

Renesas S5D9 用サンプル(e2studio WIRE_DHCP_TCP_AES_COMMAND)の説明

(e2studio Version:2022-10 / SSP Version 2.4.0)

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ください。
- **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。

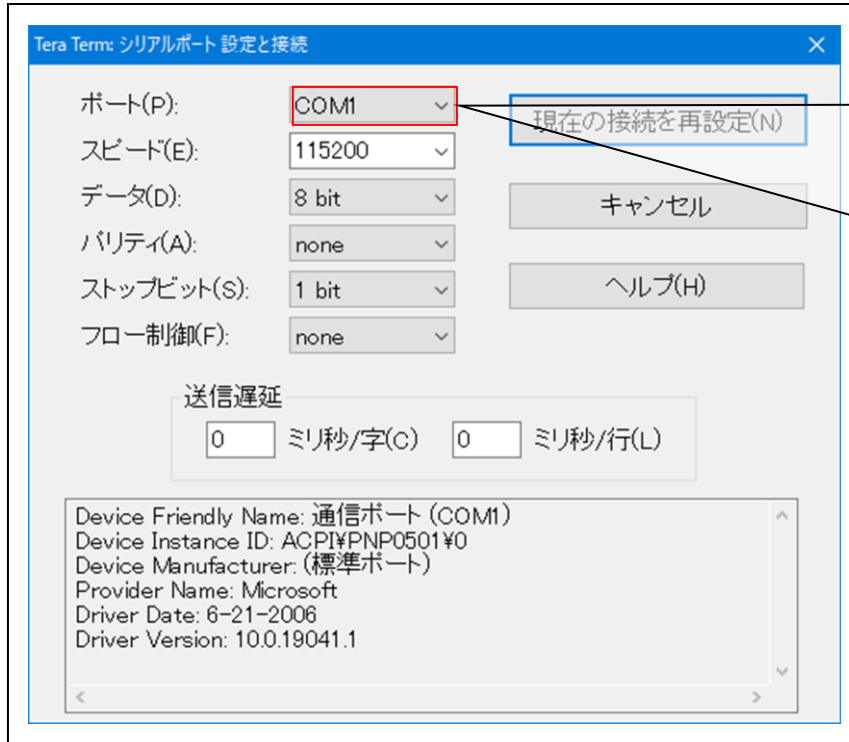
2. サンプルのプロジェクト名

ワークスペース名	概要	プロジェクト名
S5D9_e2std_WIRE_NetX_1	有線 LAN 接続した DHCP と TCP 通信のサンプル セキュリティエンジン (SCE7) Crypto ドライバーの AES を使用したサンプル (暗 号・復号)	WIRE_DHCP_TCP_AES_COMMAND Azure RTOS モードで動作 NetX DHCP Client (g_dhcp_client0) TCP 通信(Client) (nx_tcp_socket_.....) 暗号・復号(AES) (g_sce_aes_r.p_api->encrypt) (g_sce_aes_r.p_api->decrypt)

統合開発環境
Renesas e2studio(Version 2022-10)
SSP(Version2.4.0)

