Building wolfSSL with Xilinx

July 2017

1.0 INTRO

This document covers the basics of building wolfSSL with Xilinx. This document assumes that a bitstream has already been created from Vivado and just covers building and using wolfSSL. Source code can be seen on GitHub at https://github.com/wolfSSL/wolfssl.

1.1 Limitations

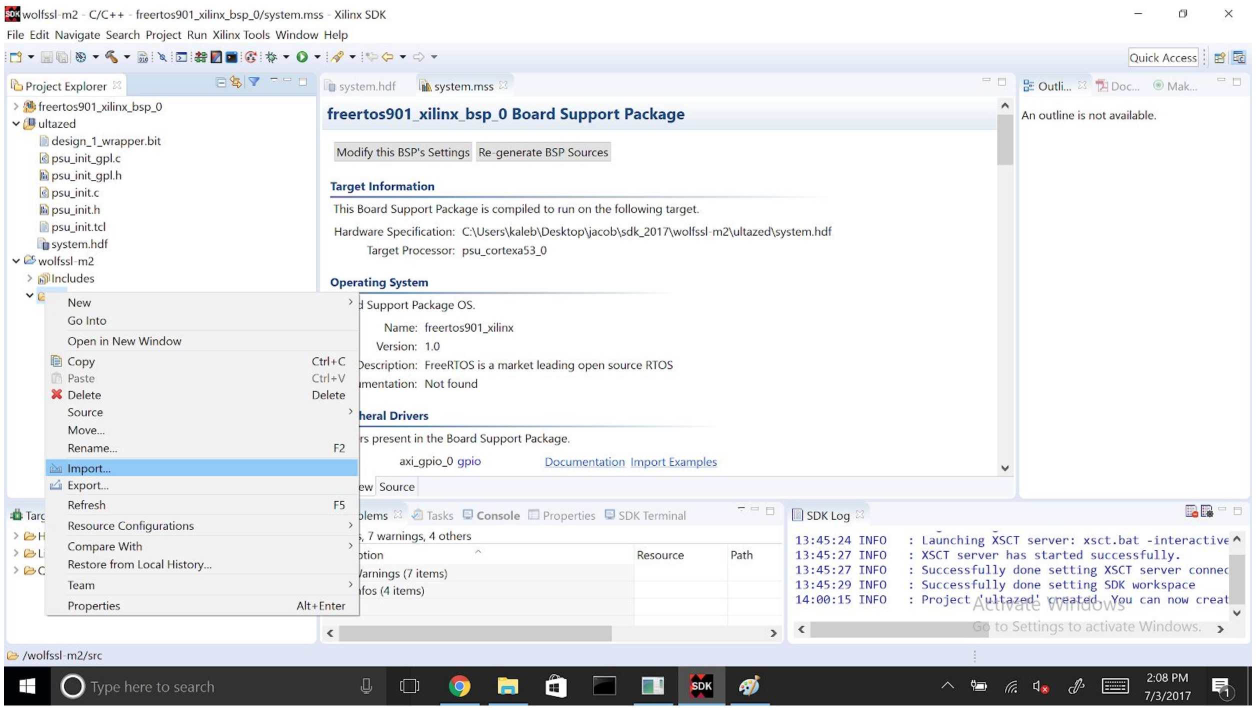
Both SHA3 and AES-GCM with using WOLFSSL_XIlINX_CRYPT expect that the length of input data is a multiple of 4 i.e. (4, 12, 16...).

2.0 Building wolfSSL with Xilinx SDK version 2017.1

2.1 Importing wolfSSL code for creating a library

Create a new application with (File->New->Application Project) after filling out the name of the project select freertos and then next and choose an empty project.

Next import wolfSSL source files. To do this expand the newly created project in the Project Explorer tab and right click on src then select Import.



This Board Support Package is compiled to run on the following target.

Hardware Specification: C:\Users\kaleb\Desktop\jacob\sdk_2017\wolfssl-m2\ultazed\system.hdf

Target Processor: psu_cortexa53_0

Name: freertos901_xilinx

Version: 1.0

Description: FreeRTOS is a market leading open source RTOS

Documentation: Not found

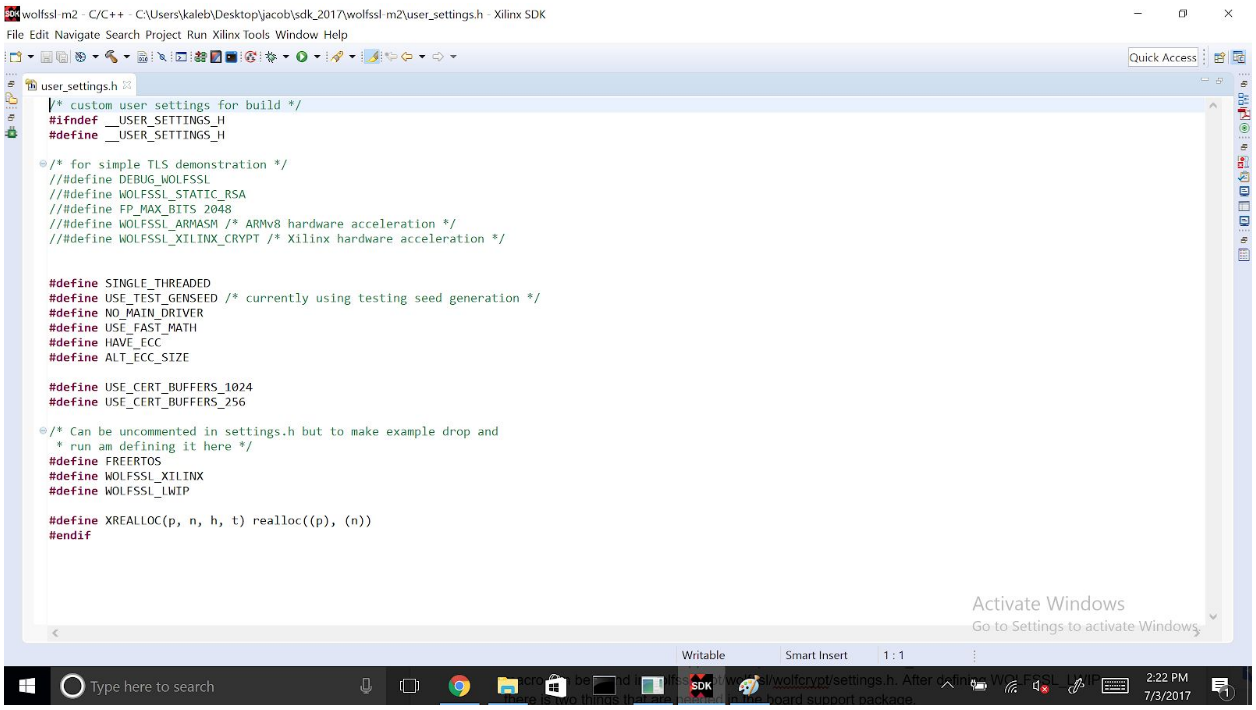
axi_gpio_0 gpio

```
13:45:24 INFO   : Launching XSCT server: xsct.bat -interactive
13:45:27 INFO   : XSCT server has started successfully.
13:45:27 INFO   : Successfully done setting XSCT server connec
13:45:29 INFO   : Successfully done setting SDK workspace
14:00:15 INFO   : Project 'ultazed' created. You can now creat
```

