

Technical Note

TecStar

Silicon Labs 社 Zigbee Zigbee NodeTest の使い方

2018 年 6 月

株式会社 **マクニカ**
テクスター カンパニー

目次

1 はじめに	3
2 NodeTest の使い方	4
2-1 NodeTest を EFR32MG にダウンロードする.....	4
2-2 使用できるコマンド	7
2-3 使用例 (PER, RSSI の測定)	10
2-4 使用例 (出力レベルの変更)	12
参考文献	14

1 はじめに

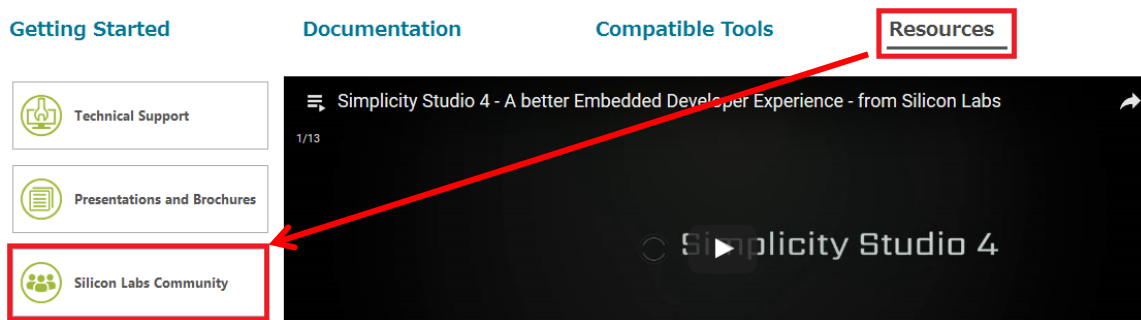
この資料は、Silicon Laboratories（以下、Silicon Labs）社製 Zigbee 対応製品（EFR32MG、MGMxxx）の Zigbee 評価方法について簡易にまとめたものです。内容に誤りがないよう注意は払っておりますが、もし Silicon Labs 社が提供するドキュメント等と差異がございましたら、メーカー提供のものを優先してご参照ください。

また、Silicon Labs 社の ナレッジベース（FAQ）やコミュニティフォーラム（ユーザ同士で問題解決。Silicon Labs のエンジニアも頻繁にコメントしています）には、本資料で取り上げていない様々な情報が記載されております。

製品をご使用頂く過程で疑問や課題が生じることもあると思いますが、他のユーザが既に解決方法を見つけている場合も多々ございます。非常に有益ですので、ぜひご活用下さい。

◆ アクセス方法

Simplicity Studio から



Web Site から

<https://www.silabs.com/community> （Silicon Labs 社製品全般）

<https://www.silabs.com/community/wireless/zigbee-and-thread> （Zigbee/Thread に特化）

◆ 使用方法



2 NodeTest の使い方

NodeTest は、Silicon Labs 社が用意したビルド済みのアプリケーションで、無線の特性評価や、機能試験、ハードウェア検証などにご使用頂くことを想定しています。無線の低レイヤの制御が可能で、以下の作業を行う際に活用頂けます。