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

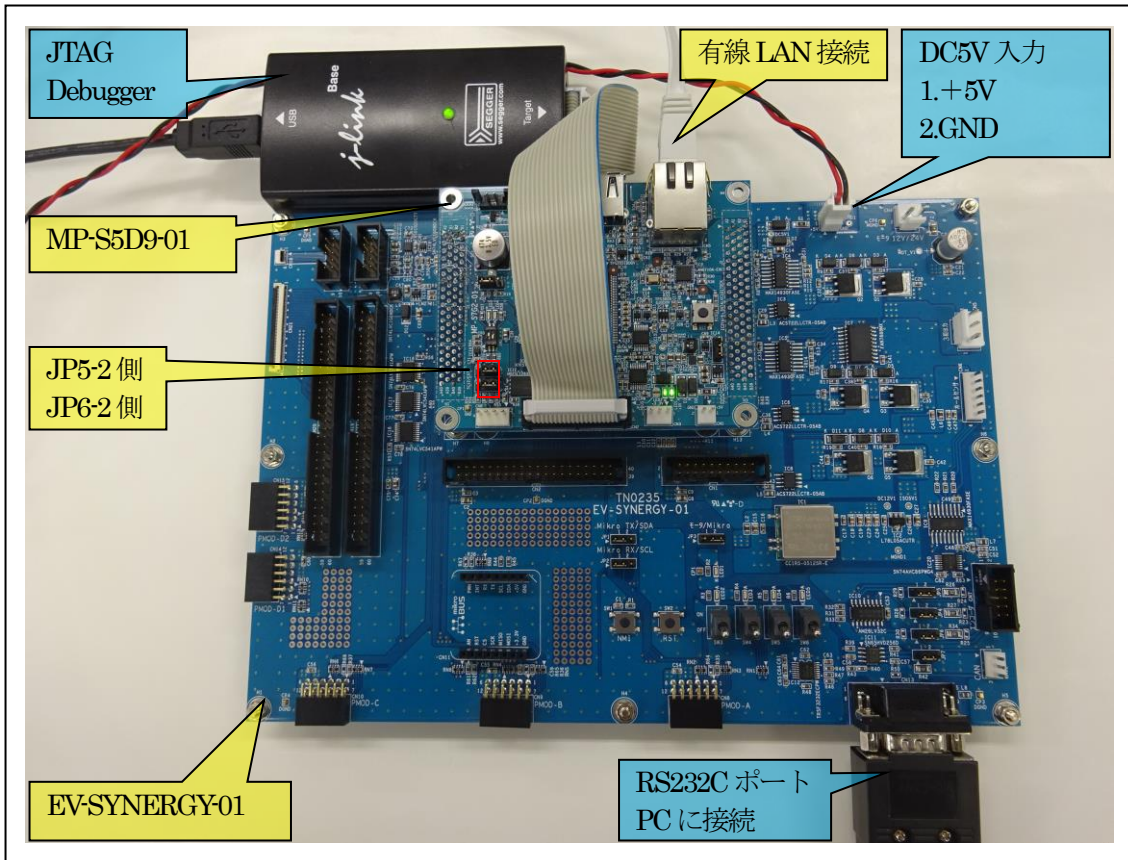


USB シリアルコンバー
タ使用時に CR コ
ードがカットされる
設定の場合は、**受
信 : LF** にして下さ
い。

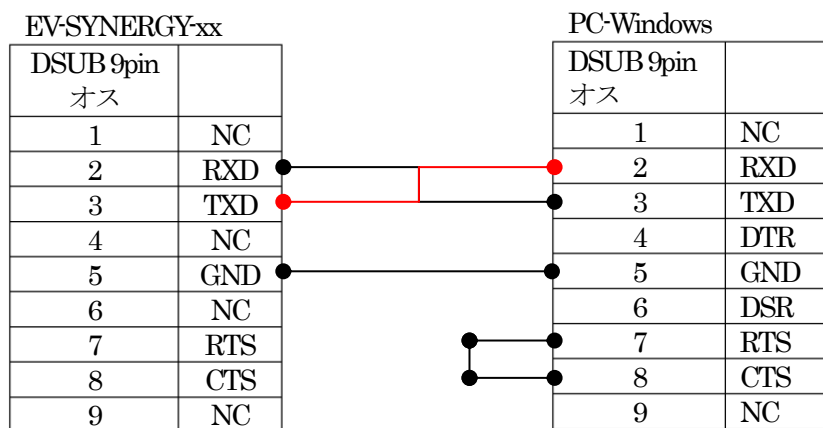
赤枠の設定にする。

4. 動作構成

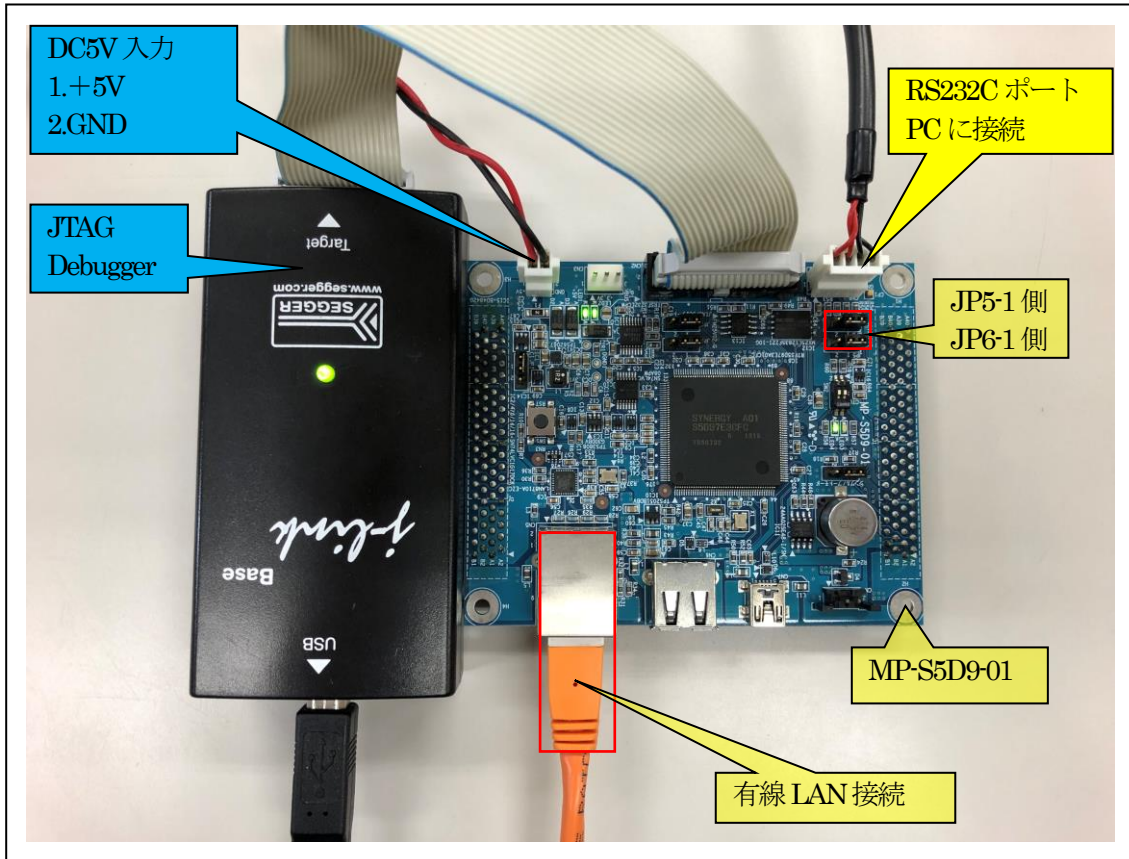
【EV-SYNERGY01】を使用の場合



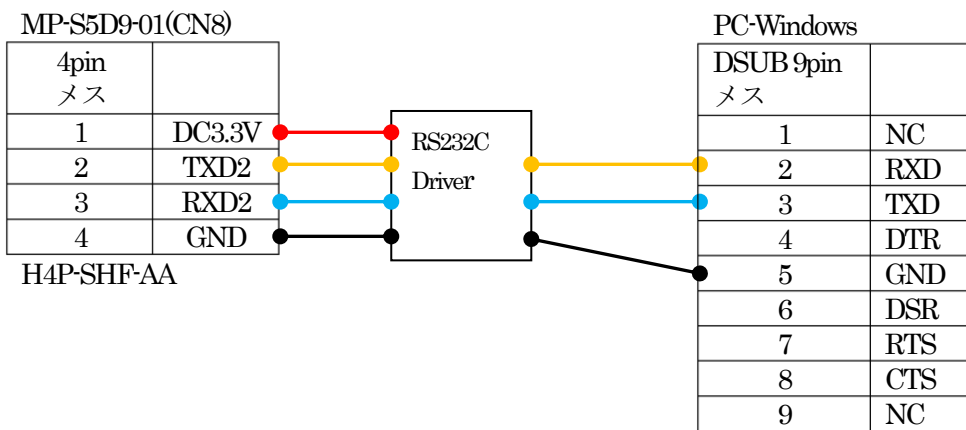
- ①PC機と接続するRS232Cケーブルは、市販「クロスケーブル」でも可能です。
- ②USB-シリアル変換ケーブルを使用される場合は、「StarTech.com社 ICUSB232FIN」推奨
- ③自作する場合は、下記の配線になります。



【MP-S5D9-01】のみ使用の場合



- ①PC機と接続するRS232Cケーブルは、製作が必要です。
- ②「RS232C-Driver」は、下記URLの「RS232CAB4」を推奨します。
http://tool-kobo.ddo.jp/Files/Product/RS232_422/RS232CAB.htm