The files that need to be imported are in wolfssl-root/wolfcrypt/src and wolfssl-root/src/. Also optionally wolfssl-root/wolfcrypt/src/port/arm or wolfssl-root/wolfcrypt/src/port/xilinx to take advantage of hardware acceleration.

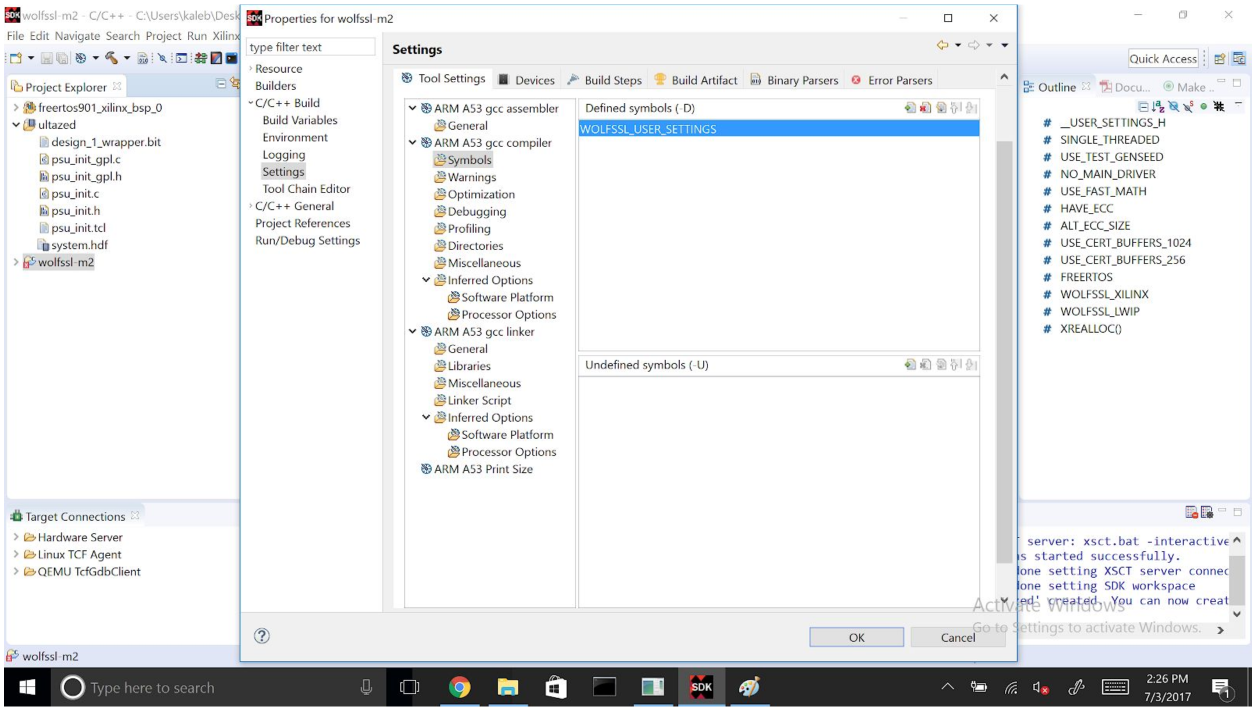
NOTE:
Currently evp.c and bio.c should not be compiled if not using OPENSSL_EXTRA, by default misc.c should also not be compiled. Misc.c only needs to be compiled if the macro NO_INLINE is defined. The file wolfcrypt/src/aes_asm.s should also not be compiled and will cause issues if compiled on an architecture that does not support at&t asm syntax.

When building wolfSSL as a library define NO_MAIN_DRIVER to make sure that the wolfcrypt test and benchmark applications do not use a “main” function if added to the project. For building with Xilinx the macro WOLFSSL_XILINX is needed, optionally the macro WOLFSSL_XILINX_CRYPT turns on available Xilinx hardened crypto. The following is an example user_settings.h for building wolfSSL. Note that if the RSA 2048 patch has been applied to xsecure_rsa than WOLFSSL_XILINX_PATCH should be defined to use it.



```
/* custom user settings for build */
#ifndef __USER_SETTINGS_H
#define __USER_SETTINGS_H

/* for simple TLS demonstration */
//#define DEBUG_WOLFSSL
//#define WOLFSSL_STATIC_RSA
//#define FP_MAX_BITS 2048
//#define WOLFSSL_ARMASM /* ARMv8 hardware acceleration */
//#define WOLFSSL_XILINX_CRYPT /* Xilinx hardware acceleration */


#define SINGLE_THREADED
#define USE_TEST_GENSEED /* currently using testing seed generation */
#define NO_MAIN_DRIVER
#define USE_FAST_MATH
#define HAVE_ECC
#define ALT_ECC_SIZE

#define USE_CERT_BUFFERS_1024
#define USE_CERT_BUFFERS_256

/* Can be uncommented in settings.h but to make example drop and
 * run am defining it here */
#define FREERTOS
#define WOLFSSL_XILINX
#define WOLFSSL_LWIP

#define XREALLOC(p, n, h, t) realloc((p), (n))
#endif
```

