

Renesas R5F565NE(CK-RX65N)用サンプル

(e2studio RX65N_gnu_dhcp_udp_aes_gt202)の説明

(e2studio Version:2024-04 / FreeRTOS Version 10.4.3-rx-1.0.8)

1. Sample の免責について

- **Sample** に関する Tel/Fax でのご質問に関してはお受けできません。ただし、メールでのご質問に関してはお答えするよう努力はしますが、都合によりお答えできない場合もありますので予めご了承願います。
- **Sample** ソフトの不具合が発見された場合の対応義務はありません。また、この関連ソフトの使用方法に関する質問の回答義務もありませんので承知の上ご利用下さい。
- **Sample** ソフトは、無保証で提供されているものであり、その適用可能性も含めて、いかなる保証も行いません。また、本ソフトウェアの利用により直接的または間接的に生じたいかなる損害に関しても、その責任を負わないものとします。

2. サンプルのプロジェクト名

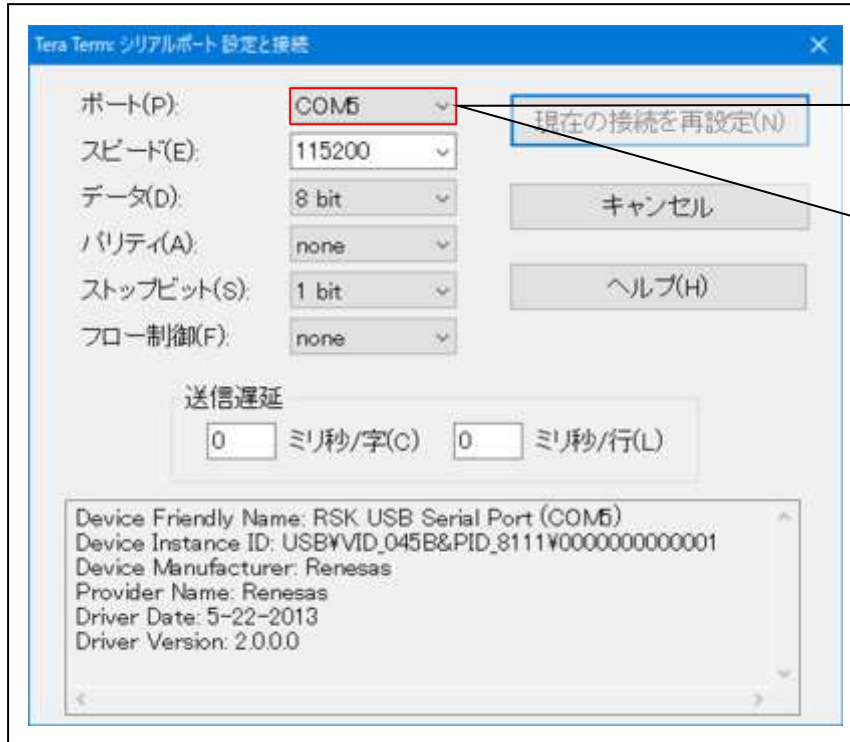
| ワークスペース名 | 概要 | プロジェクト名 |
|-----------------------------|--|--|
| FreeRTOS_sample_gnu_wifi_CK | 無線 WiFi-module(GT202-QCA4002)を使用した DHCP と UDP 通信のサンプル セキュリティ API Crypto ライブラリ(Mbed TLS)AES-CBC を使用したサンプル (暗号・復号) | RX65N_gnu_dhcp_udp_aes_gt202b FreeRTOS モードで動作 FreeRTOS-Plus-TCP (dhcp_client) UDP 通信(Client) (FreeRTOS_sendto..) (FreeRTOS_recvfrom...) 暗号・復号(AES-CBC) (MBEDTLS_AES_ENCRYPT) (MBEDTLS_AES_DECRYPT) |

| |
|--|
| 統合開発環境 |
| Renesas e2studio(Version 2024-04) |
| FreeRTOS (Version 10.4.3-rx 1.0.8) |
| FreeRTOS-Plus-TCP(Version4.0.0) |
| Mbed TLS(Version3.5.2) |
| GCC for Renesas RX(Version 8.3.0.202405) |
| テキストファイル・エンコード(sjis) |