5. 「S5D9_e2std_WIRE_NetX_1」サンプルの説明

5-1. 「WIRE_DHCP_TCP_AES_COMMAND」フォルダ構成とファイル名

S5D9_e2std_WIRE_NetX_1\WIRE_DHCP_TCP_AES_COMMAND		
Debug	WIRE_DHCP_TCP_AES_COMMAND.elf	ELF ファイル、JTAG で使用
	WIRE_DHCP_TCP_AES_COMMAND.map	MAP ファイル、アドレス情報管理
	WIRE_DHCP_TCP_AES_COMMAND.srec	モトローラーHEX ファイル
	その他	自動生成ファイル
Script	S5D9.ld	ロケーション定義ファイル
Src	Command.c	コマンド入力処理のサンプルファイル
	Command.h	Command.c 用ヘッダーファイル
	tcp_cmd_thread_entry.c	TCP Cmd Thread サンプルファイル
	wire_dhcp_thred_entry.c	DHCP client Thread サンプルファイル
sce7_aes	sce7_aes.c	AES の暗号・復号化サンプルファイル
	sce7_aes.h	sce7_aes.c 用ヘッダーファイル
MP-S5D9-01 (リンク指定)	e2p.c	E2PROM 処理モジュール
	e2p.h	e2p.c 用ヘッダーファイル
	led.c	LED 処理モジュール
	led.h	led.c 用ヘッダーファイル
	sci2.c	シリアル通信処理モジュール
	sci2.h	sci2.c 用ヘッダーファイル
	stchar.c	文字系処理モジュール
	stcahr.h	stchar.c 用ヘッダーファイル
synergy_gen	Generate を行うと作成されるファイル	
Synergy	Generate を行うと作成されるファイル	
synergy_cfg		
Configuration.xml	プロジェクト Generation ファイル	
PIN-MP-S5D9-01.pincfg	PIN configuration 用ファイル	
WIRE_DHCP_TCP_AES_COMMAND.jlink	Jlink デバッガー用ファイル	
その他	自動生成ファイル	