- 無線の特性評価
- 製造やスタックのパラメータ設定 (Token)
- 製造後の機能評価
- 無線認証試験を行う際の無線制御

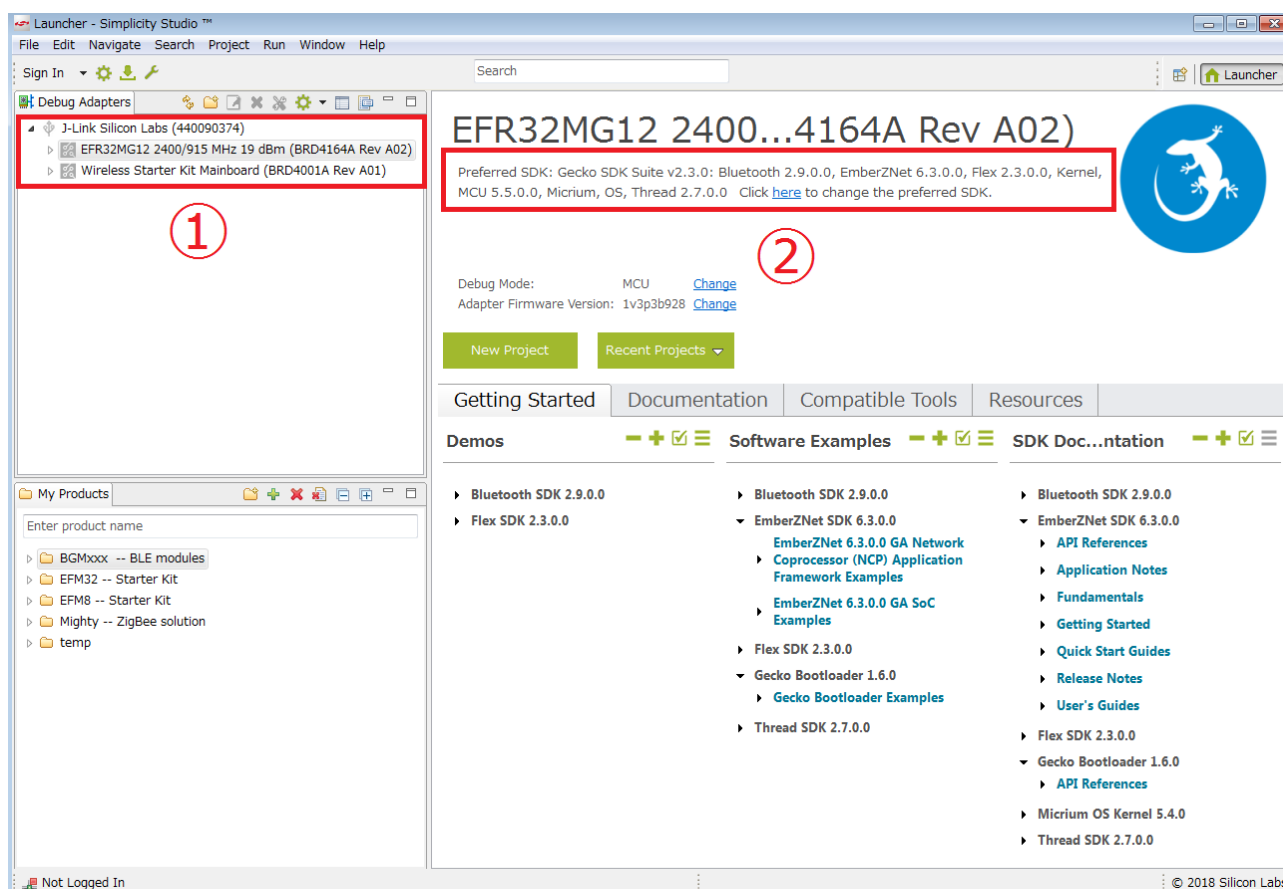
NodeTest についてはアプリケーションノート (AN1019) が用意されておりますので、詳細はそちらをご覧ください。本資料は AN1019 の補足資料です。