| |
|------------------|
| ハード環境 |
| CK-RX65N (ルネサス製) |

3. Tera Term Pro のインストール

- ① 「teraterm-4.106.exe」 を検索してダウンロードする。
- ② PC にインストールし実行する
- ③ シリアルポートの設定

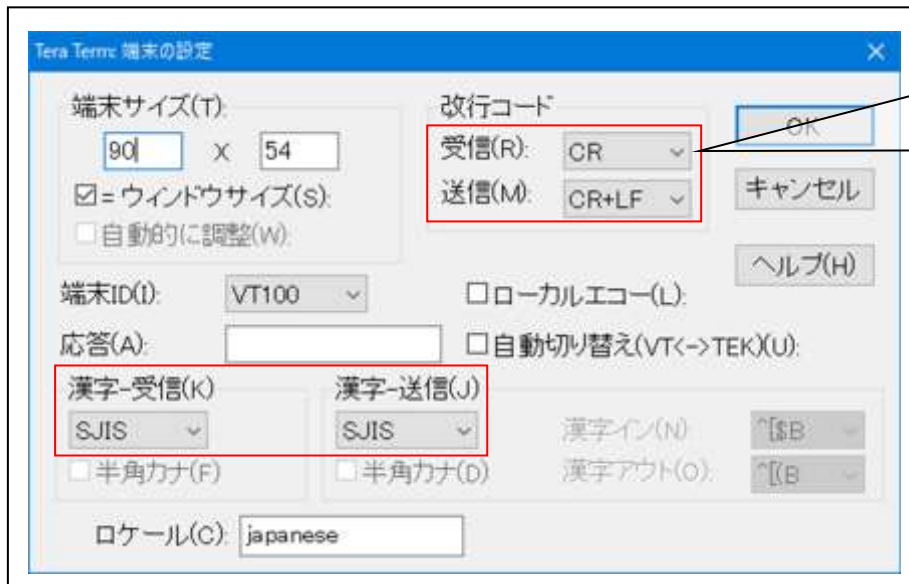


COM 番号は、
PC 側でシリアル通信可能な番号を指定する。

115200BPS
8bit
none
1bit
none

の仕様にする。

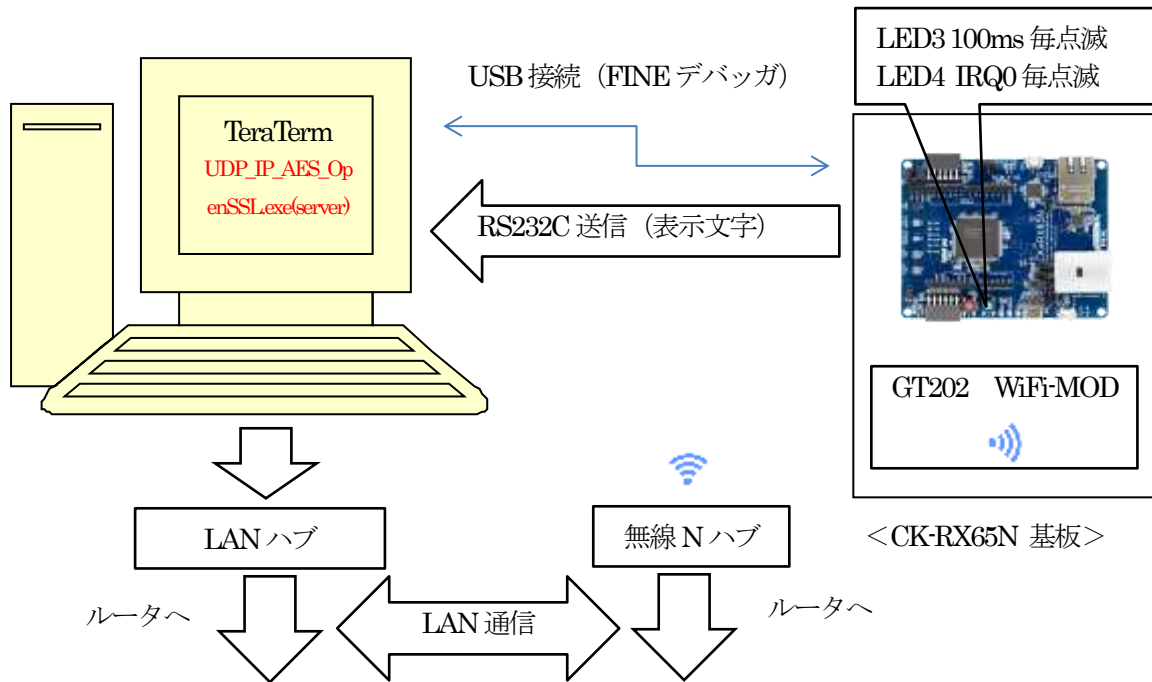
④ 端末の設定



USB シリアルコンバータ使用時に CR コードがカットされる設定の場合は、**受信: LF** にして下さい。

赤枠の設定にする。

4. 動作構成

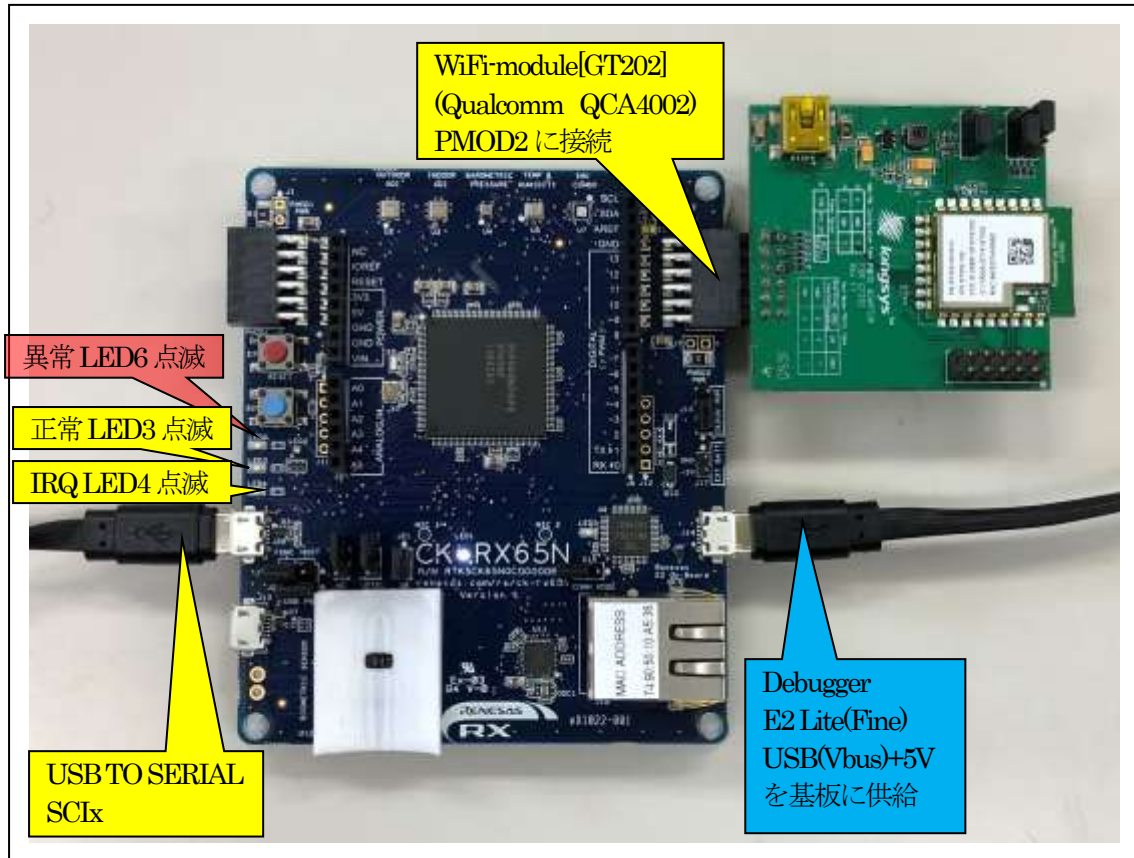


<UDP_IP_AES_OpenSSL.exe>
【Plain mode】
 1. 平文を受信する。
 2. 受信した平文をそのまま送信する。

【AES-CBC mode】
 1. 暗号文を受信する。
 2. 受信した暗号文を復号して表示する。
 3. 復号文を暗号化した文章を送信する。

<RX65N_gnu_dhcp_udp_aes_gt202>
【Plain mode】
 1. 平文を送信する。
 2. 受信した平文を TeraTerm に表示する。

【AES-CBC mode】
 1. AES(CBC)暗号化した文章を送信する。
 2. 受信した暗号文を復号して TeraTerm に表示する。



| ジャンパ | | 備考 |
|------|------------|--|
| J2 | ショート | Current Measurement point for MCU |
| J15 | オープン | Select debugger comms mode |
| J16 | 1 - 2 ショート | DEBUG |
| J21 | ショート | Enable USB boot mode |
| J22 | オープン | Select USB boot mode power supply method |
| J11 | オープン | Configures the MCU for normal boot mode |

5. 「RX65N_gnu_dhcp_udp_aes_gt202」 サンプルの説明

5-1. フォルダ構成とファイル名【<ホルダ名>を示す】

| <FreeRTOS_sample_gnu_wifi_CK> | | |
|-----------------------------------|--|--------------------------|
| <RX65N_gnu_dhcp_udp_aes_gt202> | DHCP / UDP 通信/AES-CBC(暗号・復号) サンプルプロジェクト | |
| <HardwareDebug> | RX65N_gnu_dhcp_udp_aes_gt202.elf | ELF ファイル、デバッガで使用 |
| | RX65N_gnu_dhcp_udp_aes_gt202.map | MAP ファイル、アドレス情報 |
| | RX65N_gnu_dhcp_udp_aes_gt202.mot | モトローラーHEX ファイル |
| | その他 | 自動生成ファイル |
| <src> | <FreeRTOS> | RTOS カーネル |
| | <FreeRTOS-Plus-TCP> | TCP/IP スタック |
| | <mbedtls_development> | 暗号・復号ライブラリ |
| | <AWS_aes> | AES-CBC |
| | aws_aes.c | 暗号・復号処理のソース |
| | aws_aes.h | aws_aes.c のヘッダー |
| | <fRTOS_config> | 未使用 |
| | <fRTOS_skeleton> | |
| | udp_aes_task.c | UDP 通信処理のソース |
| | task_function.h | tcp_aes_task.c のヘッダー |
| | <fRTOS_startup> | スタートアップ |
| | freertos_object_int.c | tcp_aes_task の作成 |
| | freertos.start.c | CMT0 の初期化と MAIN_TASK の起動 |
| | freertos.start.h | reertos.start.c のヘッダー |
| | <smc_gen> | スマートコンフィグレータにより作成 |
| | <wifi_gt202> | GT202 用ドライバソース一式 |
| | demo_printf.c | コンソール入出力処理のソース |
| | demo_printf.h | demo_printf.c のヘッダー |
| | hardware_setup.c | 周辺 I/O デバイス初期化ソース |
| | hardware_setup.h | hardware_setup.c のヘッダー |
| | main_task.c | メインタスク |
| | sf_wifi_nsal_api.h | wifi コントロールのヘッダ |
| | FreeRTOSConfig.h | FreeRTOS のコンフィグレーションファイル |
| FreeRTOSIPConfig.h | FreeRTOS+TCP のコンフィグレーションファイル | |
| linker_script.ld | リンカ用スクリプトファイル | |
| RX65N_gnu_dhcp_udp_aes_gt202.scfg | スマートコンフィグレータの管理ファイル | |
| その他 | 自動生成ファイル | |

