April 29, 2016

wolfSSL + IAR Workbench + embOS

This document will summarize and explain how to build a wolfssl library for linking against in IAR Workbench and how to then use a basic embOS evaluation project to run the wolfcrypt tests.

Contents of this document:

Required Software for this guide:

IAR Embedded Workbench IDE - ARM v7.60.1 (or similar version)
embOS v4.16 (for Cortex-M and IAR compiler)

Required items (Hardware) for this guide:

Atmel SAM V71 Xplained Ultra (Or equivelent Cortex-M Evaluation Board)
  Note: Must have J-Trace adapter (SWD or SWD + ETM)
j-Trace for ARM Cortex-M Processors
  Note: You can see here:
https://www.segger.com/j-trace-for-cortex-m.html
  Note: You can also purchase other models, we used one from IAR v3.2
20 pin target ribbon cable
USB Cable with j-Trace adapter end
Micro usb cable

Create wolfcrypt_lib.a

1. Open IAR Workbench -> Project -> Create New Project -> Empty Project
    a. Browsed to wolfssl/IDE/IAR-EWARM/
    b. create new directory "wolfcrypt_build_lib"
    c. Called it wolfcrypt_lib.ewp and hit "Save"

2. Go to Project -> Add Group...
    a. called group wolfcrypt_sources

3. Right click on wolfcrypt_sources group -> Add -> Add Files...

4. Browse to "wolfssl"/wolfcrypt/src directory and select the following files:

```
#----------------------------------------------#
#     aes.c                   memory.c         #
#     asn.c                   misc.c           #
#     chacha.c                poly1305.c       #
#     chacha20_poly1305.c     pwdbased.c       #
#     coding.c                rabbit.c         #
#     des3.c                  random.c         #
#     dh.c                    rsa.c            #
#     dsa.c                   sha.c            #
#     ecc.c                   sha256.c         #
#     hash.c                  sha512.c         #
#     hmac.c                  tfm.c            #
#     md4.c                   wc_encrypt.c     #
#     md5.c                   wc_port.c        #
#----------------------------------------------#
```

5. Once those are all added go to
   Project -> Properties C/C++ Compiler -> Preprocessor (Tab)
   a. In the field "Defined symbols: (one per line)"
      add WOLFSSL_USER_SETTINGS
   b. In the field "Additional include directories: (one per line)"
      Click the box with "..."
      Click <Click to add>
      Browse to "wolfssl" directory on your PC and click "Select" then "Ok"
      You should now see "C:\<path to>\wolfssl" in that field
      Click "OK"

6. Now we will create user_settings.h to hold our WOLFSSL_USER_SETTINGS
   a. Browse to "wolfssl" directory on your PC and add a new file

   b. name the file "user_settings.h"
   c. You should now have C:\<path-to>\wolfssl\user_settings.h
   d. For this evaluation we did the following:

```
#====== Contents of user_settings.h ======#
#ifndef _EMBOS_USER_SETTINGS_H_
#define _EMBOS_USER_SETTINGS_H_

    #define WOLFCRYPT_ONLY
    #define ECC_SHAMIR
    #define NO_64BIT
    #define SIZEOF_LONG 4
    #define SIZEOF_LONG_LONG 8
    #define HAVE_ECC
    #define HAVE_CHACHA
    #define HAVE_POLY1305
    #define WOLFSSL_SHA512
    #define WOLFSSL_SHA384
    #define HAVE_AESGCM
    #define NO_INLINE
    #define USE_WOLFSSL_MEMORY
    #define WOLFSSL_TRACK_MEMORY

#endif /* _EMBOS_USER_SETTINGS_H_ */
#====== End of user_settings.h file ======#
```

7. In Project -> Options -> General Options -> Output (Tab)
   Check the option for "Library" instead of "Executable"
   Click "OK"

8. Go to Project -> Rebuild all (The library should build)

9. Confirm the library is now located here:
   C:\<path

to>\wolfssl\IDE\IAR-EWARM\wolfcrypt_build_lib\Debug\Exe\wolfcrypt_lib.a

We are now set to link to this library in the evaluation project

---

Evaluate in embOS project:

1. go to embOS website and download a trial for your platform
   https://www.segger.com/downloads/embos ->
   embOS trial for Cortex-M and IAR compiler