<https://www.silabs.com/documents/public/application-notes/AN1019-NodeTest.pdf>

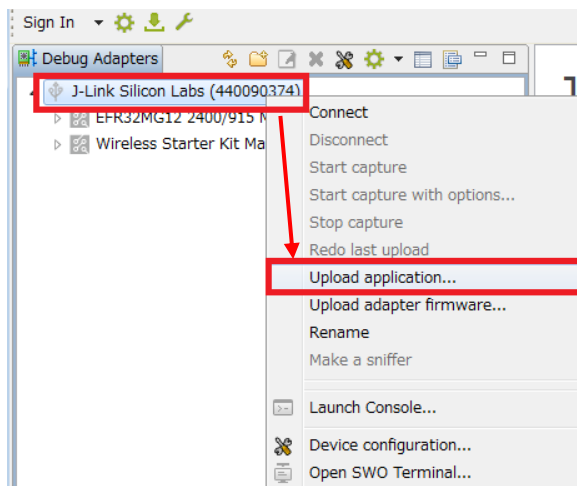
2-1 NodeTest を EFR32MG にダウンロードする

Simplicity Studio を起動し、評価ボードを PC に接続します。

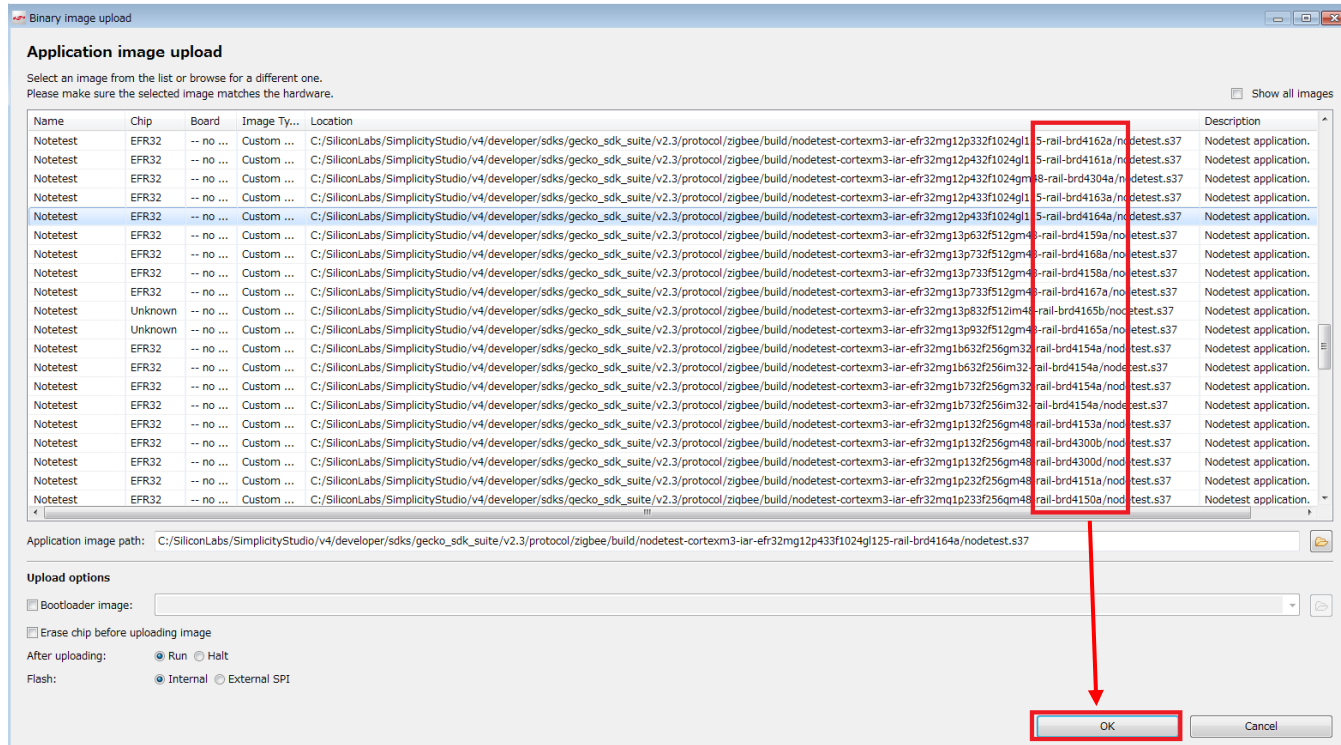
- ① 評価ボードが正しく認識されているか (評価基板の品番を覚えておいてください。後程使います)
 - ② SDK として EmberZNet が選択されているか
- を確認してください。



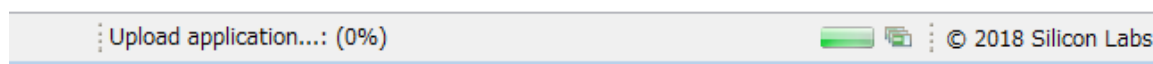
Debug Adapters ウィンドウで J-Link Silicon Labs (xxxx) を選択し、右クリックし、Upload application を選択します。(Debug Adapters ウィンドウは、Simplicity Studio のバージョンによっては Device ウィンドウと表示されています)



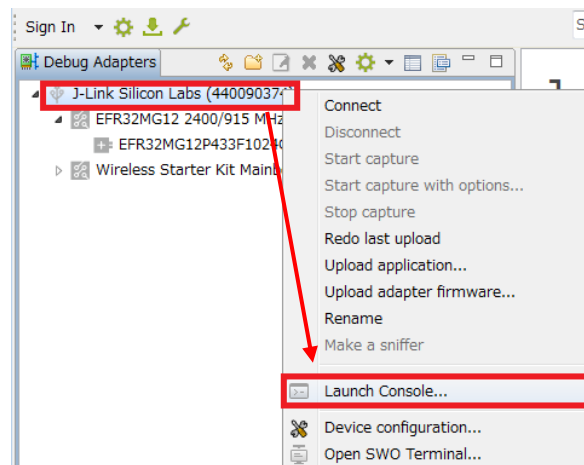
ダウンロードするデータを選択して OK をクリックします。EFR32 向けの NodeTest が多数ありますので、Location の欄を拡げて、ファイル名に使用する評価基板の品番を含んだものを選んでください。下図では BRD4164A 用のデータを選択しています。



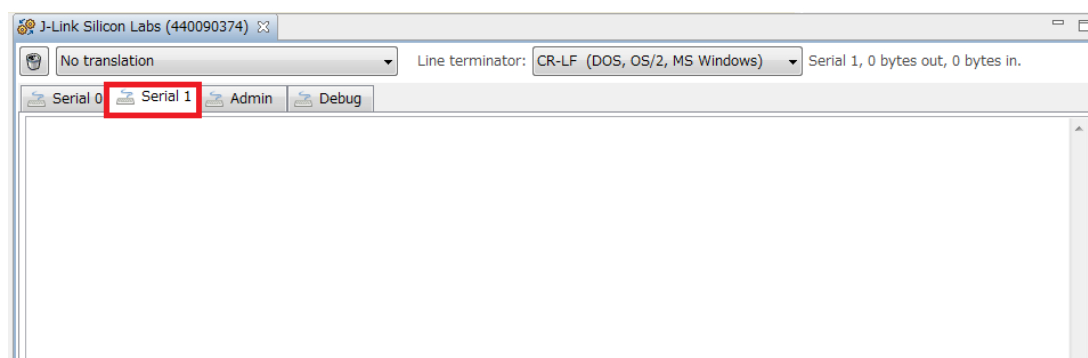
画面下に Upload application と一瞬表示され、ダウンロードが完了します。