5-2. サンプルの動作説明

<WIRE DHCP Thread>

1) DHCPによるIPアドレスの取得を待つ

Term 画面

< 1 > 「<tcp_thread waiting to get IP address>」

< 2 > 「<interface status check>」

MP 基板に実装した EEPROM より MAC アドレス読みドライバにセットする。

< 3 > 「<Start WIRE NetX DHCP>」

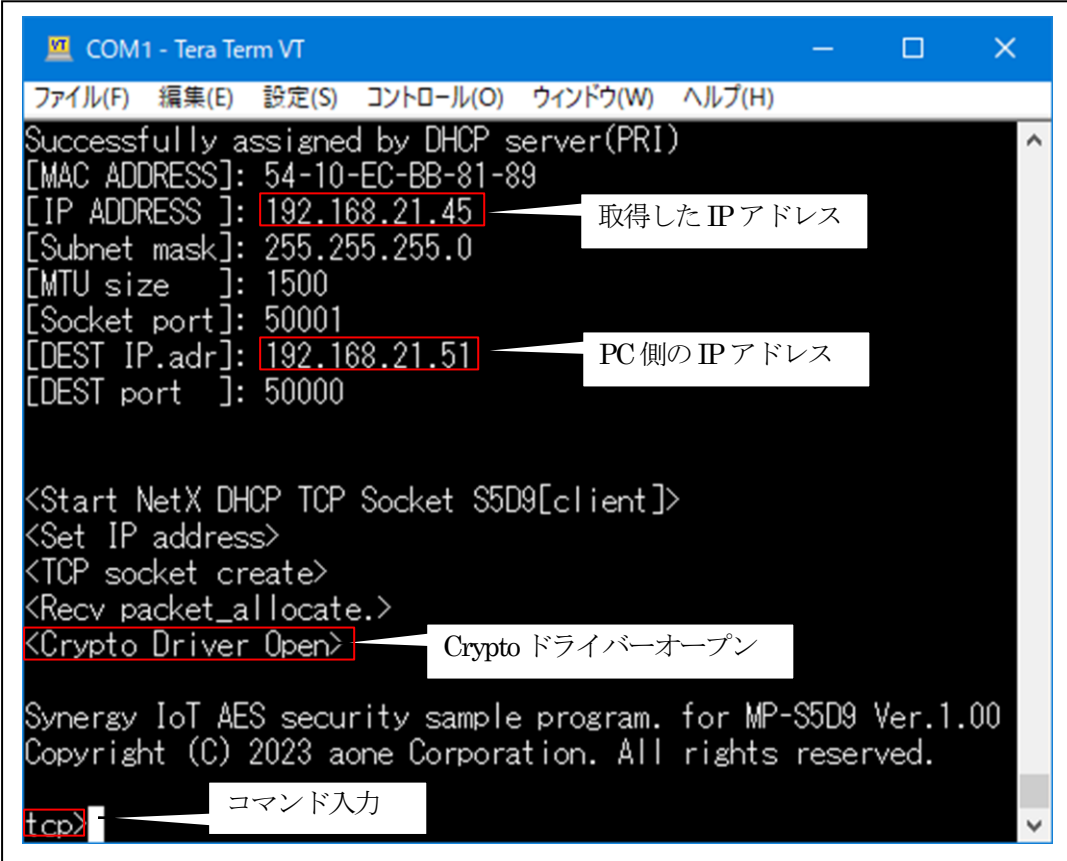
< 4 > 「<Wait DHCP State Change>」

< 5 > 「<Start WIRE NetX DHCP>」

< 6 > 「<Wait DHCP BOUND>」の順次処理して表示する。

・IP アドレス取得成功により、MP 基板上の LED 4 を led blink thread で 200msec 毎に点滅

2) IP ドレス取得情報を Term 画面に表示する。



```

COM1 - 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 server(PRI)
[MAC ADDRESS]: 54-10-EC-BB-81-89
[IP ADDRESS ]: 192.168.21.45
[Subnet mask]: 255.255.255.0
[MTU size   ]: 1500
[Socket port]: 50001
[DEST IP.adr]: 192.168.21.51
[DEST port  ]: 50000

<Start NetX DHCP TCP Socket S5D9[client]>
<Set IP address>
<TCP socket create>
<Recv packet_allocate.>
<Crypto Driver Open>

Synergy IoT AES security sample program. for MP-S5D9 Ver.1.00
Copyright (C) 2023 aone Corporation. All rights reserved.

tcp>
  
```

取得した IP アドレス

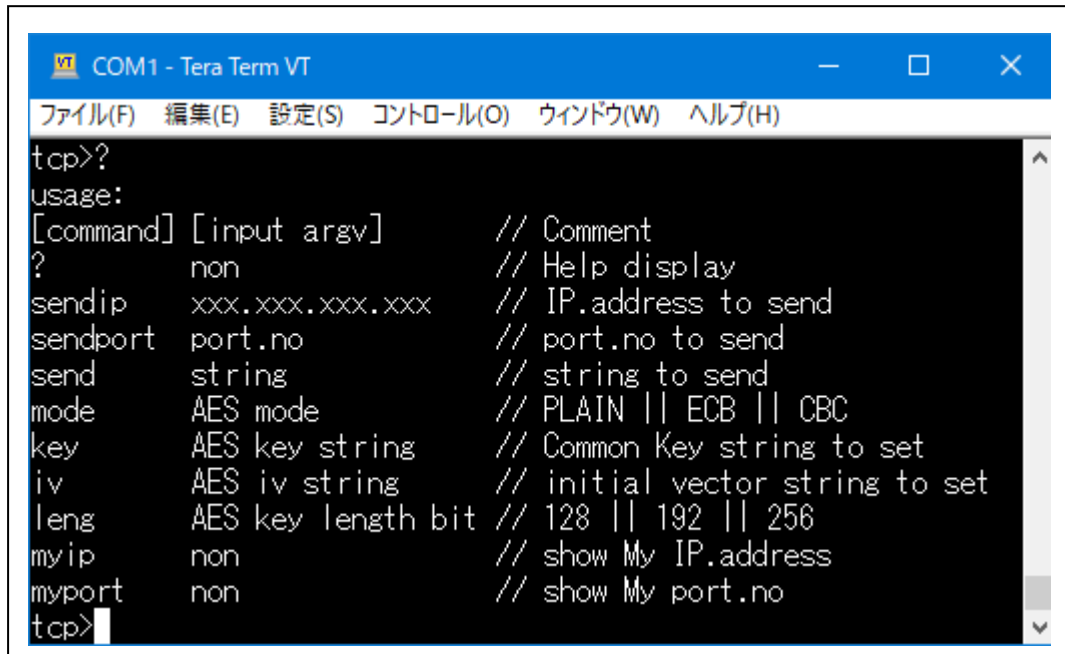
PC 側の IP アドレス

Crypto ドライバーオープン

コマンド入力

<TCP Cmd Thread>

- 3) オープニングメッセージを表示してコマンド入力を待つ。
 - ・TCPThread 起動後、MP 基板上の LED4 を 100msec 毎に点滅
- 4) コマンドの説明
 - (1) ? (ヘルプコマンド) コマンド一覧の表示



```

COM1 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウインドウ(W) ヘルプ(H)
tcp>?
usage:
[command] [input argv] // Comment
? non // Help display
sendip xxx.xxx.xxx.xxx // IP.address to send
sendport port.no // port.no to send
send string // string to send
mode AES mode // PLAIN || ECB || CBC
key AES key string // Common Key string to set
iv AES iv string // initial vector string to set
leng AES key length bit // 128 || 192 || 256
myip non // show My IP.address
myport non // show My port.no
tcp>
  
```

- (2) `sendip` (送信先 IP アドレスの登録と参照)

<機能>

- ・送信先 (PC 機側) の IP アドレスを登録する。

<例>

```

xxx> sendip 192.168.21.9↵
xxx> sendip↵
[send IP]: 192.168.21.51
  
```

- (3) `sendport` (送信先ポート番号の登録と参照)

