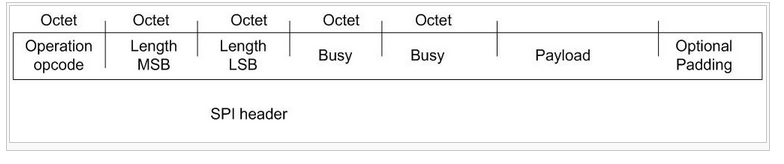
**Tiwi-SL Sample Code: First SPI Command Sent**

SPI protocol is shown:



The padding is one byte and is only added when , without it, the number of bytes in the entire message would be odd.

The two length bytes are a 16-bit value sent MSB (so not reversed), and indicate the number of bytes after the SPI header (so payload plus padding bit (if present)).

The operation code is either read or write.

#define SPI\_READ 3

#define SPI\_WRITE 1

In the case of the first write, it is SPI\_WRITE.

The Payload is an HCI message, the first byte of which is the “type”:

#define HCI\_TYPE\_CMND 0x1

#define HCI\_TYPE\_DATA 0x2

#define HCI\_TYPE\_PATCH 0x3

#define HCI\_TYPE\_EVNT 0x4

In the case of the first write, it’s HCI\_TYPE\_CMND, which must be followed by a two-byte command. The command in this case is HCI\_CMND\_SIMPLE\_LINK\_START, which is defined as:

#define HCI\_CMND\_SIMPLE\_LINK\_START 0x4000

16-bit commands and parameters are sent as LSB, so this command is sent as 0x00,0x40

The next byte is the length of the parameters. For this first call, it’s just one byte, so 0x01.

For this first message, we must tell the CC3000 whether to get the patches from the EEPROM (DEFAULT), to request the patches from the host MCU (FORCE\_HOST), or not to use any patches (FORCE\_NONE):

#define SL\_PATCHES\_REQUEST\_DEFAULT (0)

#define SL\_PATCHES\_REQUEST\_FORCE\_HOST (1)

#define SL\_PATCHES\_REQUEST\_FORCE\_NONE (2)

Here now are the messages for Smart Config, starting from power up:



1 (SPI\_WRITE)

0,5 (the length of the message after the 5-byte SPI header, sent as two bytes, most significant byte first)

0,0 (two busy bytes)

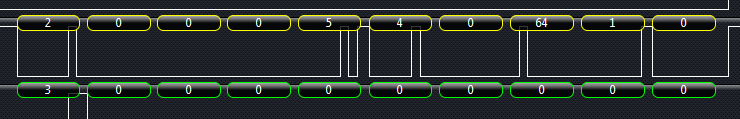
1 (HCI\_TYPE\_CMND)

0,64 (which is CMND\_SIMPLE\_LINK\_START, 0x4000, send as least significant byte first)

1 parameter length (WLAN\_SL\_INIT\_START\_PARAMS\_LEN)

1 (SL\_PATCHES\_REQUEST\_DEFAULT)

First receive:



3 (SPI\_READ)

0 means nothing on a read

0 means nothing on a read

0 means nothing on a read

0 means nothing on a read

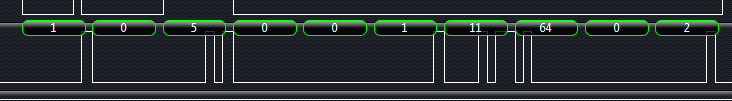
0 means nothing on a read

4 HCI\_TYPE\_EVENT

0,64 (which is CMND\_SIMPLE\_LINK\_START, 0x4000, send as least significant byte first)

1 length past HCI header, which is 4 bytes

0 not sure if this has any meaning, maybe a SUCCESS status

Second send: 

1 (SPI\_WRITE)

0,5 (the length sent as two bytes, most significant byte first)

0,0 (two busy bytes)

1 (HCI\_TYPE\_CMND)

11,64 (which is HCI\_CMND\_READ\_BUFFER\_SIZE, 0x400B, send as least significant byte first)