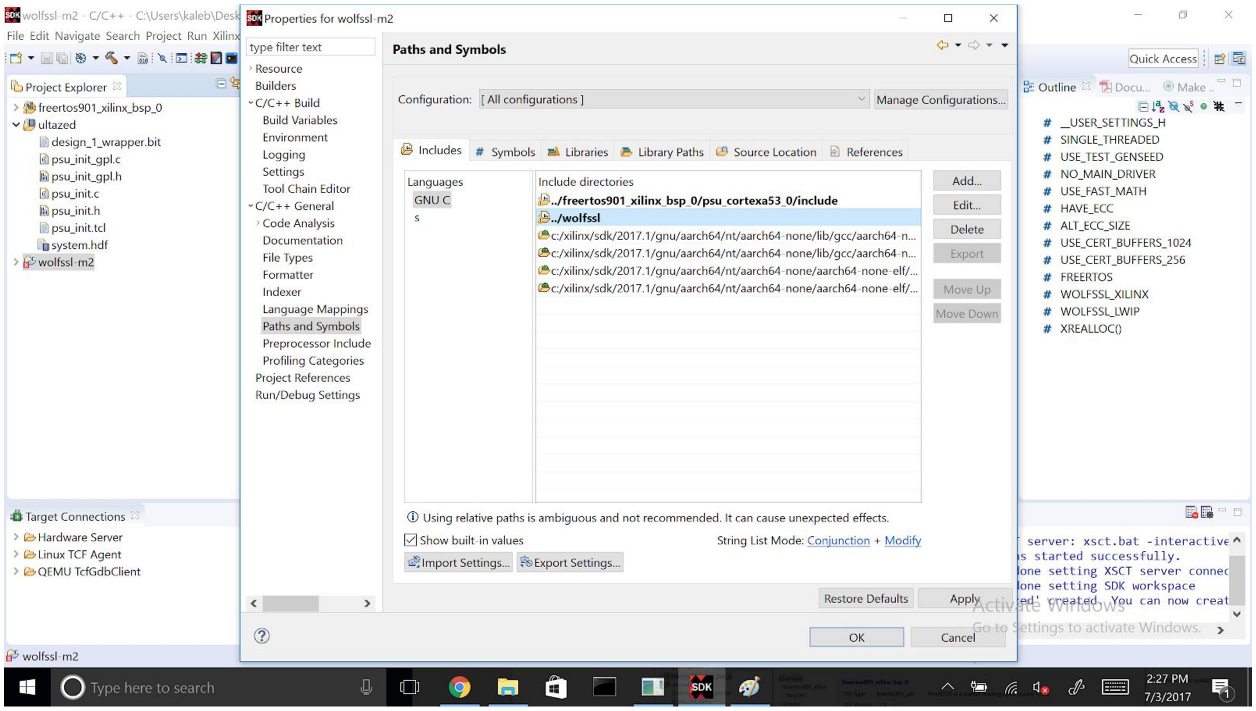
It is recommended that a user_settings.h file is used. This keeps all macros defined separate from the main wolfSSL code, allowing for updating wolfSSL version without erasing user defined macros. To use a user_settings.h file add the macro WOLFSSL_USER_SETTINGS to Symbols in C/C++ Build -> Setting.



Defined symbols (-D)

WOLFSSL_USER_SETTINGS

Now add in the path to root wolfSSL directory for header files, and the path to the user_settings.h file if used. In the following image the root wolfSSL directory is wolfssl and is located up one directory relative to the current project.



Properties for wolfssl-m2 — Paths and Symbols

Configuration: [All configurations]

Tabs: Includes, Symbols, Libraries, Library Paths, Source Location, References

Languages: GNU C, s

Include directories:

```
../freertos901_xilinx_bsp_0/psu_cortexa53_0/include
../wolfssl
c:/xilinx/sdk/2017.1/gnu/aarch64/nt/aarch64-none/lib/gcc/aarch64-n...
c:/xilinx/sdk/2017.1/gnu/aarch64/nt/aarch64-none/lib/gcc/aarch64-n...
c:/xilinx/sdk/2017.1/gnu/aarch64/nt/aarch64-none/aarch64-none-elf/...
c:/xilinx/sdk/2017.1/gnu/aarch64/nt/aarch64-none/aarch64-none-elf/...
```

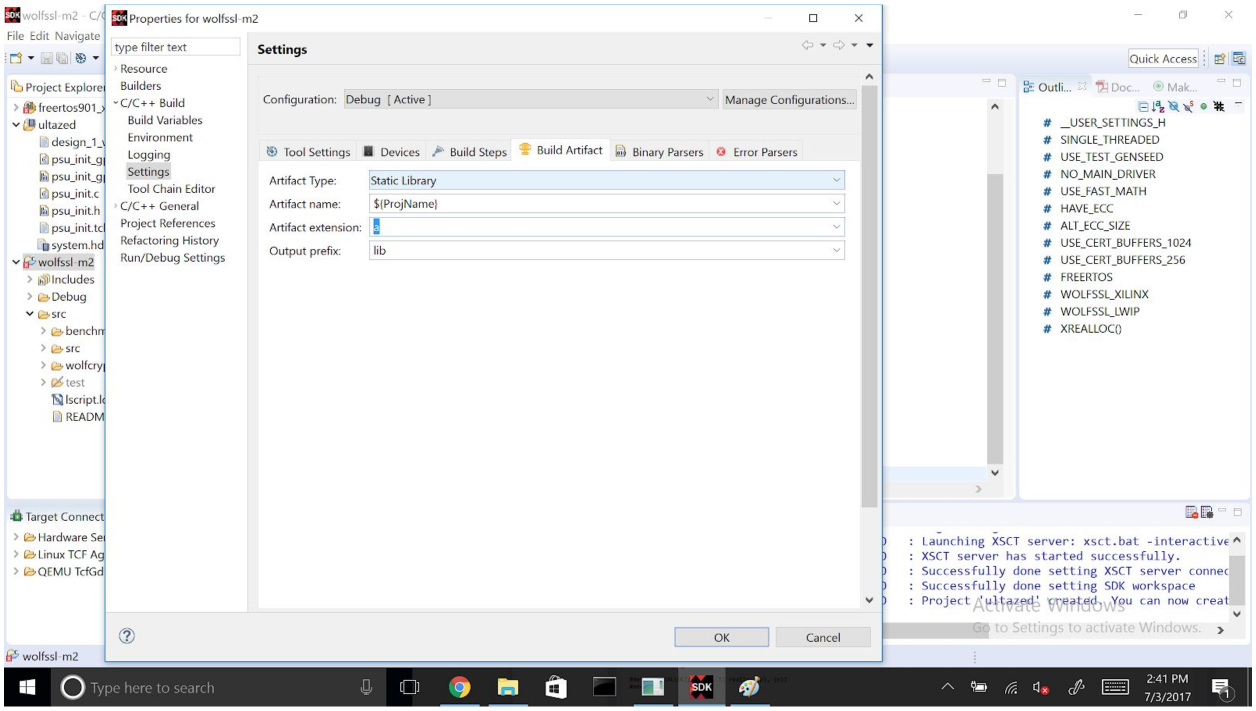
Using relative paths is ambiguous and not recommended. It can cause unexpected effects.