5-2. Macro Defines の説明

| Macro Name | 値 | 説明 |
|------------------|---|--|
| ipconfigUSE_DHCP | 0 | DHCP Client Disable ◎ヘッダーファイルに直接 IP アドレスを記述 FreeRTOSConfig.h : <pre> /* IP address configuration. */ #define configIP_ADDR0 192 #define configIP_ADDR1 168 #define configIP_ADDR2 21 #define configIP_ADDR3 95 </pre> |
| | 1 | DHCP Client Enable |

5-3. サンプルの動作説明（基板側 CK-RX65N）

1) DHCP 無効時 (ipconfigUSE_DHCP = 0)

Term 画面

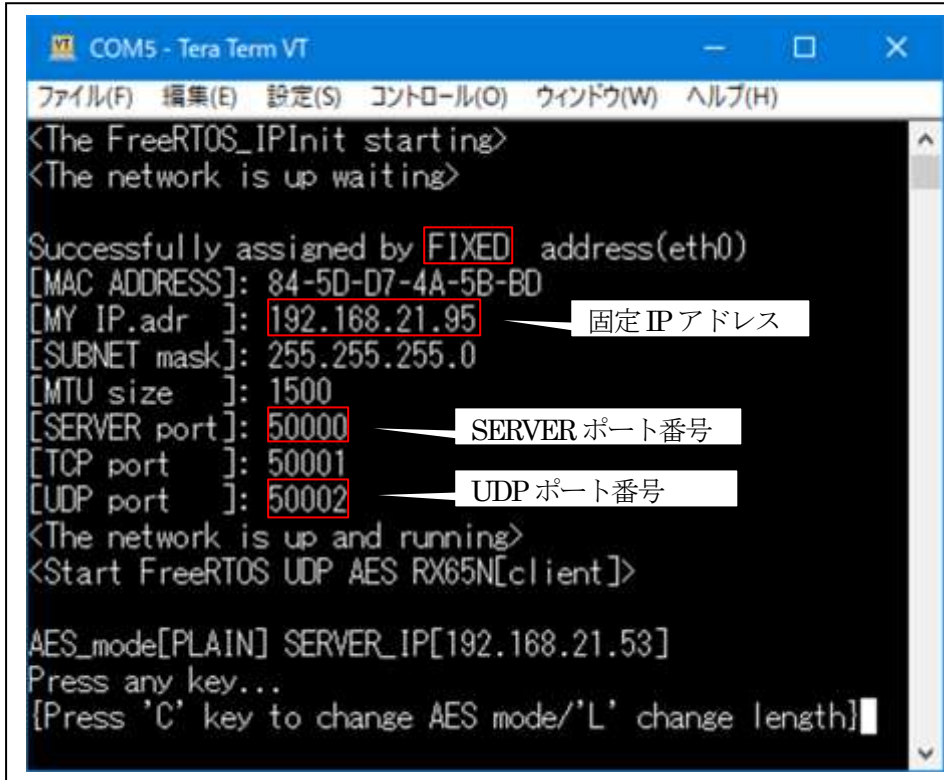
< 1 > 「The FreeRTOS_IPInit starting」

< 2 > 「The network is up waiting」

<成功画面>

IP アドレス確立により、

基板上的 LED3（緑色）を 100msec 毎に点滅・LED4（青色）IRQ0 割り込み毎に点滅



```

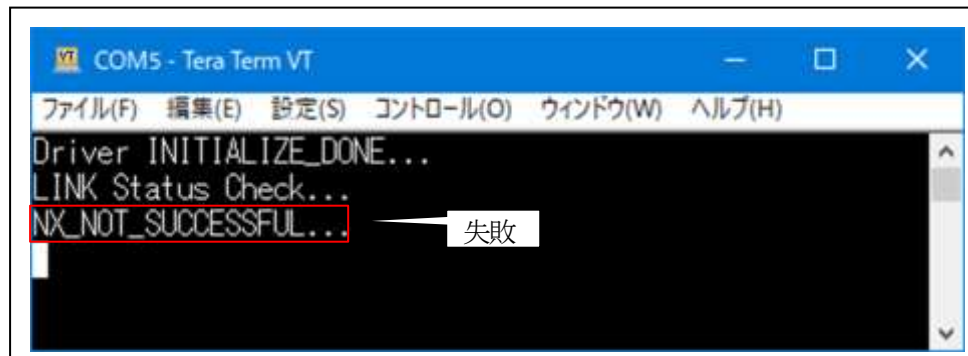
COM5 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
<The FreeRTOS_IPInit starting>
<The network is up waiting>

Successfully assigned by FIXED address(eth0)
[MAC ADDRESS]: 84-5D-D7-4A-5B-BD
[MY IP.adr ]: 192.168.21.95
[SUBNET mask]: 255.255.255.0
[MTU size ]: 1500
[SERVER port]: 50000
[TCP port ]: 50001
[UDP port ]: 50002
<The network is up and running>
<Start FreeRTOS UDP AES RX65N[client]>

AES_mode[PLAIN] SERVER_IP[192.168.21.53]
Press any key...
[Press 'C' key to change AES mode/'L' change length]
  
```

<失敗画面>

IP アドレス未確立により、基板上的 LED6（赤色）を 100msec 毎に点滅



```

COM5 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
Driver INITIALIZE_DONE...
LINK Status Check...
NX_NOT_SUCCESSFUL...
  
```

2) DHCP 有効時 (ipconfigUSE_DHCP = 1)

