F-VFILES Documentation

The purpose of this contribution is to provide the identical functionality of these CBL_ functions provided by the Fujitsu and Microfocus compilers. Note that this has only been tested on the MSYS2 MINGW 32 bit version of GNUCOBOL and the 32 bit Fujitsu COBOL v5.0 compiler.

Note that the version which I have installed is Version 5.0 which is quite old. So you need to test the created DLL to ensure that it is working correctly.

FUJITSU CBL Subroutines User's Guide

==> this link is for a newer version of the Fujitsu Compiler Documentation but the manual for the older version is not available online.

This first group of entry points provide the same functionality as the Fujitsu CBL_ functions.

CBL_SPLIT_FILENAME
CBL_OPEN_VFILE
CBL_READ_VFILE
CBL_WRITE_VFILE
CBL_CLOSE_VFILE

The following are extra functions which are not found in Fujitsu. I found them useful for testing.

CBL_VFILES ==> if the CBL_VFILES.DLL is in the current directory then when called from a GNUCOBOL program this provides addressability without needing to use any COB_ environment variables.

CBL_MEMORY_STATUS ==> this returns the amount of memory used by the your program.

CBL_CPU_CYCLES ==> this returns the number of cpu clock cycles. I found it useful for performance testing.

CBL_GET_HEAP_POINTER ==> this is a diagnostic function to allow you to access the internal heap area structures from your COBOL program. Each VFILE consists of a HEAP entry and multiple DATA SEGMENTS implemented as a double linked list. So don’t use this in any production environment. Modifying any HEAP or DATA SEGMENT directly in your code will probably result in undefined behavior, so don’t do it.

CBL_START_TIME ==> store the current time internally for use by the CBL_TIME_DIFF function. So no parameters used for this call.

CBL_TIME_DIFF ==> this returns the elapsed time since the previous CBL_START_TIME was called.

DUMP_HEX ==> similar to the hex-of function in GNUCOBOL. Pass any address (CALL BY REFERENCE) followed by a LENGTH (CALL BY VALUE).

The following ENVIRONMENT VARIABLES are used by the CBL_VFILES.DLL

Note when called from a GNUCOBOL program, these environment variables can be set by the COBOL program and read by the CBL_VFILES.DLL However if the calling program is a FUJITSU cobol program then the environment variables set in the COBOL program are NOT visible in the CBL_VFILES.DLL, they must be set in the Windows console before starting the FUJITSU cobol program.

set CBL_DIAGNOSTIC=ON ==> if this environment variable is present with a value of ON, then diagnostic information will be written to the file VFILE-DIAGNOSTIC.LOG in the current directory.

set CBL_MEM_ALLOC_INIT=ON ==> if this environment variable is present with a value of ON, then the data segments will be initialized to binary zeros when allocated, otherwise they will NOT be initialized.

set CBL_MEM_ALLOC_KB=nnnn ==> if this environment variable is present then the numeric value between 16 and 1024 will be the size of the data segments in Kilobytes when they are allocated. Any value entered will be rounded to a multiple of 4. Any value out of range will be set to either 16 or 1024 KB. If not present or not numeric, then the default allocation size will be 64 KB.

Note that the allocation size can impact performance especially when using large virtual files. Also the amount of memory used is relative to the (record number - 1) * record size as this is the formula used to find the displacement in memory…

The maximum number of virtual files is 512 at any point in time.

This contribution consists of 6 files as follows, plus some sample cobol programs for testing.

- cbl_vfiles.c ==> this is the actual source for used to create the DLL's
- compile-to-dll.cmd ==> this is a Windows command file to compile to the CWH_FILES.DLL with the standard CBL_ prefixes for the entry point names.
- compile-alt-name-to-dll.cmd ==> this is a Windows command file to compile to the CWH_FILES.DLL with the entry point names changed to use a CWH_ prefix instead of the default CBL_ prefix. It is provided to help avoid name collision if needed. Otherwise the functionality is identical to the default version of the DLL.
- compile-debug.cmd ==> this is the same as the default version above only compiled with a -ggdb3 option to allow for debugging in GDB.
- fujitsu-cbl-entryfile.txt ==> this is the entry file to allow a Fujitsu COBOL program to dynamically call the CBL functions shipped with your Fujitsu compiler. Note that you have to set the environment variable @CBR_ENTRYFILE= entry information file-name and use the @DLOAD compiler option. (note this file uses absolute full path name so it will need to be modified to that used by your Fujitsu installation)
- FUJITSU_ENTRY_FILE.TXT ==> same functionality as above but to use the DLL built above (note this file uses relative path names so it must be in the current directory unless you edit it to use absolute path names)

If you encounter any issues with this software please report them on the GNUCOBOL Discussion page and I will attemp to address any issues your have.

Also if anyone would like to have a 64 bit version of this software, please comment on the GNUCOBOL Discussion page and I will look into creating a 64 bit version.

Thanks

Chuck Haatvedt