Show built-in values — String List Mode: Conjunction + Modify

Outline:

```
# _USER_SETTINGS_H
# SINGLE_THREADED
# USE_TEST_GENSEED
# NO_MAIN_DRIVER
# USE_FAST_MATH
# HAVE_ECC
# ALT_ECC_SIZE
# USE_CERT_BUFFERS_1024
# USE_CERT_BUFFERS_256
# FREERTOS
# WOLFSSL_XILINX
# WOLFSSL_LWIP
# XREALLOC()
```

Next, to make the project a library instead of an executable open the project settings and adjust the “Build Artifact”.



Properties for wolfssl-m2 — Settings

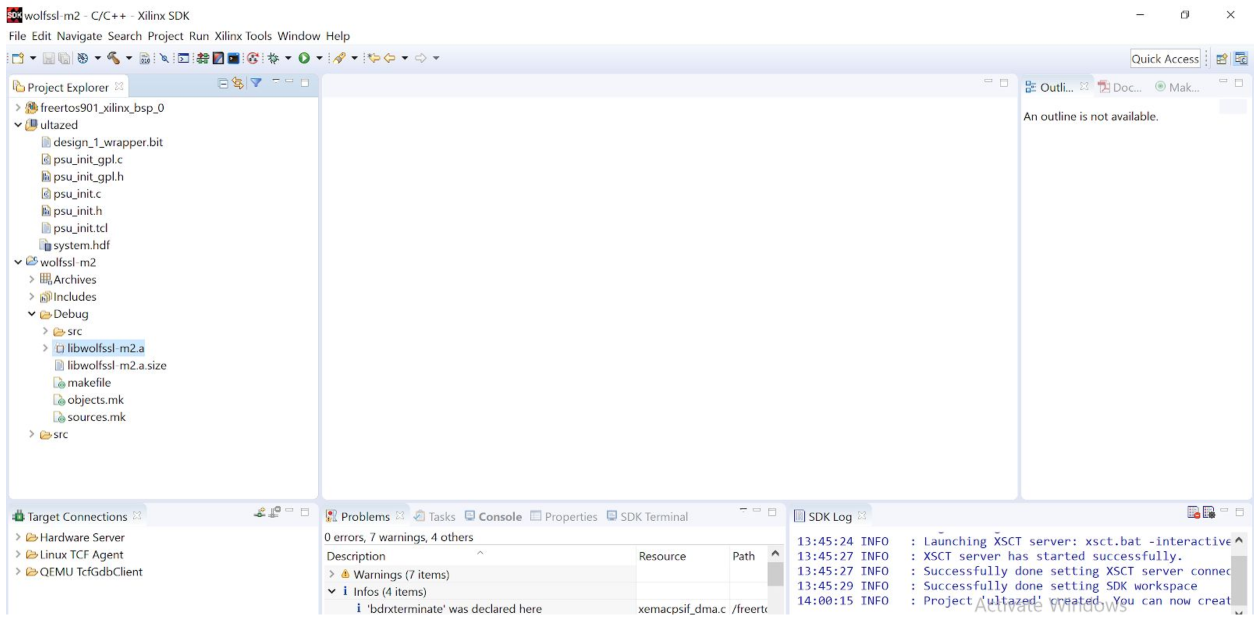
Configuration: Debug [Active]

Tabs: Tool Settings, Devices, Build Steps, Build Artifact, Binary Parsers, Error Parsers

Artifact Type:	Static Library
Artifact name:	${ProjName}
Artifact extension:	a
Output prefix:	lib

```
: Launching XSCT server: xsct.bat -interactive
: XSCT server has started successfully.
: Successfully done setting XSCT server connec
: Successfully done setting SDK workspace
: Project 'ultazed' created. You can now creat
```

Now when building a <name>.a library will be created in the active build directory.



```
> freertos901_xilinx_bsp_0
v ultazed
    design_1_wrapper.bit
    psu_init_gpl.c
    psu_init_gpl.h
    psu_init.c
    psu_init.h
    psu_init.tcl
    system.hdf
v wolfssl-m2
  > Archives
  > Includes
  v Debug
    > src
    > libwolfssl-m2.a
      libwolfssl-m2.a.size
      makefile
      objects.mk
      sources.mk
  > src
```

An outline is not available.

```
> Hardware Server
> Linux TCF Agent
> QEMU TcfGdbClient
```

0 errors, 7 warnings, 4 others

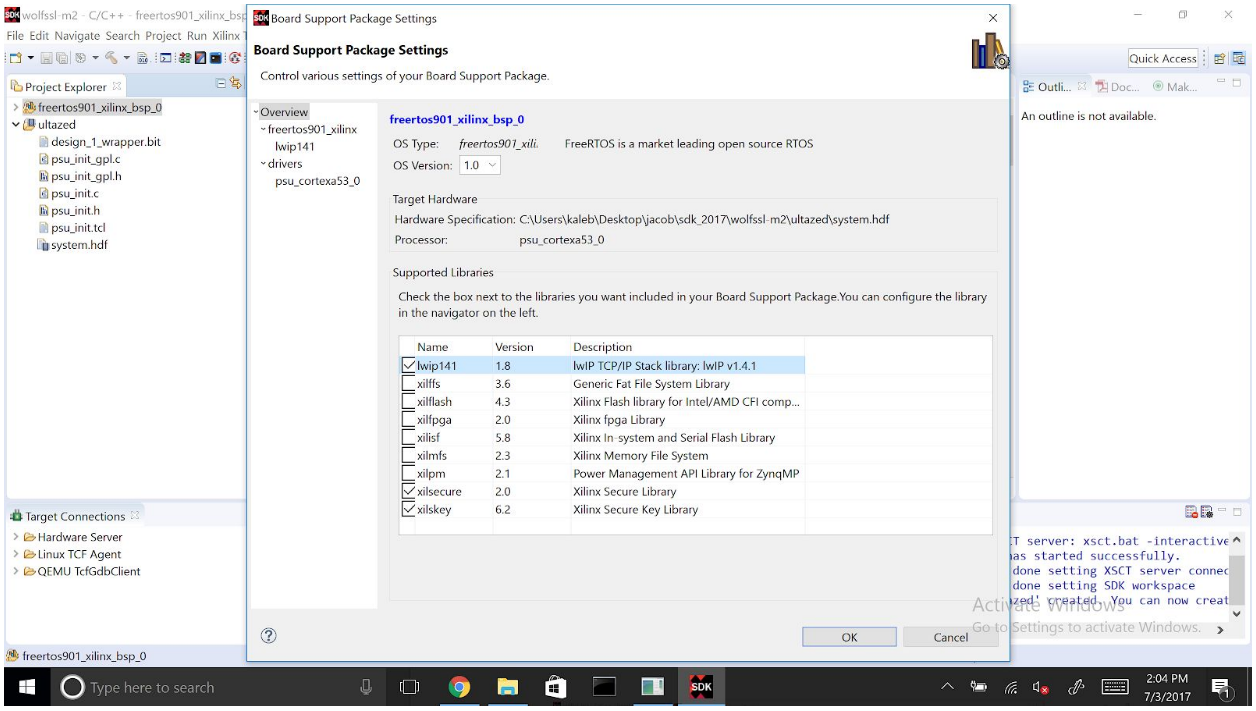
Description	Resource	Path
> Warnings (7 items)		
v Infos (4 items)		
'bdrxterminate' was declared here	xemacpsif_dma.c	/freert

```
13:45:24 INFO   : Launching XSCT server: xsct.bat -interactive
13:45:27 INFO   : XSCT server has started successfully.
13:45:27 INFO   : Successfully done setting XSCT server connec
13:45:29 INFO   : Successfully done setting SDK workspace
14:00:15 INFO   : Project 'ultazed' created. You can now creat
```