Term 画面

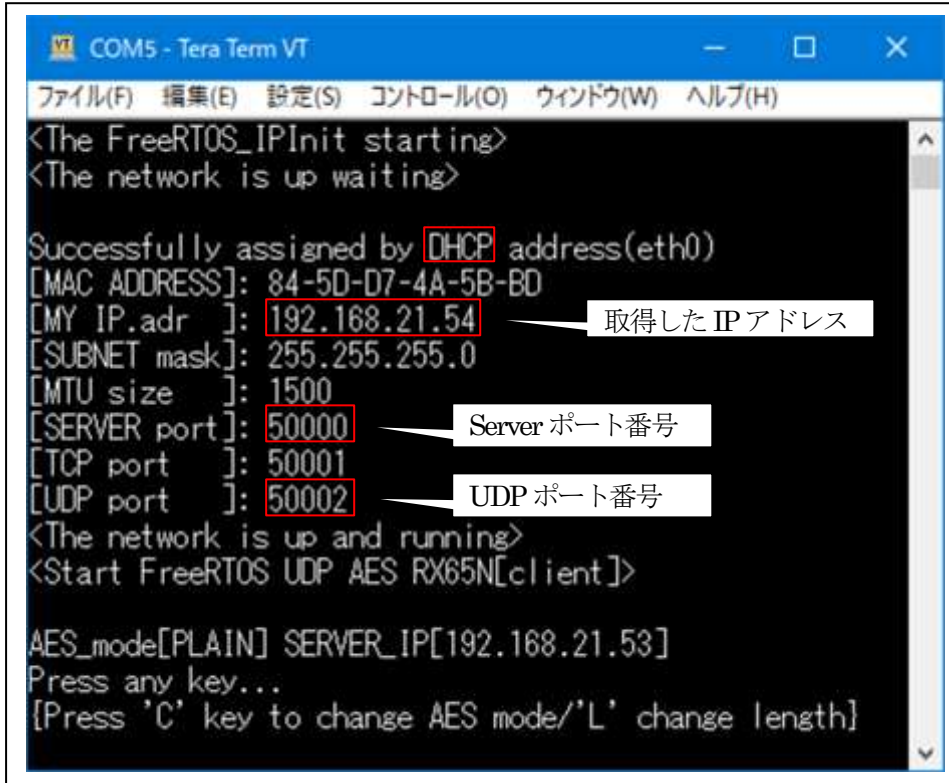
< 1 > 「<The FreeRTOS_IPInit starting>」

< 2 > 「<The network is up waiting>」

<成功画面>

IP アドレス確立により、

基板上の LED3 (緑色) を 100msec 毎に点滅・LED4 (青色) IRQ0 割り込み毎に点滅

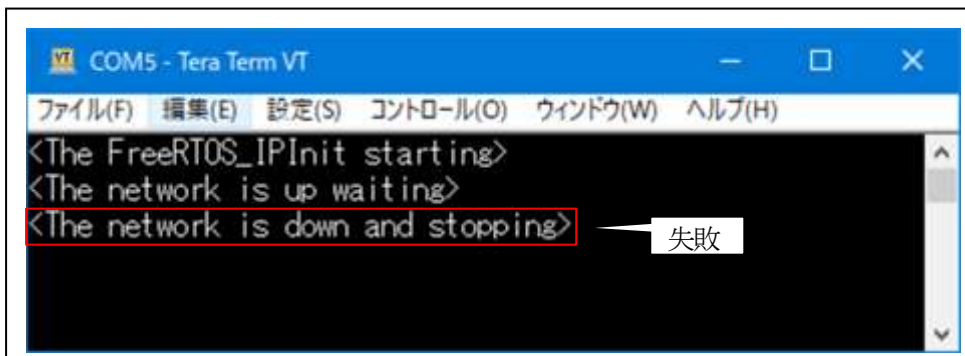


```

COM5 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
<The FreeRTOS_IPInit starting>
<The network is up waiting>
Successfully assigned by DHCP address(eth0)
[MAC ADDRESS]: 84-5D-D7-4A-5B-BD
[MY IP.adr ]: 192.168.21.54
[SUBNET mask]: 255.255.255.0
[MTU size ]: 1500
[SERVER port]: 50000
[TCP port ]: 50001
[UDP port ]: 50002
<The network is up and running>
<Start FreeRTOS UDP AES RX65N[client]>
AES_mode[PLAIN] SERVER_IP[192.168.21.53]
Press any key...
[Press 'C' key to change AES mode/'L' change length]
  
```

<失敗画面>

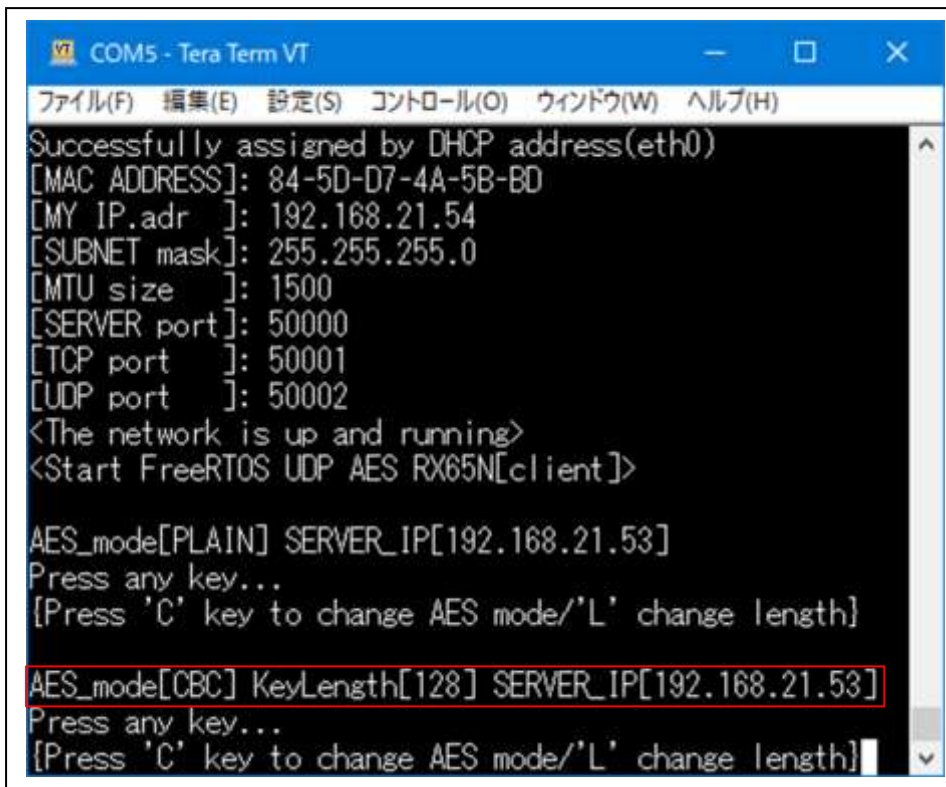
IP アドレス未確立により、基板上の LED6 (赤色) を 100msec 毎に点滅



```

COM5 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
<The FreeRTOS_IPInit starting>
<The network is up waiting>
<The network is down and stopping>
  
```


3) UDP/IP 送受信
 <UDPAESThread>



```

COM5 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
Successfully assigned by DHCP address(eth0)
[MAC ADDRESS]: 84-5D-D7-4A-5B-BD
[MY IP,adr ]: 192.168.21.54
[SUBNET mask]: 255.255.255.0
[MTU size ]: 1500
[SERVER port]: 50000
[TCP port ]: 50001
[UDP port ]: 50002
<The network is up and running>
<Start FreeRTOS UDP AES RX65N[client]>

AES_mode[PLAIN] SERVER_IP[192.168.21.53]
Press any key...
[Press 'C' key to change AES mode/'L' change length]

AES_mode[CBC] KeyLength[128] SERVER_IP[192.168.21.53]
Press any key...
[Press 'C' key to change AES mode/'L' change length]
  
```

| 表示項目 | 表示内容 | 説明 |
|------------------------|-------------------|--|
| AES_mode[x] | PLAIN CBC | 送受信モードの指定 ◎ “C” キー入力によりモード変更 PLAIN->CBC->PLAIN ◎変数のフラグにより指定 aes.c : int AES_crypto_mode; 0 : PLAIN //平文モード 1 : CBC //AES-CBC モード暗号・復号 |
| KeyLength[x] | 128 192 256 | AES-CBC モード時の Key ビット時の指定 ◎ “L” キー入力により Key ビット変更 128->192->256->128 ◎変数の数値により指定 aes.c : int AES_crypto_bit; 128 : Key ビット長が 128bit 192 : Key ビット長が 192bit 256 : Key ビット長が 256bit |
| SERVER_IP[xxx.xxx.xxx] | 固定 | 送信先(PC 側)IP アドレス ◎define にて指定 tcp_aes_thread_entry.c : #define SERVER_IP IP_ADDRESS(192,168,xx,xx) |

5-4. Windows PC 側のテストプログラムで動作確認 (PLAIN (平文) モード)