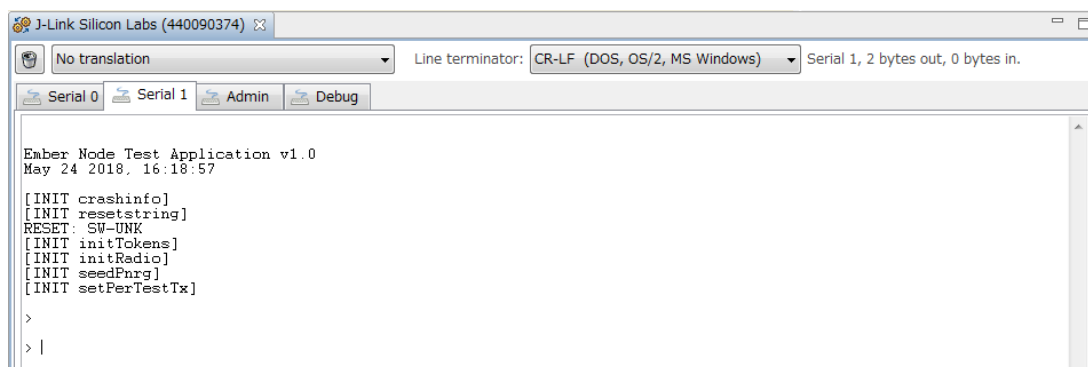
再び Debug Adapters ウィンドウで J-Link Silicon Labs (xxxx) を選択し、右クリックし、Launch Console を選択します。



コンソール画面が起動しますので、Serial 1 タブに切り替えます。

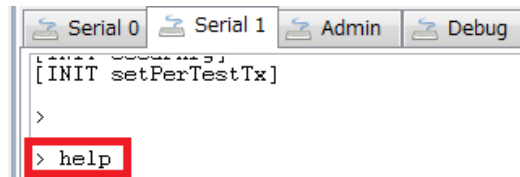


コンソール画面にカーソルを移動し、リターンキーを押すと、コマンド待ち受け状態となります。



2-2 使用できるコマンド

コンソールで `help` と入力します。そうすると、使用できるコマンド一覧が表示されます。



以下がヘルプの内容です。

=====

COMMAND [PARAMETERS] [- DESCRIPTION]

where: b=buffer, s1=int8_t, s2=int16_t, u1=uint8_t, u2=uint16_t, u4=uint32_t

---- button-test ----

button u1 - Test Buttons. Use u1!=0 for polling

crashinfo - Print crash information

---- extflash-test ----

appblver - Report the version of the installed app bootloader

efInit - Initialize the external flash driver

efshutdown - Shutdown the external flash driver

efChipErase - Erase the external flash chip

efPageErase u4u4 - (address len) Erase pages on the external flash chip

efRead u4u2 - (address len) Read data from the external flash chip

efWrite u4u2 - (address len) Write random data to the external flash chip

efWriteTest u4u4u4 - (address len seed) Write random data, lengths to the external flash chip

efReadTest u4u4u4 - (address len seed) Verify random data, lengths read from the external flash chip

efStream - Init and erase ext flash and then accept hex data to be written sequentially

efBootload - Verify image in external flash and then invoke bootloader to install it

---- gpio-test ----

gpioRead - Print GPIO state

gpioSet u4u4* - (port pinmask) or (portpin) Set the masked pins on the given GPIO port high.

gpioOut u4u4 - (port pins) or (portpin value) Set the given GPIO port to the given pin state.

gpioClear u4u4* - (port pinmask) or (portpin) Clear the masked pins on the given GPIO port.

gpioCfg u4u4u4* - (port MODEL MODEH) Set configuration mode for the given GPIO port.

gpioMode u4u4 - (portpin mode) Set configuration mode for the given GPIO port/pin.

gpioSlewrate u1u1u1 - (port slewrate slewrateAlt) Set slewrate and alternate slewrate for the given GPIO port.