2.2 Adding lwip

To enable wolfSSL support of lwip the macro WOLFSSL_LWIP should be defined. This macro can be found in wolfssl-root/wolfssl/wolfcrypt/settings.h and can also be set in user_settings.h instead. After defining WOLFSSL_LWIP there is two things that are needed in the board support package.

Change bsp to have lwip if not already enabled



wolfssl-m2 - C/C++ - freertos901_xilinx_bsp

File Edit Navigate Search Project Run Xilinx

Project Explorer

- freertos901_xilinx_bsp_0
- ultazed
  - design_1_wrapper.bit
  - psu_init_gpl.c
  - psu_init_gpl.h
  - psu_init.c
  - psu_init.h
  - psu_init.tcl
  - system.hdf

Target Connections

- Hardware Server
- Linux TCF Agent
- QEMU TcfGdbClient

Board Support Package Settings

Control various settings of your Board Support Package.

- Overview
  - freertos901_xilinx
    - lwip141
  - drivers
    - psu_cortexa53_0

freertos901_xilinx_bsp_0

OS Type: freertos901_xili. FreeRTOS is a market leading open source RTOS

OS Version: 1.0

Target Hardware

Hardware Specification: C:\Users\kaleb\Desktop\jacob\sdk_2017\wolfssl-m2\ultazed\system.hdf

Processor: psu_cortexa53_0

Supported Libraries

Check the box next to the libraries you want included in your Board Support Package.You can configure the library in the navigator on the left.

Name	Version	Description
✓ lwip141	1.8	lwIP TCP/IP Stack library: lwIP v1.4.1
xilffs	3.6	Generic Fat File System Library
xilflash	4.3	Xilinx Flash library for Intel/AMD CFI comp...
xilfpga	2.0	Xilinx fpga Library
xilisf	5.8	Xilinx In-system and Serial Flash Library
xilmfs	2.3	Xilinx Memory File System
xilpm	2.1	Power Management API Library for ZynqMP
✓ xilsecure	2.0	Xilinx Secure Library
✓ xilskey	6.2	Xilinx Secure Key Library

OK Cancel

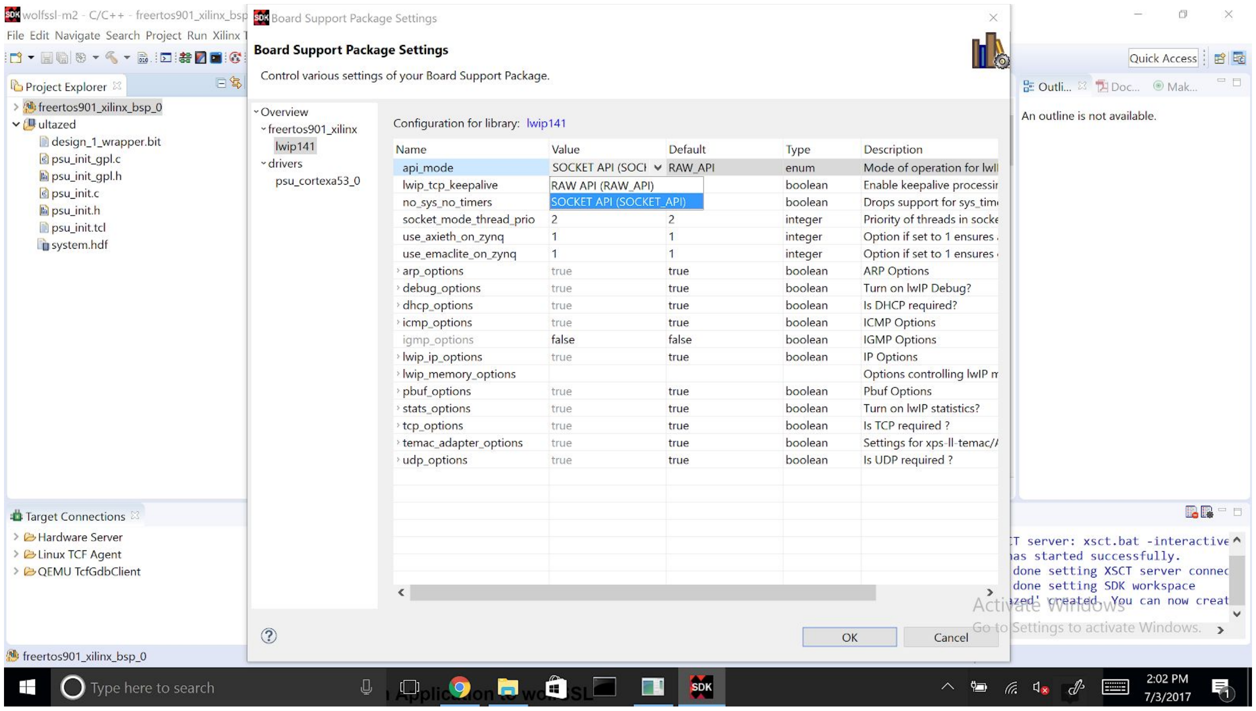
An outline is not available.

```
T server: xsct.bat -interactive
as started successfully.
done setting XSCT server connec
done setting SDK workspace
zed' created. You can now creat
```

freertos901_xilinx_bsp_0

2:04 PM 7/3/2017

Change the board support package settings of lwip to be SOCKET instead of the default RAW



Board Support Package Settings

Control various settings of your Board Support Package.