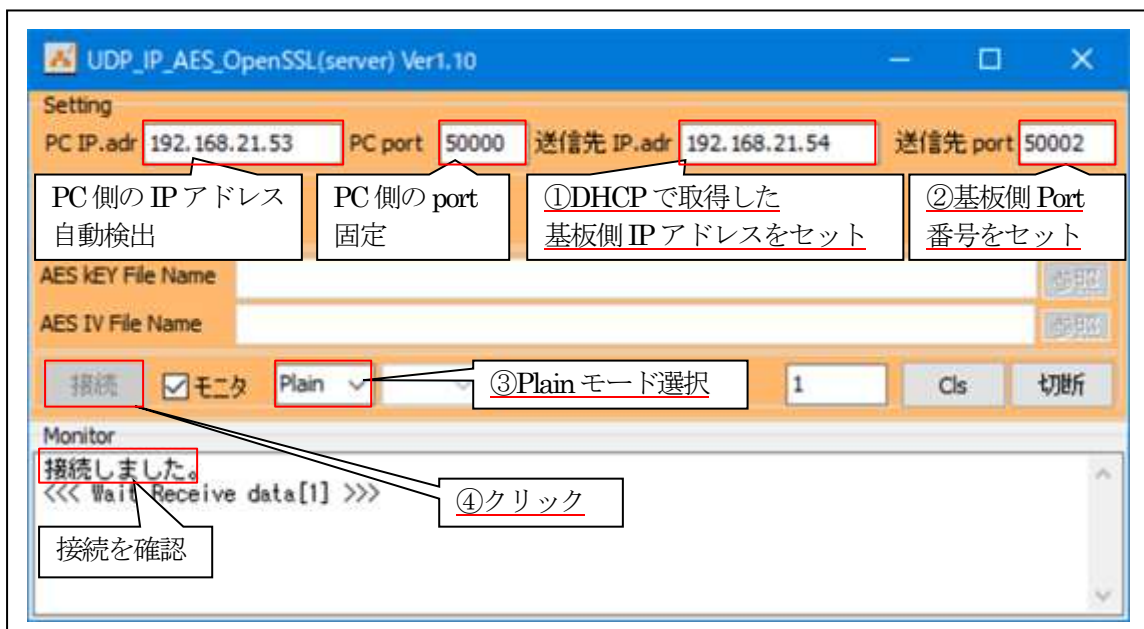
1) 「UDP_IP_AES_OpenSSL.exe」を起動する。(各モード共通)

プログラム場所【¥_PC_Test_OpenSSL】 サンプルの解凍ホルダ



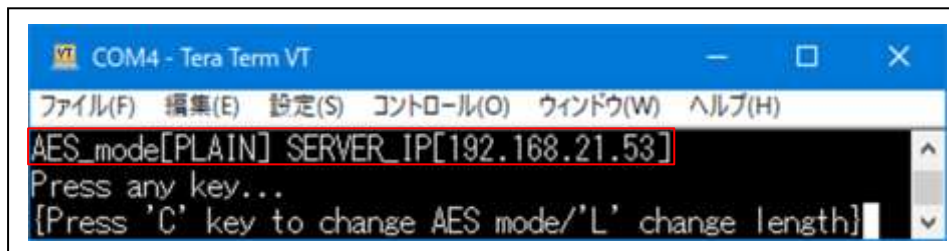
「UDP_IP_AES_OpenSSL.exe」を起動

2) 「UDP_IP_AES_OpenSSL」の各項目を設定して接続する。

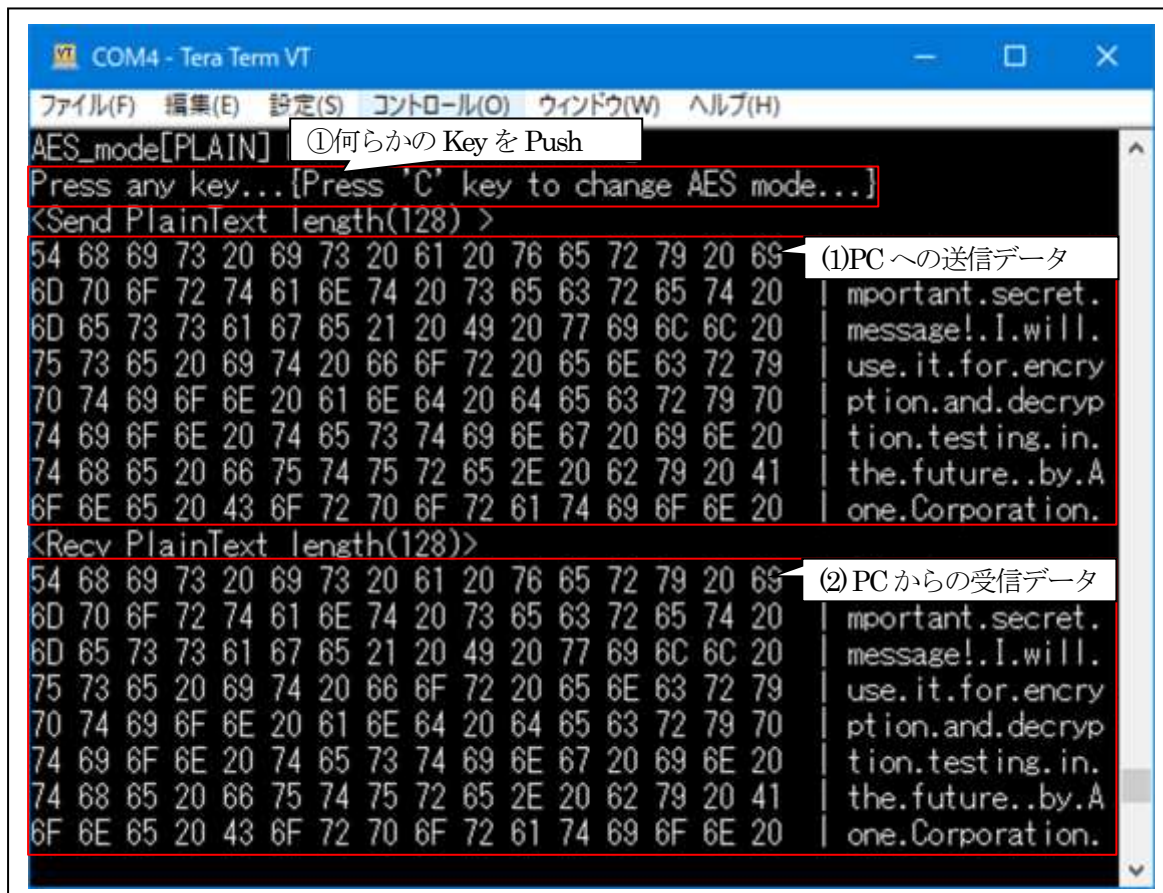


3) 基板側の各項目の確認と設定。

| 表示項目 | 説明 |
|--------------------------|--|
| AES_mode[PLAIN] | 送受信モードの指定 ◎変数のフラグにより指定 aes.c : int AES_crypto_mode = PLAIN; //0=PLAIN |
| SERVER_IP[192.168.21.53] | 送信先(PC側)IPアドレス ◎defineにて指定 udp_aes_thread_entry.c : #define SERVER_IP IP_ADDRESS(192,168,21,53) |



4) 基板側から PC(server)側へ平文テキストを送受信する。


 ☆受信処理が失敗した場合は「**Ctrl+C**」Key-Push で中断する。

5) 基板側の原文保存場所 (各モード共通)

| モジュール名 | 変数名 |
|------------------------|---|
| udp_aes_thread_entry.c | <pre>static UCHAR *src_str={ //テスト用原文 "This is a very important secret message!" "I will use it for encryption and decryption testing" "in the future. by Aone Corporation " };</pre> |

6) 「UDP_IP_AES_OpenSSL」側の送受信を確認する。

The screenshot shows the application window with the following settings:

- PC IP.adr: 192.168.21.53
- PC port: 50000
- 送信先 IP.adr: 192.168.21.54
- 送信先 port: 50002
- AES key File Name: (empty)
- AES IV File Name: (empty)
- Mode: Plain
- Packet size: 2