<機能>

- ・送信先 (PC 機側) のポート番号を登録する。

<例>

```

xxx> sendport 50000↵
xxx> sendport↵
[send port]: 50000
  
```

- (4) `send` (文字列の送信)

<機能>

- ・送信先 (PC 機側) に文字列を送信する。

<例>

```

xxx> send 12345678123456781234567812345678↵
  
```

(5) mode (暗号モードの設定と参照)

<機能>

- ・送信先(PC機側)に文字列を送信するための暗号モードを指定する。

- ①<PLAIN>平文で送信
- ②<ECB>文字列を「AES-ECBモード」で暗号化して送信する。
- ③<CBC>文字列を「AES-CBCモード」で暗号化して送信する。
- ④<CTR>文字列を「AES-CTRモード」で暗号化して送信する。
- ⑤<GCM>文字列を「AES-GCMモード」で暗号化して送信する。
- ⑥<XTS>文字列を「AES-XTSモード」で暗号化して送信する。

<例>

```
xxx> mode PLAIN↵
xxx> mode ECB↵
xxx> mode CBC↵
xxx> mode CTR↵
xxx> mode GCM↵
xxx> mode XTS↵
xxx> mode↵
[Now mode]: ECB
```

(6) key (共通鍵の設定と参照)

<機能>

- ・暗号・復号化するための共通鍵を設定する。(16進数設定)

<AES-XTS以外>

```
//key長 128bit
xxx> key 0x11111111 0x22222222 0x33333333 0x44444444↵
//key長 196bit
xxx> key 0x11111111 0x22222222 0x33333333 0x44444444 0x55555555 0x66666666 ↵
//key長 256bit
xxx> key 0x11111111 0x22222222 0x33333333 0x44444444 0x55555555 0x66666666 0x77777777 0x88888888↵
xxx> key↵
[Now Key]: 11111111 22222222 33333333 44444444
```

<AES-XTSはKey長が倍になる>

```
//key長 128bit
xxx> key 0x11111111 0x22222222 0x33333333 0x44444444 0x55555555 0x66666666 0x77777777 0x88888888↵
//key長 256bit
xxx> key 0x11111111 0x22222222 0x33333333 0x44444444 0x55555555 0x66666666 0x77777777 0x88888888
0x99999999 0xaaaaaaaa 0xbbbbbbbb 0xcccccccc 0xdddddddd 0xeeeeeeee 0xffffffff 0x11111111↵
xxx> key↵
[Now Key]: 0x11111111 0x22222222 0x33333333 0x44444444
:0x55555555 0x66666666 0x77777777 0x88888888
:0x99999999 0xAAAAAAAA 0BBBBBBBB 0CCCCCCC
:0DDDDDDDD 0EEEEEEEE 0FFFFFFF 0x11111111
```


(7) i v (初期ベクタ値の設定と参照)

<機能>

- ・暗号・復号化 (AES-CBC) モード時の初期ベクタ値を設定する。(16進数設定)

<AES-GCM 以外>

```
xxx>iv 0x02013B01 0x1DF2D6C7 0x846F189E 0xDB0A8254<
```

```
xxx>iv<[Now IV]: 0x02013B01 0x1DF2D6C7 0x846F189E 0xDB0A8254
```

<AES-GCM は、128bit に Format された 96bitIV にする必要あり>

```
xxx>iv 0x02013B01 0x1DF2D6C7 0x846F189E 0x01000000<
```

```
xxx>iv<[Now IV]: 0x02013B01 0x1DF2D6C7 0x846F189E 0x01000000
```

96bitIV: 0x02013B01 0x1DF2D6C7 0x846F189E

128bit に Format された 96bitIV: バイト列で表記

0x01,0x3B,0x01,0x02,0xC7,0xD6,0xF2,0x1D,0x9E,0x18,0x6F,0x84,0x00,0x00,0x00,0x01

AES-GCM の IV の 96bit 以降は、必ず、バイト表記で 0x00,0x00,0x00,0x01 にする必要があります。

(8) l e n g (共通鍵長の設定と参照)

<機能>

- ・共通鍵の長さ (Key length 128/192/256) を設定する。(10進数設定)

<例>

```
xxx>leng 128<
```

```
xxx>leng 192<
```

```
xxx>leng 256<
```

```
xxx>leng<
```

```
[Now Key Length]: 128
```

(9) m y i p (DHCP で取得した自 I P アドレスの参照)