Configuration for library: lwip141

Name	Value	Default	Type	Description
api_mode	SOCKET API (SOCI	RAW_API	enum	Mode of operation for lwI
lwip_tcp_keepalive	RAW API (RAW_API)		boolean	Enable keepalive processi
no_sys_no_timers	SOCKET API (SOCKET_API)		boolean	Drops support for sys_tim
socket_mode_thread_prio	2	2	integer	Priority of threads in socke
use_axieth_on_zynq	1	1	integer	Option if set to 1 ensures
use_emaclite_on_zynq	1	1	integer	Option if set to 1 ensures
arp_options	true	true	boolean	ARP Options
debug_options	true	true	boolean	Turn on lwIP Debug?
dhcp_options	true	true	boolean	Is DHCP required?
icmp_options	true	true	boolean	ICMP Options
igmp_options	false	false	boolean	IGMP Options
lwip_ip_options	true	true	boolean	IP Options
lwip_memory_options				Options controlling lwIP m
pbuf_options	true	true	boolean	Pbuf Options
stats_options	true	true	boolean	Turn on lwIP statistics?
tcp_options	true	true	boolean	Is TCP required ?
temac_adapter_options	true	true	boolean	Settings for xps-ll-temac/
udp_options	true	true	boolean	Is UDP required ?

OK Cancel

2:02 PM 7/3/2017

2.3 Adding Xilinx hardened crypto

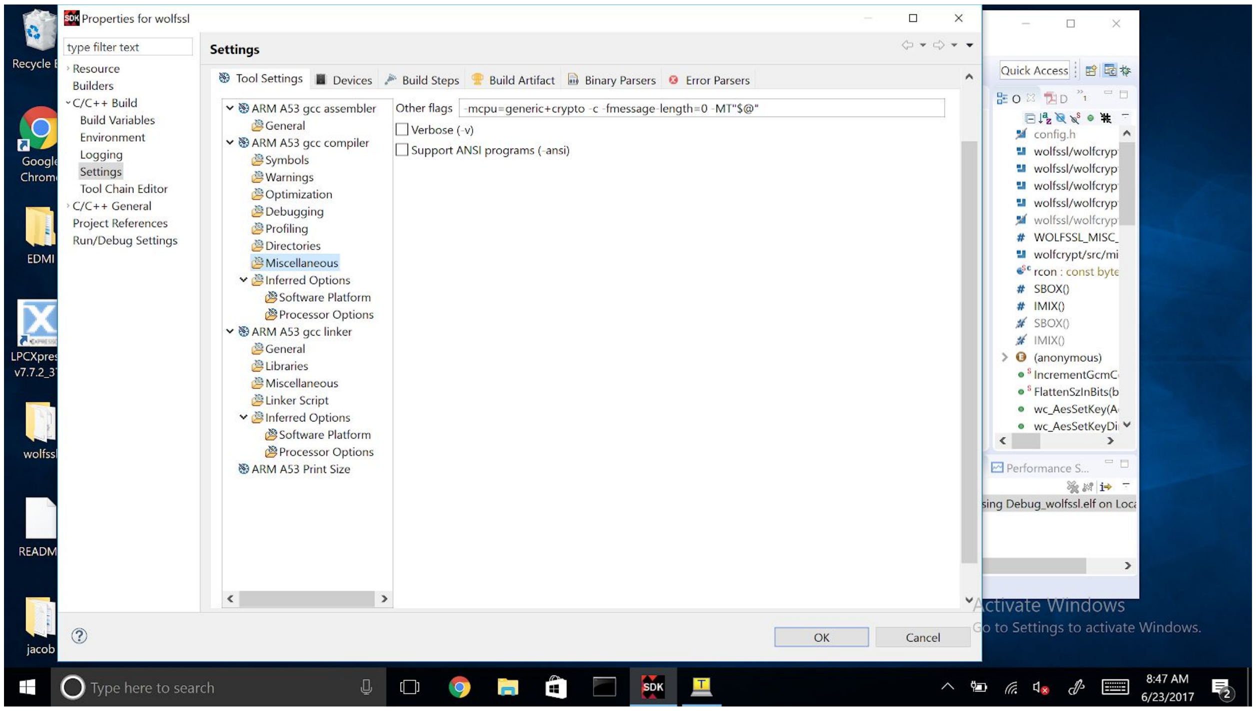
For adding in use of Xilinx hardened crypto calls define the macro WOLFSSL_XILINX_CRYPT either in user_settings.h or as an additional “Symbol” in C/C++ Build Settings, and then recompile.

2.4 Adding ARMv8 hardware acceleration

The macro WOLFSSL_ARMASM should be defined for wolfSSL. This can be done with adding it to Symbols in C/C++ Build -> Settings or by setting it in a user_settings.h file.

To build with 64bit extensions the compiler flag “-mcpu=generic+crypto” is needed. Generic can be replaced with specific cpu if looking to fine tune performance when compiling. Here is a link to options available with gcc when compiling for aarch64 https://gcc.gnu.org/onlinedocs/gcc/AArch64-Options.html.

To set with using the SDK right click on the project and go to Properties. In Properties got to C/C++ Build -> Settings -> ARM A53 gc compiler -> Miscellaneous, and add to “Other flags”



Other flags: -mcpu=generic+crypto -c -fmessage-length=0 -MT"$@"

☐ Verbose (-v)

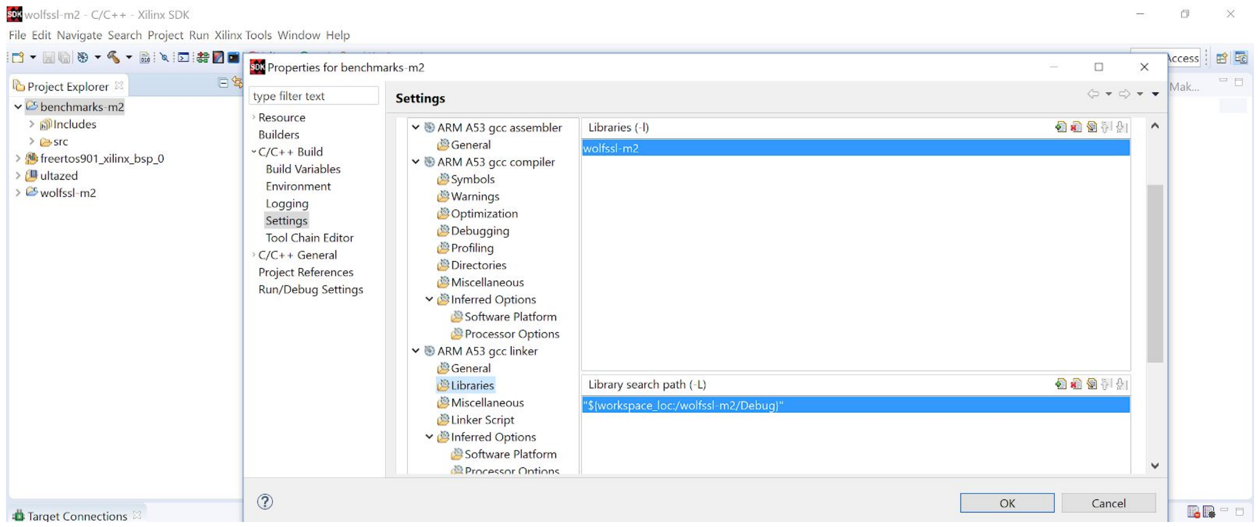
☐ Support ANSI programs (-ansi)