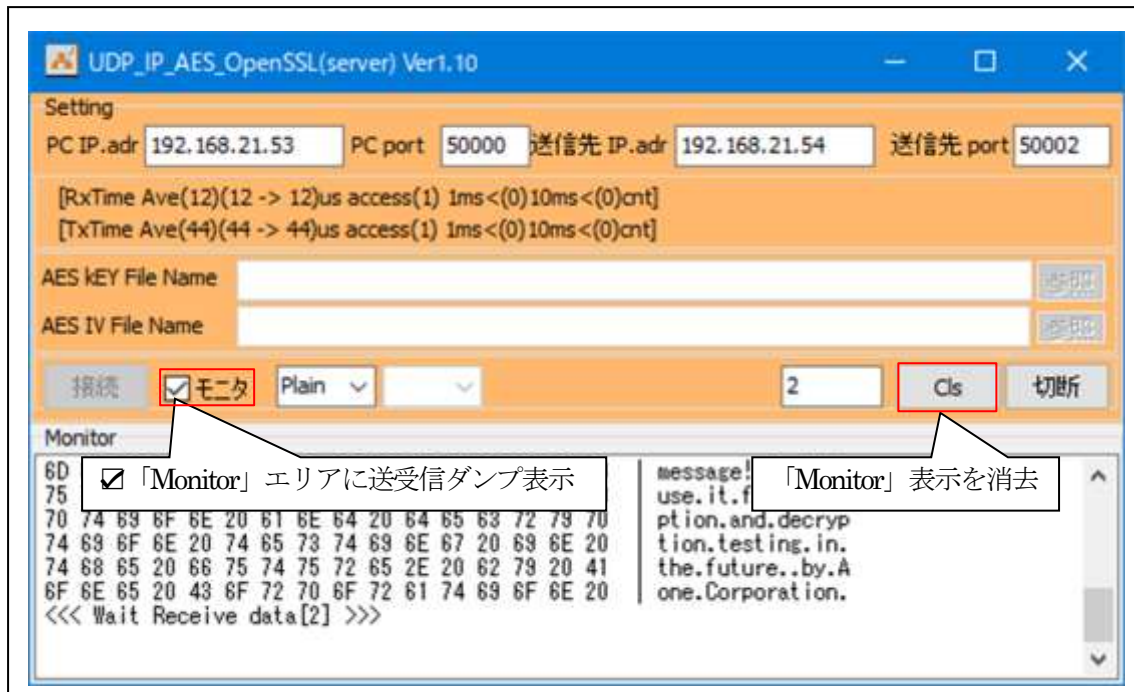
The Monitor section shows the following log entries:

```
<<< Wait Receive data[1] >>>
-(1)Receive plain_data from client
54 68 69 73 20 69 73 20 61 20 76 65 72 79 20 69 | This.is.a.very.i
6D 70 6F 72 74 61 6E 74 20 73 65 63 72 65 74 20 | mportant.secret.
6D 65 73 73 61 67 65 21 20 49 20 77 69 6C 6C 20 | message!.I.will.
75 73 65 20 69 74 20 66 6F 72 20 65 6E 63 72 79 | use.it.for.ency
70 74 69 6F 6E 20 61 6E 64 20 64 65 63 72 79 70 | ption.and.decry
74 69 6F 6E 20 74 65 73 74 69 6E 67 20 69 6E 20 | tion.testing.in.
74 68 65 20 66 75 74 75 72 65 2E 20 62 79 20 41 | the.future..by.A
6F 6E 65 20 43 6F 72 70 6F 72 61 74 69 6F 6E 20 | one.Corporation.
-(2)Send plain_data to client
54 68 69 73 20 69 73 20 61 20 76 65 72 79 20 69 | This.is.a.very.i
6D 70 6F 72 74 61 6E 74 20 73 65 63 72 65 74 20 | mportant.secret.
6D 65 73 73 61 67 65 21 20 49 20 77 69 6C 6C 20 | message!.I.will.
75 73 65 20 69 74 20 66 6F 72 20 65 6E 63 72 79 | use.it.for.ency
70 74 69 6F 6E 20 61 6E 64 20 64 65 63 72 79 70 | ption.and.decry
74 69 6F 6E 20 74 65 73 74 69 6E 67 20 69 6E 20 | tion.testing.in.
74 68 65 20 66 75 74 75 72 65 2E 20 62 79 20 41 | the.future..by.A
6F 6E 65 20 43 6F 72 70 6F 72 61 74 69 6F 6E 20 | one.Corporation.
<<< Wait Receive data[2] >>>
```

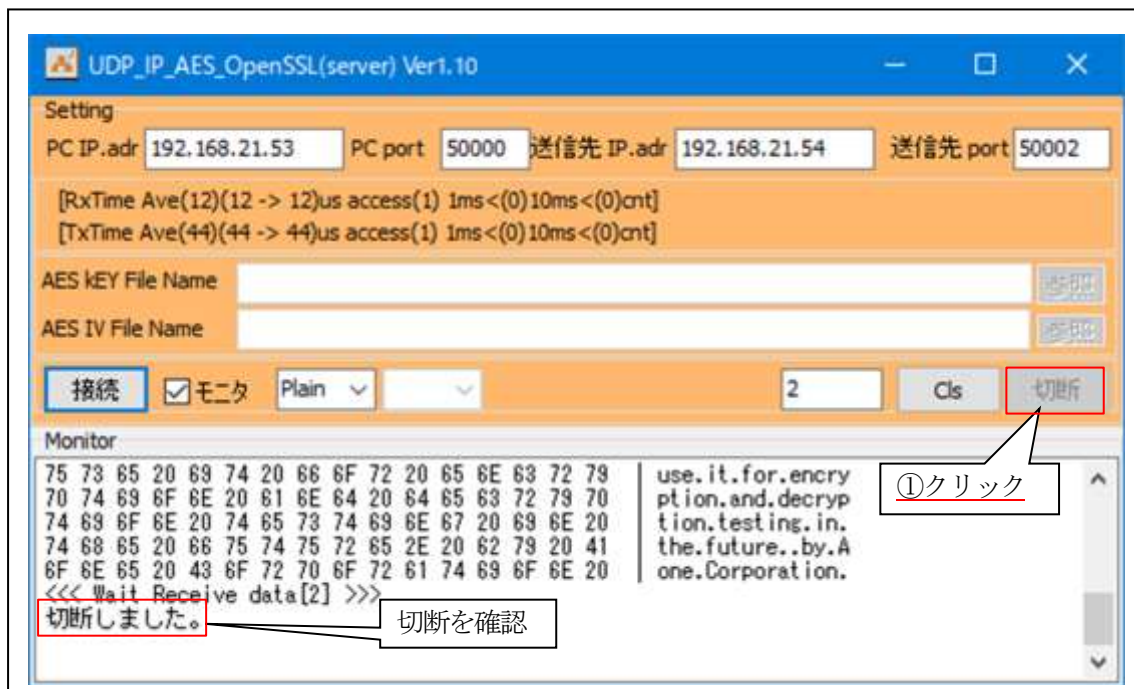
Callout (1) points to the received data section, and callout (2) points to the sent data section.

☆受信データをそのまま送信します。(ループバック)

7) 「UDP_IP_AES_OpenSSL」 その他の操作 (各モード共通)