<機能>

- ・DHCP で取得した自 I P アドレスを表示する。

<例>

```
xxx>myip<
```

```
[My IP]: 192.168.21.9
```

(10) m y p o r t (基板側で登録したポート番号の参照)

<機能>

- ・基板側で登録したポート番号を表示する。

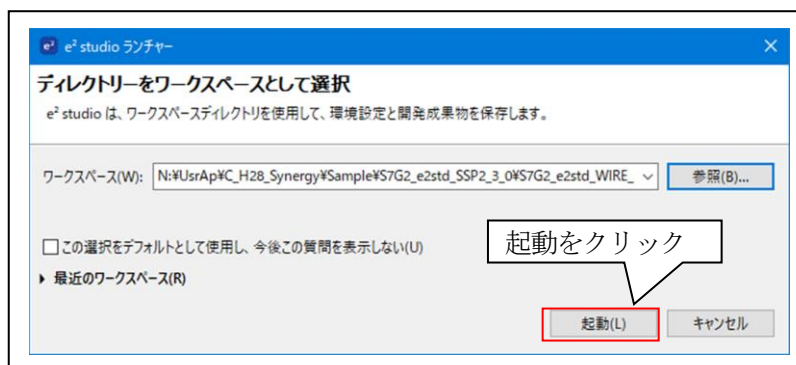
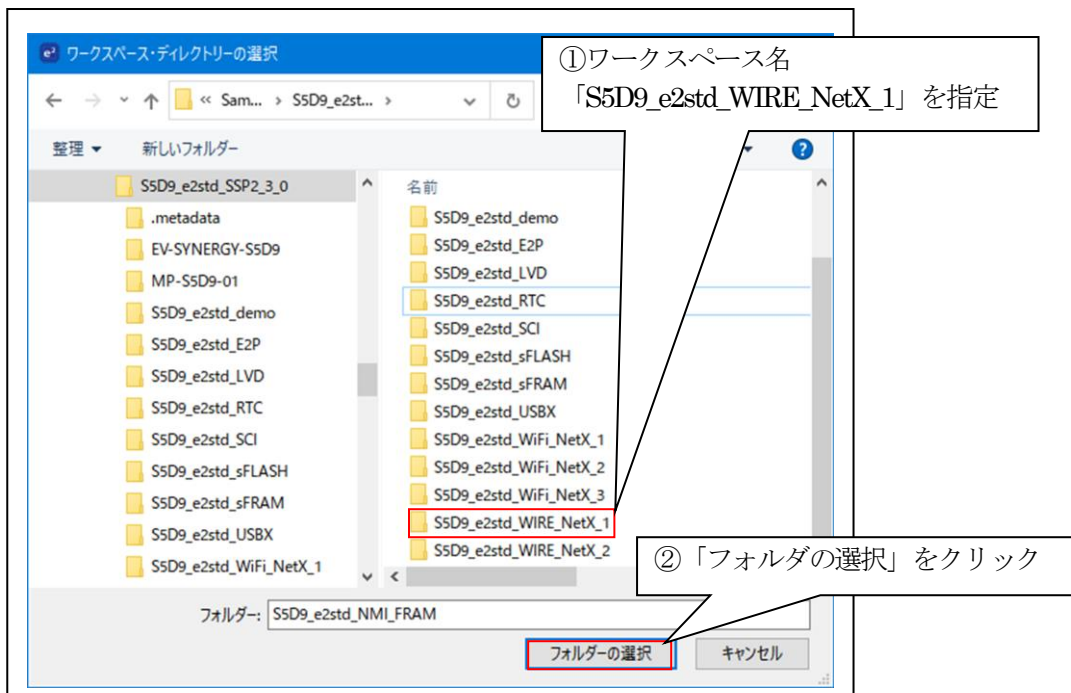
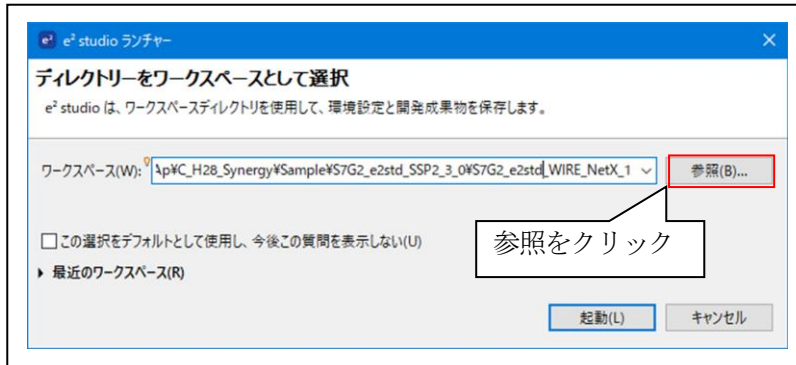
<例>

```
xxx>myport<
```

```
[My port]: 50001
```