3.0 Linking An Application to wolfSSL

Create new application

Right click go to (Properties->C/C++ Build->Settings->Tool Settings) select Libraries under

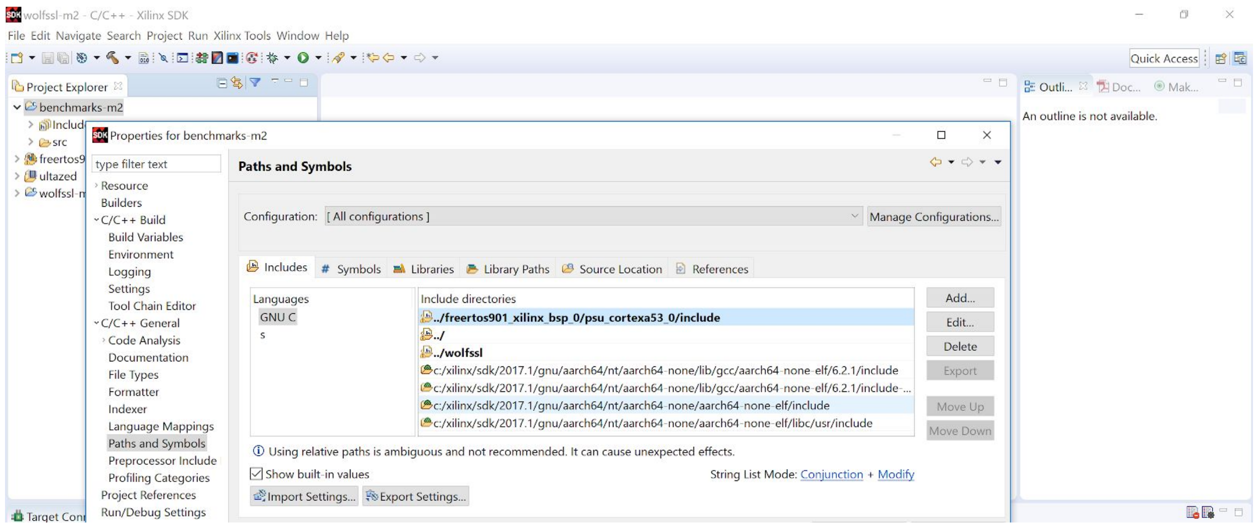
“ARM A53 gcc linker” and add the library name and path.



Libraries (-l)
wolfssl-m2

Library search path (-L)
"${workspace_loc:/wolfssl-m2/Debug}"

Add include path for wolfSSL header files.
Right click go to (Properties->C/C++ General->Paths and Symbols->Includes) select Add and add the file path to wolfSSL header files. If using a user_settings.h file remember to add the macro WOLFSSL_USER_SETTINGS to the project and to include the path to user_settings.h file to be used. The WOLFSSL_USER_SETTINGS macro can be set by (Properties->C/C++ Build->Settings->Tool Settings) and select Symbols under “ARM A53 gcc compiler”. Add WOLFSSL_USER_SETTINGS to “Defined symbols”.


wolfssl-m2 - C/C++ - Xilinx SDK
File Edit Navigate Search Project Run Xilinx Tools Window Help
Project Explorer
benchmarks-m2
Quick Access
An outline is not available.

Properties for benchmarks-m2

Paths and Symbols

Configuration: [All configurations]

Includes | Symbols | Libraries | Library Paths | Source Location | References

Languages: GNU C

Include directories:

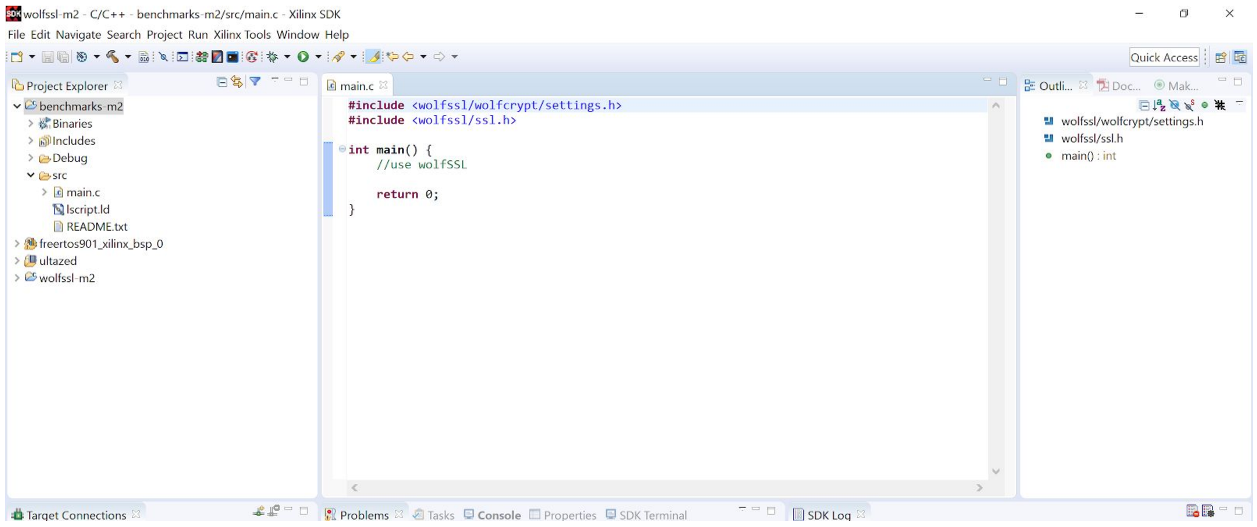
```
../freertos901_xilinx_bsp_0/psu_cortexa53_0/include
../
../wolfssl
c:/xilinx/sdk/2017.1/gnu/aarch64/nt/aarch64-none/lib/gcc/aarch64-none-elf/6.2.1/include
c:/xilinx/sdk/2017.1/gnu/aarch64/nt/aarch64-none/lib/gcc/aarch64-none-elf/6.2.1/include-...
c:/xilinx/sdk/2017.1/gnu/aarch64/nt/aarch64-none/aarch64-none-elf/include
c:/xilinx/sdk/2017.1/gnu/aarch64/nt/aarch64-none/aarch64-none-elf/libc/usr/include
```

Using relative paths is ambiguous and not recommended. It can cause unexpected effects.

