo\HELLO.MLC which is assembled using zCOBOL macro libraries zcobol+zcobol\z390 and linked into z390 executable load module HELLO.390 which is then executed to generate WTO display of "Hello World" on the GUI display log and on the console log file HELLO.LOG.

To run all the zCOBOL demo programs, you can enter the command zcrt390 which will compile and execute them. You can then view the log file for each demo program to see the output produced. For more examples, you can also run all the zCOBOL regression tests using the command ZCRTTEST and look at the source code and generated output.

Running your program

You can compile, link, and execute a COBOL program in any directory by specifying the path and name of the program in the ZC390CLG command. The source COBOL program must be in ASCII format and have the suffix CBL. You can use the compiler command ZC390C to compile to relocatable object form, and the compiler command ZC390CL to compile and link to 390 load module form with statically linked z390 and/or zCOBOL programs included. Once the programs have been successfully linked, then you can use the z390 EXEC command to execute a load module. For more about all the options available for z390 executable programs visit www.z390.org and read the z390 user guide.

zCOBOL Demo Programs in zcobol\demo zCOBOL Regression Tests in zcobol\test

HELLO	**Display "Hello World" and STOP RUN**
DATETIME	**ACCEPT current date and time and display month, day of week, and year**
COPYFILE	**Read line sequential ASCII file and copy it to an output line sequential file**
POWERS	**Calculate and display powers of 2 up to 2**31**

TESTADD1	**Test 225 combinations of ADD**
TESTADD2	**Test 225 combinations of ADD with different implied decimals**
TESTASM4.MLC	**Assembler module statically linked with TESTCAL3.CBL**
TESTBFP1	**Test Binary Floating Point support**
TESTCAL1	**CALL TESTCAL2 statically linked**
TESTCAL2	**CALL TESTCAL3 dynamically**
TESTCAL3	**Dynamically loaded zcobol module which calls statically linked TESTASM4 assembler routine**
TESTCMP1	**ADD, SUBTRACT, MULTIPLY, and DIVIDE all formats**
TESTCMP2	**Test ADD, SUBTRACT, MULTIPLY, and DIVIDE**
TESTCMP3	**Test COMPUTE with implied decimal points for data type F, G, H, P, Q, and Z.**
TESTCMP4	**Test COMPUTE with literals and different numberic values and implied decimals.**
TESTCMP5	**Test 225 combinations of COMPUTE**
TESTCMP6	**Test 225 combinations of COMPUTE with different implied decimals**
TESTCPY1	**COPY**
TESTCPY2	**nested COPY**
TESTDFP1	Test Decimal Floating Point)DFP) support
TESTDIV1	**Test 225 combinations of DIVIDE**
TESTDIV2	**Test 225 combinations of DIVIDE with different implied decimals**
TESTDSP1	**DISPLAY all formats**

TESTFIL1	Test file access
TESTFIL2	Test file access
TESTFUN1	**ACCEPT, TRANSFORM, NUMERIC, etc.**
TESTGO1	**GO TO DEPENDING ON**
TESTHFP1	Test Hexidecimal Floating Point (HFP) support
TESTIF1	**Compound IF requiring use of intermediate T/F flags**
TESTIF2	Test IF with omitted operands such as IF A = B OR C
TESTIF3	Test IF with parenthesis
TESTINT1	Test integer data types H, F, G, Q, P, and Z
TESTISP1	**INSPECT TALLYING, REPLACING, TRANSFORMING**
TESTMOV1	**MOVE all formats**
TESTMOV2	Test alignment for non-floating point moves
TESTMOV3	Test scaling for implied decimal for non-floating point moves
TESTMPY1	**Test 225 combinations of MULTIPLY**
TESTMPY2	**Test 225 combinations of MULTIPLY with different implied decimals**
TESTPM1	**PERFORM VARYING and PERFORM TIMES**
TESTPM2	Test PERFORM with duplicate paragraph names in different sections
TESTPM3	Test reading file using nested PERFORM VARYING
TESTRMD1	Test move reference modification of the form MOVE F1(var1+lit1:len1) TO F2(var2+lit2:len2)
TESTSIX1	**Test multiple subscripts**

TESTSIX2	Test SET and INDEXED form of subscripting
TESTSUB1	**Test 225 combinations of SUBTRACT**
TESTSUB2	**Test 225 combinations of SUBTRACT with different implied decimals**
TESTTRC1	**TRUNC**
TESTTRC2	**NOTRUNC with ONSIZE**
TESTTRC3	**TRUNC and NOR64 to test use of DXR versus DSG**
TESTWS1	**Working storage REDEFINE and OCCURS with padding**

zCOBOL Compiler Errors

zCOBOL is an open source project and is expected to continue to evolve for many years to come. There are still a number of missing functions in zCOBOL, and until such time as zCOBOL successfully passes the NIST ANSI COBOL 1985 Test Suite, zCOBOL should be considered to still be in a beta test state. Once zCOBOL gets beyond the NIST tests, zCOBOL will continue to add extensions and enhancements based on user requests including COBOL 2002 standard features along with other requests to support different dialect options provided by IBM, Micro Focus, Open COBOL, Veryant, and other COBOL compiler providers. The current list of pending fixes and enhancements for upcoming z390 and zCOBOL are here. You can use this online form to submit new fix and enhancement requests. All users of zCOBOL are encouraged to join the zcobol user group where users can ask questions and get answers and suggestions from developers and other users. COBOL, Java, C, and assembler developers interested in joining the zCOBOL open source project and helping with the development are welcome.

Debugging your program

Once you have successfully compiled a COBOL program into a z390 load module, you can run it with the command EXEC filename. If the program aborts or fails to produce the expected results, the next step is to debug the problem. The zCOBOL option TRACE can be specified to generate a WTO display of the name of each COBOL paragraph when it is entered. Sometimes that along with listing of the program is enough to figure out why the program did not work.

If it is necessary to examine the generated HLASM compatible assembler code, there are several steps that can be taken:

- First the assembly listing with suffix PRN produced by the zCOBOL compiler command can be examined to see if the generated assembler instructions perform the correct operation specified in the COBOL statement which precedes the generated code as a comment statement. See zcobol\demo\HELLO.PRN as an example. If the generated code is simply wrong, submit an RPI to get it fixed, and if you choose, you can assist by going to the code generation macro, changing it, testing it, and submitting a suggested fix along with the RPI.
- Next an execution trace of every assembler instruction executed along with the data values associated with each instruction can be produced by adding the option TRACE(E) which results in file with TRE suffix. If you also specify TRACE option, the WTO for every paragraph will also appear in the TRE trace file which can be handy for finding the start of code in a particular paragraph. For example, if you run the command ZC390CLG zcobol\demo \POWERS TRACE(E) then you can view the resulting executing trace file zcobol\demo\POWERS.TRE as well as the log file zcobol\demo\POWERS.LOG.
- If the execution trace fails to pinpoint the problem, another option is to include debug test and display statements in the program to further isolate where the problem is occurring.
- If it appears that a specific instruction is not performing the expected operation, the z390 pz390.java source code for that instruction can be examined and if necessary, eclipse can be used to step through each java instruction within the z390 instruction in question. But that's complicated so first you might want to create a cut down sample program with the problem and post it on the zcobol user group email list to get a quicker resolution to the problem.
- If you have suggestions on how to improve and/or extend this initial list of debugging aides, please post them on the user group or send an email to Don Higgins.

Trademarks

IBM, CICS, HLASM, MVS, OS/390, VSAM, z9, z10, and z/OS are registered trademarks of International Business Machines Corporation

Credits

Author : Don Higgins
Formatting : Walter Petras
Z390 version :