---- led-test ----

ledOn u1 - Turns on LED u1

ledOff u1 - Turns off LED u1

ledToggle u1 - Toggles LED u1

---- sleep-test ----

shutdown - Put the chip in the Deep Sleep state

---- sys-test ----

getrev - Retrieve System Revision

---- token-cortexm3-test ----

tokMap - Print the Token Memory Map

loadToks - Load stack and application token defaults

---- reset-test ----

resetstring - Print the ResetString

assertme - Resets the chip using assert()

---- token-test ----

initTokens - Invoke top level token initialization

tokRead u2 - Read all data of a token (u2=creator code)

tokReadIdx u2u1 - Read an index of a token (u2=creator code, u1=idx)

tokWrite u2 - Write all data of a token (u2=creator code)

tokWriteIdx u2u1 - Write an index of a token (u2=creator code, u1=idx)

tokDump - Dump the entire token data set

---- phy-common ----

initRadio - call emRadioInit().

seedPnrg - Seed the PNRG with random data from the radio

setPerTest u1 - (m) Set Packet Error Rate test mode to m (0-disable, 1-enable)

getChannel - Get the current channel

setChannel u1 - (c) Set the radio channel to c

getChanFreq u1 - (c) Display the center frequency for channel c

gRadPowState - Get the radio current power state

sRadPowState u1 - (s) Set the radio current power state to s (0-asleep, 1-awake)

getCca - Get Clear Channel Assessment. 1 = Clear; 0 = Busy.

getRssi - Get RSSI in dBm over 8 symbol periods.

getEd - Get 802.15.4 ED (-100dBm to -36dBm : 0x00 to 0xff) over 8 symbol periods.

edScan u4 - Perform ED scan for n samples.

getCcaThresh - Get CCA-ED threshold in dBm.

setCcaThresh s1 - (e) Set CCA-ED threshold to e dBm.

---- phy-common-pta ----

getPta - Get state of packet traffic arbitration(0-enabled,1-disabled)

setPta u1 - Enable(1)/disable(0) packet traffic arbitration

getPtaOption - Get pta options bit mask

setPtaOption u4 - Set pta options bit mask

setPtaReqArb u1 - Enable(1)/disable(0) packet traffic arbitration read PTA_REQ_GPIO before asserting

---- phy-common-rx ----

showPayload u1 - Enable(1)/disable(0,default) PHY payload display in RX output

rx - Receive mode

rxQuiet - Receive mode but don't print received packets

rxBg u1* - Enable(1)/disable(0)/toggle Receive Background mode

---- phy-common-tx ----

setPerTestTx - Setup TX for IEEE 802.15.4 PER test

getTxPower - Get current radio TX power in dBm

setTxPower s1 - (p) Set radio TX power to p dBm

getTxPowerFl - Get current radio TX power in dBm * 100

setTxPowerFl s2 - (p) Set floating point radio TX power to p dBm * 100 (for 10.25 dbm, input 1025)

getTxDelay - Get TX interpkt delay in us

setTxDelay u2 - (d) Set radio tx interpkt delay to d us

tx u2 - (n) TX n pkts (0=infinite)

txStream - TX continuous stream of random symbols

txTone - TX single carrier frequency

getTxPacket - Display TX pkt length & payload

setTxLength u1 - Set TX payload length

setTxPayload u1u1u1u1u1 - (o d0 d1 d2 d3) Set TX pkt payload data d0-3 at offset o

setCheckCca u1 - Enable(1)/disable(0) HW backoff & CCA

---- phy-em250class ----

calChannel u1 - Set the channel; perform full calibration

setSynOffset s1 - (o) Offset the synth frequency by o 9.375kHz increments.

---- phy-efr32 ----

setCtune u4 - set CTUNE register (CMU->HFXOSTEADYSTAECTRL_CTUNE)

getCtune - get CTUNE register value

---- platformtest ----

version - Prints version information.

verbose u1u1u1u1u1 - Sets verbosity of header command tag delim text

help - Prints this help menu

list b - list commands of specified module name

! - Recalls the previous command

[tab] - Attempts to tab complete command

** NOTE: all input values are in hex!

** Commands are NOT case sensitive

2-3 使用例(PER, RSSI の測定)

PC + Starter Kit を 2 組用意し、PC1 側を RX(受信)、PC2 側を TX(送信)にして、通信を行ってみます。



◆ PC1(RX 側)の設定

rx コマンドを使用して、受信モードにします。コンソールで「 rx 」と入力し、リターンキーを押します。

```

Serial 0 Serial 1 Admin Debug
Ember Node Test Application v1.0
May 24 2018, 16:20:33

[INIT crashinfo]
[INIT resetstring]
RESET:PWR-HV
[INIT initTokens]
[INIT initRadio]
[INIT seedPnrg]
[INIT setPerTestTx]

>
> rx
rx
{{(rx)} test start ('e'nd)}
#{{(rx)}
  {num} {oflo} {seq} {per} {err} {lqi} {rssi}{ed} {gain} {status} {time} {fp}{length}}