☑ Show built-in values

String List Mode: Conjunction + Modify

Lastly include wolfSSL header files and enjoy using wolfSSL!

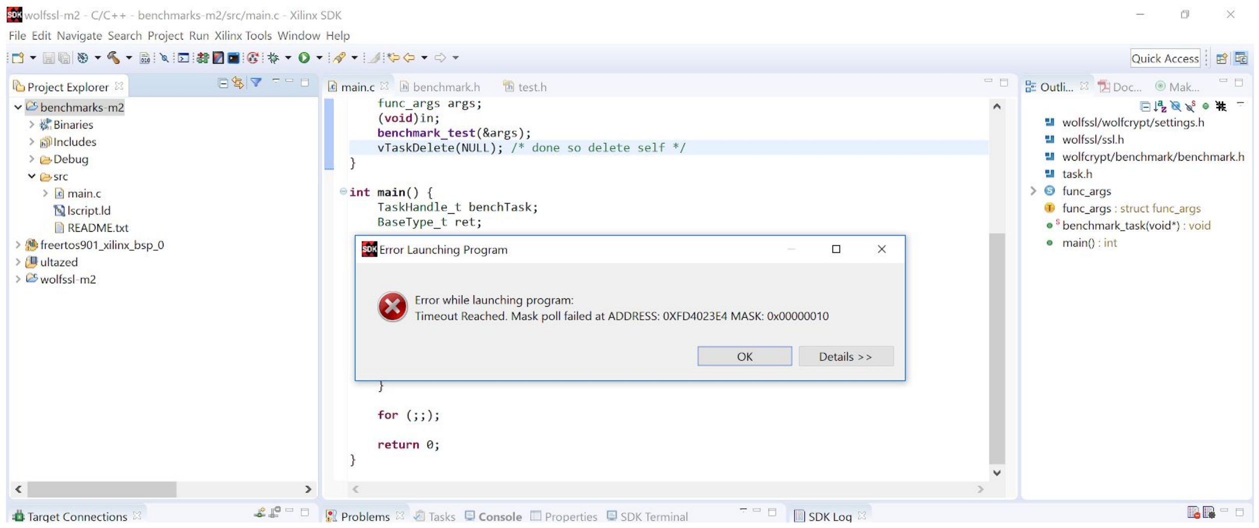


main.c

```
#include <wolfssl/wolfcrypt/settings.h>
#include <wolfssl/ssl.h>

int main() {
    //use wolfSSL

    return 0;
}
```

Strange Behavior

When flashing using jtag a mask poll error is found.



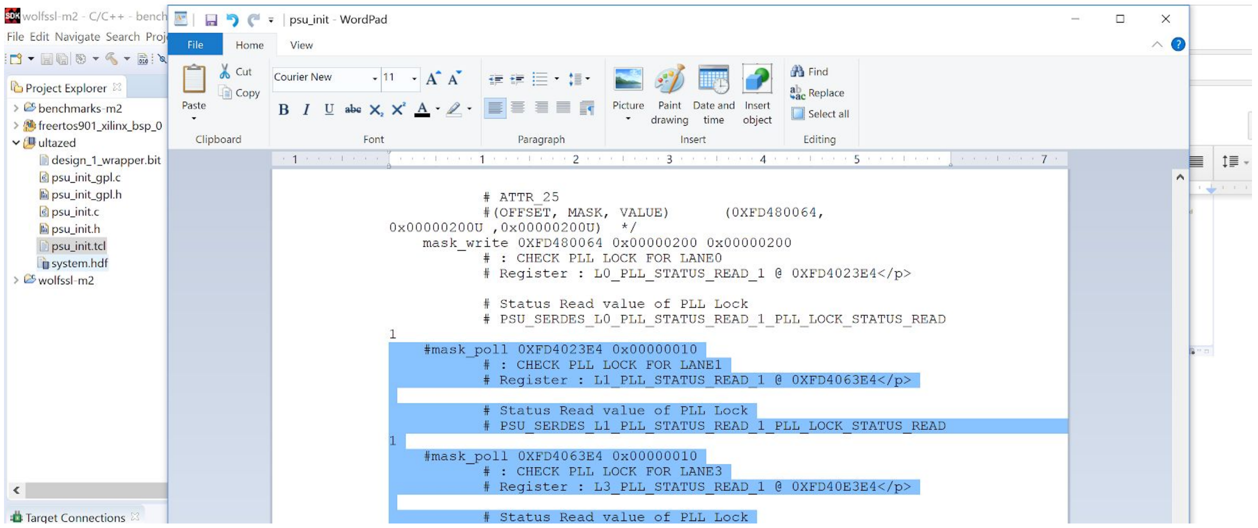
```
    func_args args;
    (void)in;
    benchmark_test(&args);
    vTaskDelete(NULL); /* done so delete self */
}

int main() {
    TaskHandle_t benchTask;
    BaseType_t ret;
```

Error Launching Program

```
Error while launching program:
Timeout Reached. Mask poll failed at ADDRESS: 0XFD4023E4 MASK: 0x00000010
```

```
    }

    for (;;);

    return 0;
}
```

To work around this issue comment out the three mask_poll calls in psu_init.tcl file.



```
                # ATTR_25
                #(OFFSET, MASK, VALUE)      (0XFD480064,
0x00000200U ,0x00000200U)  */
         mask_write 0XFD480064 0x00000200 0x00000200
                # : CHECK PLL LOCK FOR LANE0
                # Register : L0_PLL_STATUS_READ_1 @ 0XFD4023E4</p>

                # Status Read value of PLL Lock
                # PSU_SERDES_L0_PLL_STATUS_READ_1_PLL_LOCK_STATUS_READ
1
         #mask_poll 0XFD4023E4 0x00000010
                # : CHECK PLL LOCK FOR LANE1
                # Register : L1_PLL_STATUS_READ_1 @ 0XFD4063E4</p>

                # Status Read value of PLL Lock
                # PSU_SERDES_L1_PLL_STATUS_READ_1_PLL_LOCK_STATUS_READ
1
         #mask_poll 0XFD4063E4 0x00000010
                # : CHECK PLL LOCK FOR LANE3
                # Register : L3_PLL_STATUS_READ_1 @ 0XFD40E3E4</p>

                # Status Read value of PLL Lock
```

What to do in the case that wolfSSL header files are not found:

Check that the “Paths and Symbols” include the location of the wolfssl root directory.