2 just a padding byte to make it an even number of bytes sent

Second receive:



4 HCI\_TYPE\_EVENT

11,64 (which is HCI\_CMND\_READ\_BUFFER\_SIZE, 0x400B, send as least significant byte first)

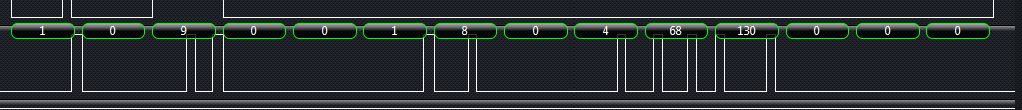
4.0 length past HCI header, which is 4 bytes (LSB, so flip)

6 number of free buffers

220,5 buffer length as LSB so 1500

0 padding byte

Third send:



1 (HCI\_TYPE\_CMND)

8,0 HCI\_CMND\_EVENT\_MASK (0x0008)

4 WLAN\_SET\_MASK\_PARAMS\_LEN

68,130,0,0 = HCI\_EVNT\_WLAN\_KEEPALIVE, HCI\_EVNT\_WLAN\_UNSOL\_INIT,

HCI\_EVNT\_WLAN\_ASYNC\_PING\_REPORT (or-ed together as 4 byte value)

Third Receive:



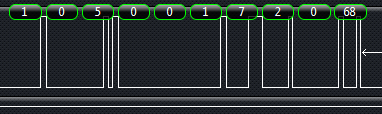
4 HCI\_TYPE\_EVENT

8,0 HCI\_CMND\_EVENT\_MASK (0x0008)

5,0 length (0X0005)

0,0,0,0 = 32 bit param (sample code seems to ignore these bytes)

Fourth send:



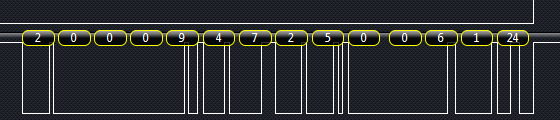
1 (HCI\_TYPE\_CMND)

7,2 HCI\_CMND\_READ\_SP\_VERSION (0x0207)

5 PARAMS\_LEN

68 = padding

Third Receive:



4 HCI\_TYPE\_EVENT

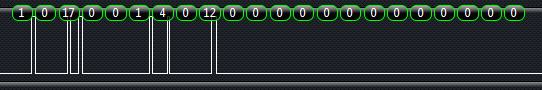
7,2 HCI\_EVNT\_READ\_SP\_VERSION (0x0207)

5 length

0,0,6,1,24 = 1 and 24 are the major and minor versions

According to <http://processors.wiki.ti.com/index.php/CC3000_Release_Notes>, 1.24 is the service pack version, which corresponds to 1.11.1 Release Package version.

Fifth Send:



1 (HCI\_TYPE\_CMND)

4,0 = 0x0004, HCI\_CMND\_WLAN\_IOCTL\_SET\_CONNECTION\_POLICY

12 = length

0,0,0,0 = should\_connect\_to\_open\_ap

0,0,0,0 = ulShouldUseFastConnect

0,0,0,0 = ulUseProfiles

Fifth receive:



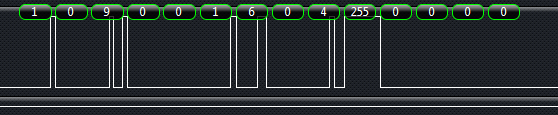
4 HCI\_TYPE\_EVENT

4,0 = 0x0004, HCI\_CMND\_WLAN\_IOCTL\_SET\_CONNECTION\_POLICY

5 length

0,0,0,0 = 32 bit param (sample code ignores)

Sixth send:



1 (HCI\_TYPE\_CMND)

6,0 = HCI\_CMND\_WLAN\_IOCTL\_DEL\_PROFILE (0x0006)

4 = length of params

255,0,0,0 = 0x000000ff (4 byte value) for “index”

0 = padding byte

Sixth receive:



4 HCI\_TYPE\_EVENT

6,0 = 0x0006, HCI\_CMND\_WLAN\_IOCTL\_DEL\_PROFILE (0x0006)

5 length

0,0,0,0 = 32 bit param (sample code ignores these bytes)

Seventh send:



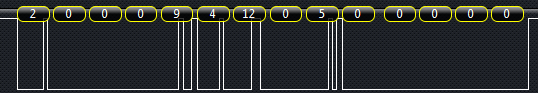
1 (HCI\_TYPE\_CMND)

12,0 = HCI\_CMND\_WLAN\_IOCTL\_SIMPLE\_CONFIG\_SET\_PREFIX (0x000C)

3 = length of params

84,84,84 = [‘T’,’T’,’T’]0

Sevent receive:



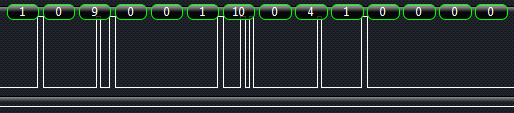
4 HCI\_TYPE\_EVENT

12,0 = HCI\_CMND\_WLAN\_IOCTL\_SIMPLE\_CONFIG\_SET\_PREFIX (0x000C)

5 length

0,0,0,0,0 = sample code ignores these bytes

Eighth send:



1 (HCI\_TYPE\_CMND)

10,0 = HCI\_CMND\_WLAN\_IOCTL\_SIMPLE\_CONFIG\_START (0X000a)

4 = length of params

1,0,0,0 = 4 byte value for algoEncryptedFlag (tells whether info. is encrypted), as LSB, so 0x0001

0 padding byte

Eight receive:



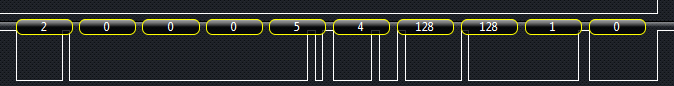
4 HCI\_TYPE\_EVENT

10,0 = HCI\_CMND\_WLAN\_IOCTL\_SIMPLE\_CONFIG\_START (0X000a)

5 length

0,0,0,0,0 = sample code ignores these bytes

Received:



4 HCI\_TYPE\_EVENT

128,128 = HCI\_EVNT\_WLAN\_ASYNC\_SIMPLE\_CONFIG\_DONE (0x8080)

1 length

0 = sample code ignores these bytes

Sent:



1 (HCI\_TYPE\_CMND)

3,2 = HCI\_CMND\_NVMEM\_CREATE\_ENTRY (0X0203)

8 = length of params

12,0,0,0 = FieldID (0x000c)

16,0,0,0 = new length (0x0010)

0 padding byte

Received:



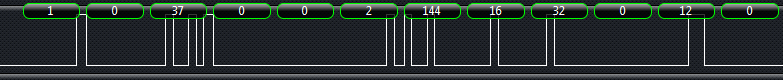
4 HCI\_TYPE\_EVENT

3,2 = HCI\_CMND\_NVMEM\_CREATE\_ENTRY (0X0203)

1 length

0 = sample code ignores these bytes

Sent:



2 = HCI\_TYPE\_DATA

144 = HCI\_CMND\_NVMEM\_WRITE (0x90)

16 = NVMEM\_WRITE\_PARAMS\_LEN (key length here)

32,0 = length of all data after HCI header (so 0x0020 or 32)

12,0,0,0 = FILE ID (0x000c)

12,0,0,0 = (not sure, since it’s hardcoded and uncommented)

16,0,0,0 = length of data (length of encryption key)

0,0,0,0 = “Offset”, which is zero

115,109,97,114,116,99,111,110,102,105,103,65,69,83,49,54 = “smartconfigAES16”

Receive:



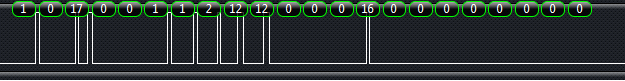
4 = HCI\_TYPE\_EVE NT

0202 = HCI\_EVNT\_NVMEM\_WRITE (0x0202)