```

◆ PC2(TX 側)の設定

tx コマンドを使用して、パケット送信します。コンソールで「 tx 10 」(tx と 10 の間にスペース)と入力し、リターンキーを押します。Tx の後の数字は送信するパケット数です。16 進数ですので、10 は 16 パケットを意味します。

```

Serial 0 Serial 1 Admin Debug
Ember Node Test Application v1.0
May 24 2018, 16:18:57

[INIT crashinfo]
[INIT resetstring]
RESET:PWR-HV
[INIT initTokens]
[INIT initRadio]
[INIT seedPnrg]
[INIT setPerTestTx]

>
> tx 10
tx 10
Txing 16 packets 25500 us apart. 'e'nd...
16 packets transmitted.
Last packet status: 0x0000
Last packet backoff: 0x0000
Frame pending in last ACK: No.
SFD time (via TX complete callback): 0x102A
> |

```

◆ PC1 (RX 側) で結果を確認する

受信したパケット数、パケットエラーレート (PER)、信号強度 (RSSI) などが確認できます。

```

Ember Node Test Application v1.0
May 24 2018, 16:20:33

[INIT crashinfo]
[INIT resetstring]
RESET: PWR-HV
[INIT initTokens]
[INIT initRadio]
[INIT seedPnrg]
[INIT setPerTestTx]

>
> rx
rx
{{{(rx)}} test start ('e'nd)}}
#{{{(rx)}}}
{ {num} {oflo} {seq} {per} {err} {lqi} {rssi} {ed} {gain} {status} {time} {fp} {length}}
{ {1} {0} {1} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x000E1444} {0} {0x12}}
{ {2} {0} {2} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x000E7C94} {0} {0x12}}
{ {3} {0} {3} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x000EE4E8} {0} {0x12}}
{ {4} {0} {4} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x000F4D38} {0} {0x12}}
{ {5} {0} {5} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x000FB589} {0} {0x12}}
{ {6} {0} {6} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x0001DD9} {0} {0x12}}
{ {7} {0} {7} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x00008629} {0} {0x12}}
{ {8} {0} {8} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x0000EE79} {0} {0x12}}
{ {9} {0} {9} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x000156CA} {0} {0x12}}
{ {10} {0} {10} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x0001BF1A} {0} {0x12}}
{ {11} {0} {11} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x0002276A} {0} {0x12}}
{ {12} {0} {12} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x00028FBB} {0} {0x12}}
{ {13} {0} {13} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x0002F80B} {0} {0x12}}
{ {14} {0} {14} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x0003605B} {0} {0x12}}
{ {15} {0} {15} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x0003C8A9} {0} {0x12}}
{ {16} {0} {16} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x000430FC} {0} {0x12}}