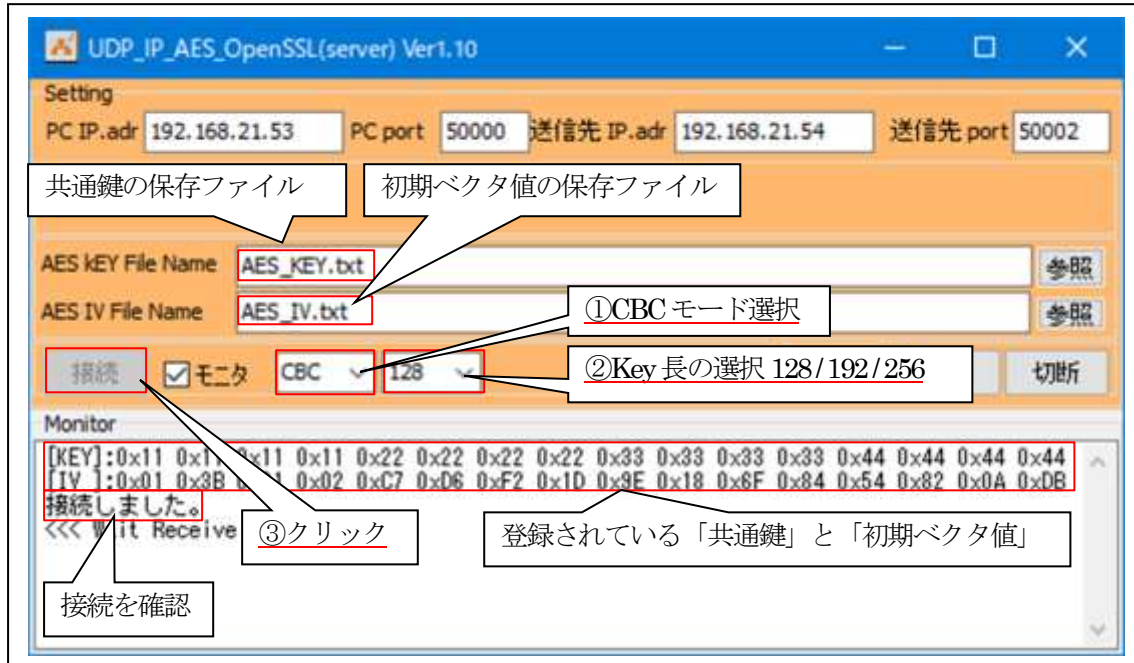


8) UDP_IP-Portを「切断」する。(各モード共通)



5-5. Windows PC 側のテストプログラムで動作確認 (AES-CBC モード)

- 1) 「UDP_IP_AES_OpenSSL.exe」を起動する。
- 2) 「UDP_IP_AES_OpenSSL」の各項目を設定して接続する。



- 3) 「AES_KEY.txt」「AES_IV.txt」の説明

```

「AES_KEY.txt」共通鍵テキストファイル
// default aes_common_key 共通鍵 128bit | 192bit | 256bit
// コメント行は、//のみの使用にしてください。
0x11,0x11,0x11,0x11,0x22,0x22,0x22,0x22,0x33,0x33,0x33,0x33,0x44,0x44,0x44,0x44, // 128bit
0x55,0x55,0x55,0x55,0x66,0x66,0x66,0x66, // ↑ + 192bit
0x77,0x77,0x77,0x77,0x88,0x88,0x88,0x88, // ↑ + 256bit
  
```

☆共通鍵を変更する場合は、基板側と同等の鍵を適当なエディタで変更する。

```

「AES_IV.txt」初期ベクタ値テキストファイル
// default aes_initial_vect 初期化ベクタ
// コメント行は、//のみの使用にしてください。
0x01,0x3B,0x01,0x02,0xC7,0xD6,0xF2,0x1D,0x9E,0x18,0x6F,0x84,0x54,0x82,0x0A,0xDB
  
```

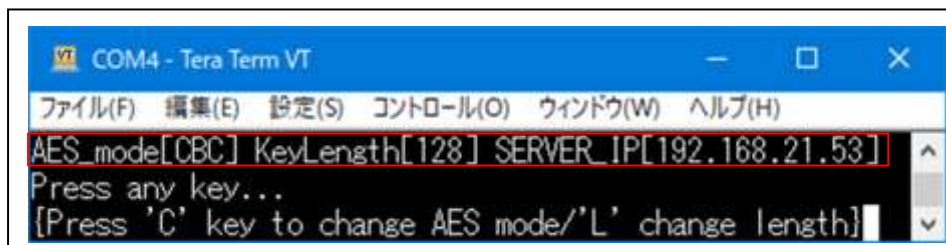
☆初期ベクタ値を変更する場合は、基板側と同等の初期ベクタ値を適当なエディタで変更する。

4) 基板側の各項目の確認と設定。

| 表示項目 | 説明 |
|-----------------------------------|---|
| AES_mode[CBC] | 送受信モードの指定 ◎変数のフラグにより指定 aes.c : int AES_crypto_mode = CBC; // 1=CBC |
| KeyLength[128] | AES-CBC モード時の Key ビット時の指定 ◎変数の数値により指定 aes.c : int AES_crypto_bit = 128; |
| SERVER_IP[192.168.21.53] | 送信先(PC 側)IP アドレス ◎define にて指定 udp_aes_thread_entryc : #define SEVER_IP IP_ADDRESS(192,168,21,53) |

| 共通鍵データの保存モジュールと変数 [aes.c] |
|---|
| <pre>uint8_t AES_key[32] = { // default aes_common_key 共通鍵 128bit 192bit 256bit 0x11,0x11,0x11,0x11,0x22,0x22,0x22,0x22,0x33,0x33,0x33,0x33,0x44,0x44,0x44,0x44, // 128bit 0x55,0x55,0x55,0x55,0x66,0x66,0x66,0x66, // ↑ + 192bit 0x77,0x77,0x77,0x77,0x88,0x88,0x88,0x88, // ↑ + 256bit };</pre> |

| 初期ベクタ値データの保存モジュールと変数 [aes.c] |
|---|
| <pre>uint8_t AES_iv[16] = { // default aes_initial_vect 初期化ベクタ 0x01,0x3B,0x01,0x02,0xC7,0xD6,0xF2,0x1D,0x9E,0x18,0x6F,0x84,0x54,0x82,0x0A,0xDB };</pre> |



5) 基板側から PC(server) 側へ AES-CBC 暗号テキストを送信する。

COM5 - Tera Term VT

ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)

AES_mode[**CBC**] KeyLength[128] ①何らかの Key を Push

Press any key... [Press 'C' key to change AES mode...]

<PlainText length(128) >

54 68 69 73 20 69 73 20 61 20 76 65 72 79 20 69
 6D 70 6F 72 74 61 6E 74 20 73 65 63 72 65 74 20
 6D 65 73 73 61 67 65 21 20 49 20 77 69 6C 6C 20
 75 73 65 20 69 74 20 66 6F 72 20 65 6E 63 72 79
 70 74 69 6F 6E 20 61 6E 64 20 64 65 63 72 79 70
 74 69 6F 6E 20 74 65 73 74 69 6E 67 20 69 6E 20
 74 68 65 20 66 75 74 75 72 65 2E 20 62 79 20 41
 6F 6E 65 20 43 6F 72 70 6F 72 61 74 69 6F 6E 20
 message:..I..with.
 use.it.for.ency
 ption.and.decry
 tion.testing.in.
 the.future..by.A
 one.Corporation.

<Send EncryptText length(128)>

44 50 E8 F0 DA 72 DE 0D 8E D4 8F 1D B3 04 73 C0
 29 30 C8 09 06 DE 48 C2 90 52 08 49 8B 69 63 E1
 64 32 1C 1D C1 8D 80 18 17 45 61 B7 3B B2 25 11
 36 6A 2A CB 36 25 FD 42 07 66 45 31 D2 DF 74 3E
 E4 44 F8 65 49 72 88 B5 30 ED 13 FD 8B A3 06 43
 17 E0 26 1F 9E C5 24 FD 88 18 69 0A 27 89 A7 78
 30 8D 5D B0 0F C9 A1 B2 68 81 40 A8 8C 27 F2 B0
 13 5D 17 A6 FE 18 DA 13 8F 91 7D 06 53 3A 33 BA
 .D.eI.r.和....J.C
 ..&..が\$....i.'7x
 0.]-.ノ.ih.@i.'-
 .]ヲ.レ...}S:30