5 = length

0,0,0,0,0 ignored

Received:



1 = HCI\_TYPE\_CMND

1,2 = HCI\_CMND\_NVMEM\_READ (0x0201)

12 = length

12,0,0,0 = File ID (NVMEM\_AES128\_KEY\_FILEID)

16,0,0,0 = Length

0,0,0,0 = Offset

0 = padding



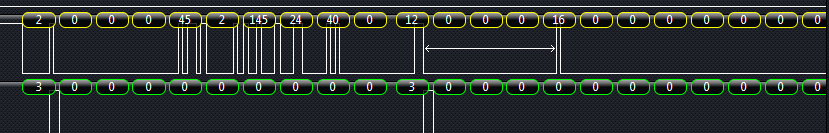
4 = HCI\_TYPE\_EVE NT

1,2 = HCI\_CMND\_NVMEM\_READ (0x0201)

1 = length

0 = status

Received:



2 = HCI\_TYPE\_DATA

145 = HCI\_DATA\_NVMEM

24 = HCI Packet argsize

40,0 = HCI Packet Length as LSB, so 40

12,0,0,0 = FILE ID as LSB, so 12 (NVMEM\_AES128\_KEY\_FILEID)

16,0,0,0 = length

0,0,0,0 = offset

16 more zeros = not sure

115,109,97,114,116,99,111,110,102,105,103,65,69,83,49,54 = “smartconfigAES16”

Next Write:



1 = 1 (HCI\_TYPE\_CMND)

1,2 = 0x0201 = HCI\_CMND\_NVMEM\_READ

12 = length

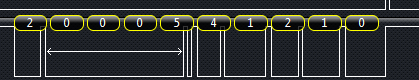
13,0,0,0 = File ID (NVMEM\_SHARED\_MEM\_FILEID)

67,0,0,0 = SMART\_CONFIG\_PROFILE\_SIZE (67)

0,0,0,0 = offset

0 = padding byte

Next Received:



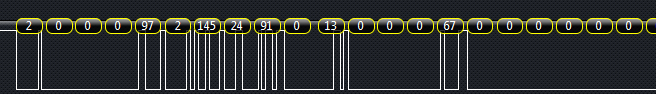
4 = HCI\_TYPE\_EVE NT

1,2 = 0x0201 = HCI\_CMND\_NVMEM\_READ

1 = length

0 = status

Next Received:



2 = HCI\_TYPE\_DATA

145 = HCI\_DATA\_NVMEM

24 = HCI Packet argsize

91,0 = HCI Packet Length as LSB, so 91

13,0,0,0 = File ID (NVMEM\_SHARED\_MEM\_FILEID)

67,0,0,0 = length

0,0,0,0 = offset

16 more zeros = not sure

110,101,114,100,107,105,116, = “nerdkit”

32,3,180,4,251,250,33,211,214,76,222,209,45,201,249,142,104,75,199,190,12,65,58,138,253,235,25,171,42,209,76,10,241,146,209,76,10,241,146,255…(20 255’s)

Next Send:



1 = 1 (HCI\_TYPE\_CMND)

5,0 = 0x0005 = HCI\_CMND\_WLAN\_IOCTL\_ADD\_PROFILE

61 = length

… (includes SSID and password)

Response:

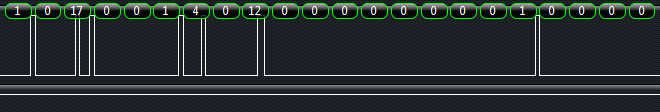


4 = HCI\_TYPE\_EVE NT

5,0 = 0x0005 = HCI\_EVNT\_WLAN\_IOCTL\_ADD\_PROFILE

5 = length

0,0,0,0,0 ?



1 = 1 (HCI\_TYPE\_CMND)

4,0 = 0x0004 = HCI\_CMND\_WLAN\_IOCTL\_SET\_CONNECTION\_POLICY

12 = length