Summary from TI on porting CC3000 host-driver code to Microblaze:

Using the sensor application example:

The connection from Microblaze to the CC3000 is **SPI** plus 2 GPIOs (**WLAN_EN** and **IRQ**).

===========================

IRQ line is an input to microblaze and its interrupt handling should be done in an IRQ handler similar to what is done in **spi.c** lines 620 to 665. All you need is to place the actual code into your IRQ handler.

Related functions that should be modified:

ReadWlanInterruptPin()

WlanInterruptEnable()

WlanInterruptDisable()

SpiCleanGPIOISR() in **spi.c** should clear the flag for the pin

SpiPauseSpi should disable the IRQ Interrupt

SpiResumeSpi should enable the IRQ interrupt

==========================

For the GPIOs, **WLAN_EN** is just a microblaze output that should be toggled High or Low .

WriteWlanPin() just toggles **WLAN_EN** high or low

========================

The SPI interface is just **SIMO, SOMI, SCK,** and **CSn**.

init_spi() function needs to be changed so that the SPI interface is properly initialized.

Microblaze should be a master with 3-wire system (**CSn** is separate) and a clock of 1 to 16MHz.

The definition of clock polarity and phase are:

Data is changed on the first **UCLK** edge and captured on the following edge.

Clock polarity : The inactive state is low.

For **CSN**, the respective I/O control is done through defines at the top of spi.c:

```
#define ASSERT_CS()      (P1OUT &= ~BIT3)
#define DEASSERT_CS()    (P1OUT |= BIT3)
```

Change these to toggle **CS** as needed (it's active low, that is, it is low when it is asserted).

SpiWriteDataSynchronous needs to be modified. `UCB0TXBUF = *data;` writes whatever data is pointing to via spi, while **UCB0RXBUF** doesn't seem to do anything with the data read.

SpiReadDataSynchronous is similar to above except data read is placed in **data[i]**

========================

initDriver() needs to be changed so that the clock initialization is removed.

========================

Unsolicited event timer should be created (it's a simple timer that fires every 500ms) and should be properly used in:

```
unsolicicted_events_timer_init()
unsolicicted_events_timer_disable()
```

=========================

`__delay_cycles(6000000);` or similar msp430 specific commands should be modified to create delays in whatever way you want. These should be blocking delays. (6000000 cycles is about 0.24 seconds).

==========================

buffers spi_buffer and **wlan_tx_buffer** should just be put in memory (I assume microblaze has enough).

Remove those macros and pragma definitions around those two buffers

==========================

There are other things in the software to be changed. For example switch handling and LEDs

(start from the independent host driver, and look at how the sensor application calls the APIs)