```

受信しづらい環境にして、エラーの出方を見てみました。下図は 15 番目のパケットを受信失敗した場合です。

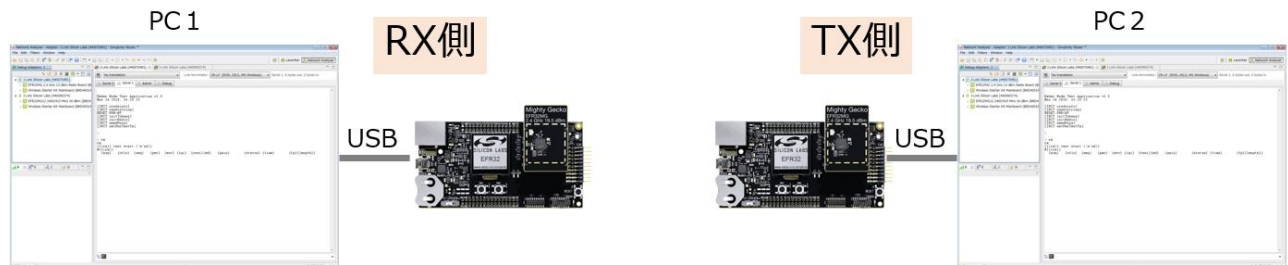
```

> rx
rx
{{{(rx)}} test start ('e'nd)}}
#{{{(rx)}}}
{ {num} {oflo} {seq} {per} {err} {lqi} {rssi} {ed} {gain} {status} {time} {fp} {length}}
{ {1} {0} {1} {0} {64} {0x40} {-84} {0x1C} {0x000000FF} {0x4000} {0x000BBE2E} {0} {0x12}}
{ {2} {0} {2} {0} {64} {0x40} {-84} {0x1C} {0x000000FF} {0x4000} {0x000C2680} {0} {0x12}}
{ {3} {0} {3} {0} {68} {0x44} {-83} {0x21} {0x000000FF} {0x4000} {0x000C8ED2} {0} {0x12}}
{ {4} {0} {4} {0} {64} {0x40} {-84} {0x1C} {0x000000FF} {0x4000} {0x000CF723} {0} {0x12}}
{ {5} {0} {5} {0} {64} {0x40} {-84} {0x1C} {0x000000FF} {0x4000} {0x000D5F73} {0} {0x12}}
{ {6} {0} {6} {0} {64} {0x40} {-84} {0x1C} {0x000000FF} {0x4000} {0x000DC7C3} {0} {0x12}}
{ {7} {0} {7} {0} {64} {0x40} {-84} {0x1C} {0x000000FF} {0x4000} {0x000E3014} {0} {0x12}}
{ {8} {0} {8} {0} {60} {0x3C} {-85} {0x17} {0x000000FF} {0x4000} {0x000E9864} {0} {0x12}}
{ {9} {0} {9} {0} {60} {0x3C} {-85} {0x17} {0x000000FF} {0x4000} {0x000F00B4} {0} {0x12}}
{ {10} {0} {10} {0} {60} {0x3C} {-85} {0x17} {0x000000FF} {0x4000} {0x000F6904} {0} {0x12}}
{ {11} {0} {11} {0} {60} {0x3C} {-85} {0x17} {0x000000FF} {0x4000} {0x000FD155} {0} {0x12}}
{ {12} {0} {12} {0} {64} {0x40} {-84} {0x1C} {0x000000FF} {0x4000} {0x000039A3} {0} {0x12}}
{ {13} {0} {13} {0} {64} {0x40} {-84} {0x1C} {0x000000FF} {0x4000} {0x0000A1F3} {0} {0x12}}
{ {14} {0} {14} {0} {64} {0x40} {-84} {0x1C} {0x000000FF} {0x4000} {0x00010A43} {0} {0x12}}
{ {15} {0} {16} {6} {68} {0x44} {-83} {0x21} {0x000000FF} {0x4000} {0x0001DAE4} {0} {0x12}}

```

2-4 使用例(出力レベルの変更)

PC + Starter Kit を 2 組用意し、PC1 側を RX(受信)、PC2 側を TX(送信)にして、通信を行ってみます。
この状態で、PC2(Tx 側)の出力レベルを変更してみます。



◆ PC1(RX 側)の設定

rx コマンドを使用して受信モードにします。

◆ PC2(TX 側)の設定

getTxPowerFl コマンドを使用して、現在の出力設定を確認します。コンソールで「 getTxPowerFl 」(Fl はエフエルです)と入力し、リターンキーを押します。0.1dBm ステップですので、現在は 61...つまり 6.1 dBm の設定になっているようです。

```
Serial 0 Serial 1 Admin Debug
> getTxPowerFl
getTxPowerFl
{{{getTxPowerFl}}} {actualPower: 61}dBm
>
```

この状態で、tx コマンドを使ってパケット送信します。

```
Serial 0 Serial 1 Admin Debug
> tx 5
tx 5
Txing 5 packets 25500 us apart. 'e'nd...
5 packets transmitted.
Last packet status: 0x0000
Last packet backoff: 0x0000
Frame pending in last ACK: No.
SFD time (via TX complete callback): 0xB170
>
```

◆ PC1(RX 側)で結果を確認する

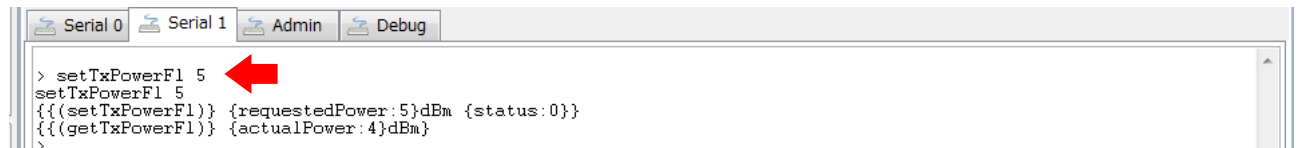
信号強度 RSSI を見ると、-43dBm となっています。

```
Serial 0 Serial 1 Admin Debug
> rx
rx
{{{rx}}} test start ('e'nd)}
#{{{rx}}}
{num} {oflo} {seq} {per} {err} {lqi} {rssi} {ed} {gain} {status} {time} {fp} {length}
{1} {0} {1} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x000FE02E} {0} {0x12}
{2} {0} {2} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x00004883} {0} {0x12}
{3} {0} {3} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x0000B0D5} {0} {0x12}
{4} {0} {4} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x00011923} {0} {0x12}
{5} {0} {5} {0} {228} {0xE4} {-43} {0xDE} {0x000000FF} {0x4000} {0x00018173} {0} {0x12}
```