<Recv EncryptText length(128)>

44 50 E8 F0 DA 72 DE 0D 8E D4 8F 1D B3 04 73 C0
 29 30 C8 09 06 DE 48 C2 90 52 08 49 8B 69 63 E1
 64 32 1C 1D C1 8D 80 18 17 45 61 B7 3B B2 25 11
 36 6A 2A CB 36 25 FD 42 07 66 45 31 D2 DF 74 3E
 E4 44 F8 65 49 72 88 B5 30 ED 13 FD 8B A3 06 43
 17 E0 26 1F 9E C5 24 FD 88 18 69 0A 27 89 A7 78
 30 8D 5D B0 0F C9 A1 B2 68 81 40 A8 8C 27 F2 B0
 13 5D 17 A6 EE 18 DA 13 8F 91 7D 06 53 3A 33 BA
 0.]-.ノ.ih.@i.'-
 .]ヲ.レ...}S:30

<DecryptText length(128) >

54 68 69 73 20 69 73 20 61 20 76 65 72 79 20 69
 6D 70 6F 72 74 61 6E 74 20 73 65 63 72 65 74 20
 6D 65 73 73 61 67 65 21 20 49 20 77 69 6C 6C 20
 75 73 65 20 69 74 20 66 6F 72 20 65 6E 63 72 79
 70 74 69 6F 6E 20 61 6E 64 20 64 65 63 72 79 70
 74 69 6F 6E 20 74 65 73 74 69 6E 67 20 69 6E 20
 74 68 65 20 66 75 74 75 72 65 2E 20 62 79 20 41
 6F 6E 65 20 43 6F 72 70 6F 72 61 74 69 6F 6E 20
 ption.and.decry
 tion.testing.in.
 the.future..by.A
 one.Corporation.

②基板側に保存してある原文のダンプ表示

③原文を暗号化して PC(server)側に送信した暗号文のダンプ表示

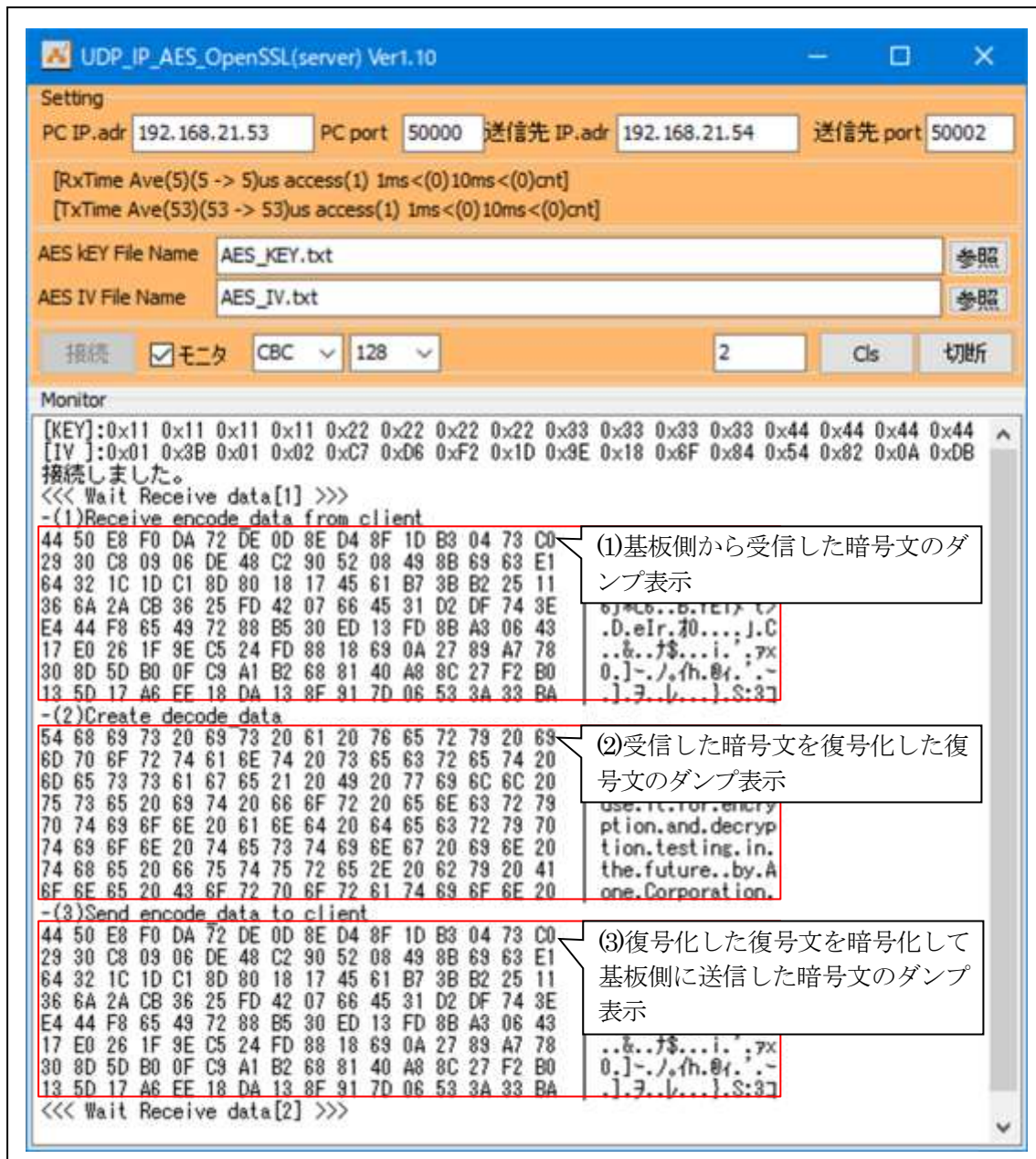
④PC(server)が受信した暗号文を復号した文章を PC (server) 側で暗号化した暗号文を受信したダンプ表示

④受信した PC(server)からの暗号文を復号化した複合文のダンプ表示

☆受信処理が失敗した場合は「**Ctrl+C**」Key-Push で中断する。

- ① 「(1)PlainText」と「(4)DecryptText」が同等の場合、基板側と PC 側が同等な復号処理（デコード）であることの実証になる。
- ② 「(2)Send EncryptText」と「(3)Recv EncryptText」が同等の場合、基板側と PC 側が同等な暗号処理（エンコード）であることの実証になる。

6) 「UDP_IP_AES_OpenSSL」側の送受信を確認する。



- ① 「(1)Receive encode_data from client」と「(3)Send encode_data to client」が同等の場合、基板側とPC側が同等な暗号処理（エンコード）であることの実証になる。
- ② 「(2)Create decode_data」と「基板」側の「(4)DecryptText」が同等の場合、基板側とPC側が同等な復号処理（デコード）であることの実証になる。

6. 注意事項

- ・本文書の著作権は、エーワン（株）が保有します。
- ・本文書を無断での転載は一切禁止します。
- ・本文書に記載されている内容についての質問やサポートはお受けすることが出来ません。
- ・本文章に関して、ルネサス エレクトロニクス社への問い合わせは御遠慮願います。
- ・本文書の内容に従い、使用した結果、損害が発生しても、弊社では一切の責任を負わないものとします。
- ・本文書の内容に関して、万全を期して作成しましたが、ご不審な点、誤りなどの点がありましたら弊社までご連絡くだされば幸いです。
- ・本文書の内容は、予告なしに変更されることがあります。

7. 商標

- ・e2studio・RX65Nは、ルネサス エレクトロニクス株式会社の登録商標または商品名称です。
- ・CK-RX65Nは、ルネサス エレクトロニクス株式会社の商品名です。
- ・その他の会社名、製品名は、各社の登録商標または商標です。

8. 参考文献

- ・「RX65N ユーザーズマニュアル ハードウェア編」 ルネサス エレクトロニクス株式会社
- ・「e2studioユーザーズマニュアル 入門ガイド」 ルネサス エレクトロニクス株式会社
- ・「FreeRTOS」 Amazon.com, Inc.
- ・ルネサス エレクトロニクス株式会社提供のサンプル集
- ・その他

〒486-0852

愛知県春日井市下市場町6-9-20

エーワン株式会社

<https://www.aone.co.jp>