2. After download and unzipping open IAR Workbench
   a. choose Project -> Add Existing Project
   b. Navigate to the downloaded trial then:
      Start/BoardSupport/Atmel/SAMV71_XPlainedUltra/Start_SAMv71.ewp

3. Go to Project -> Options -> C/C++ Compiler -> Preprocessor (Tab)
   a. Add this to the "Defined symbols: (one per line)" Box:
      WOLFSSL_USER_SETTINGS
   b. Now in the above box "Additional include directories:"
      Click on the box with "..."
      Click <Click to add>
      Browse to "wolfssl" directory on your computer and click "Select"
   c. This project will use the same user_settings.h as in the above project
      where we built the library so both have the exact same settings

4. Go to Project -> Options -> Linker -> Library (Tab)
   Next to the field "Additonal libraries: (one per line)"
   Click the box with "..."
   Click <Click to add>
   Browse to C:\<path
to>\wolfssl\IDE\IAR-EWARM\wolfcrypt_build_lib\Debug\Exe\wolfcrypt_lib.a
   Click "Open" and "OK" (twice)

5. In the main project file "OS_StartLEDBlick.c"

```
    #--- Headers ---#
        #include <stdio.h>
        #include <wolfcrypt/test/test.h>

    #--- Task for wolfCrypt ---#
        static OS_STACKPTR int StackWOLF[20000];
        static OS_TASK TCWOLF;
        static void wolfTestRun(void) {
            printf("Begin wolfcrypt tests\n");
            wolfcrypt_test(NULL);
            printf("Wolfcrypt tests complete\n");
        }

    #--- Create the task in main() BEFORE calling OS_Start() ---#
        OS_CREATETASK(&TCWOLF, "wolfcrypt test Task", wolfTestRun,
100, StackWOLF);
```

6. To get Logging working for seeing test results:
    a. Go to Project -> Options -> General Options -> Library Configuration (Tab)
    b. In the field "Library low-level interface implementation"
        Check the radio button for Semihosted
        Then in the inner field marked "stdout/stderr"
            Check the radio button for "Via semihosting" and click "OK"

7. Now we're ready to build and debug the project.
    a. Project -> Options -> Debugger
        In the field marked "Driver" Select drop-down and choose J-Link/J-Trace
        Click "OK"

b. Project -> Rebuild All
c. Connect your Cortex-M evaluation board to j-Trace and j-Trace to PC
d. Connect the micro-USB to the debug port of the Cortex-M and PC for power
e. Project -> Download and Debug

8. Once the Debug environment spawns go to View -> Terminal I/O
This is where the "printf" to stdout will be directed

9. In the Debug Menu bar look for the little square with three arrows pointing
to the right. When you mouse over it should say "GO"
Click this option and in the Terminal I/O Window you should see something
like this (depends on which functionality you set in user_settings.h)

SEE NOTE 1 BELOW IF YOU OBSERVE AN ERROR (-40) IN RSA TEST

```
Begin wolfcrypt tests
MD5      test passed!
MD4      test passed!
SHA      test passed!
SHA-256  test passed!
SHA-384  test passed!
SHA-512  test passed!
HMAC-MD5 test passed!
HMAC-SHA test passed!
HMAC-SHA256 test passed!
HMAC-SHA384 test passed!
HMAC-SHA512 test passed!
GMAC     test passed!
HC-128   test passed!
Rabbit   test passed!
```

```
Chacha   test passed!
POLY1305 test passed!
ChaCha20-Poly1305 AEAD test passed!
DES      test passed!
DES3     test passed!
AES      test passed!
AES-GCM  test passed!
RANDOM   test passed!
RSA      test passed!
DH       test passed!
DSA      test passed!
PWDBASED test passed!
ECC      test passed!

total   Allocs =       988
total   Bytes  =   1471829
peak    Bytes  =     29576
current Bytes  =         0
```

Thank you for using this guide and we hope this was helpful to you. If you have
any suggestions / feedback for us please contact us:
support@wolfssl.com
info@wolfssl.com

NOTE 1:

If you are working off of a base example project and you observe memory errors when malloc is called, make sure that the linker script has set the HEAP high enough.
Project -> Options -> Linker -> (Config Tab) -> Linker configuration file

Check this file for __size_heap__ it is typically set to 0x200 by default.

There are places in wolfSSL that malloc as much as 4k at a time and peak usage can be as high as 29K if using full functionality.