では、次に出力レベルを変更してみます。(5.5dB 程度下げてみます)

◆ PC2(TX 側)の設定

setTxPowerFl コマンドを使用して、出力設定を変更します。コンソールで「 setTxPowerFl 5」(Fl はエフェル)と入力し、リターンキーを押します。0.1dBm ステップなので、5...つまり 0.5dBm に設定しようとしたのですが、実現できる設定値との兼ね合いで 4...つまり 0.4dBm に設定されました。

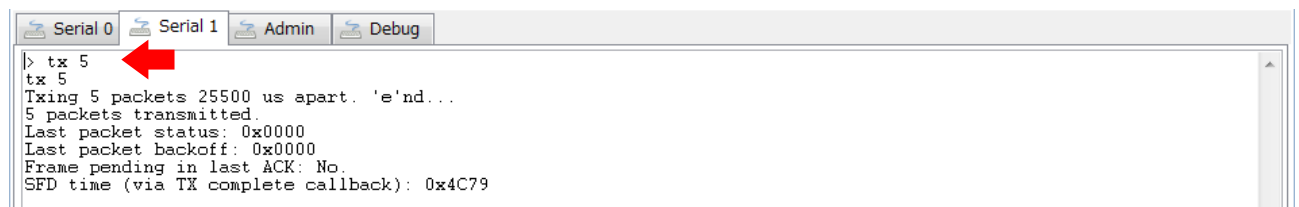


```

> setTxPowerFl 5
setTxPowerFl 5
{{{setTxPowerFl}}} {requestedPower:5}dBm {status:0}}
{{{getTxPowerFl}}} {actualPower:4}dBm
>

```

この状態で、tx コマンドを使ってパケット送信します。



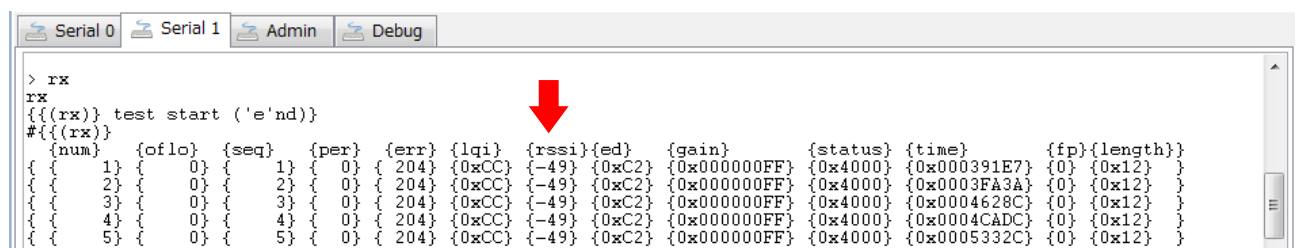
```

> tx 5
tx 5
Txing 5 packets 25500 us apart. 'e'nd...
5 packets transmitted.
Last packet status: 0x0000
Last packet backoff: 0x0000
Frame pending in last ACK: No.
SFD time (via TX complete callback): 0x4C79

```

◆ PC1(RX 側)で結果を確認する

信号強度 RSSI を見ると、-49dBm となっており、設定どおり出力レベルが下がったことが確認できます。



```

> rx
rx
{{{rx}}} test start ('e'nd)}
#{{{rx}}}
{ {num} {oflo} {seq} {per} {err} {lqi} {rssi} {ed} {gain} {status} {time} {fp} {length}}
{ { 1} { 0} { 1} { 0} { 204} {0xCC} {-49} {0xC2} {0x000000FF} {0x4000} {0x000391E7} {0} {0x12} }
{ { 2} { 0} { 2} { 0} { 204} {0xCC} {-49} {0xC2} {0x000000FF} {0x4000} {0x0003FA3A} {0} {0x12} }
{ { 3} { 0} { 3} { 0} { 204} {0xCC} {-49} {0xC2} {0x000000FF} {0x4000} {0x0004628C} {0} {0x12} }
{ { 4} { 0} { 4} { 0} { 204} {0xCC} {-49} {0xC2} {0x000000FF} {0x4000} {0x0004CADC} {0} {0x12} }
{ { 5} { 0} { 5} { 0} { 204} {0xCC} {-49} {0xC2} {0x000000FF} {0x4000} {0x0005332C} {0} {0x12} }

```

改版履歴

Version	改定日	改定内容
1.0	2018 年 06 月	・新規作成。マクニカオンラインで公開

参考文献

- Silicon Labs 社 各種ドキュメント
- Silicon Labs 社 ナレッジベース、コミュニティフォーラム