6. 「S5D9_e2std_WIRE_NetX_1」のワークスペースを指定する。

6-1. ワークスペース名の指定



6-2. プロジェクトのインポート

☆詳細操作は「[e2studio_synergy_Import.pdf](#)」の2項を参照して下さい。

7. デバッグ操作

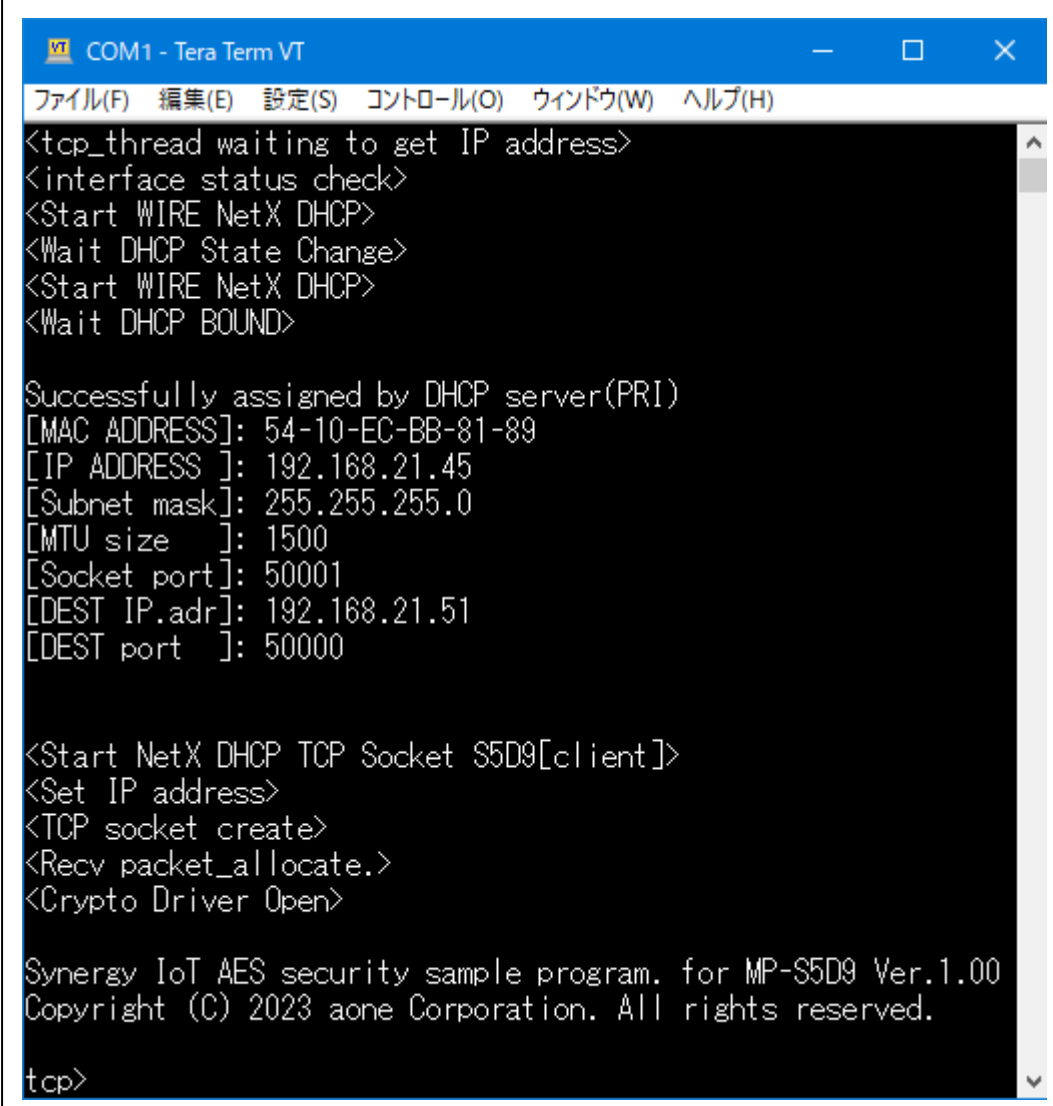
7-1. デバッグ構成の設定

☆詳細操作は「[e2studio_synergy_Import.pdf](#)」の3-1項を参照して下さい。

7-2. デバッグの開始

☆詳細操作は「[e2studio_synergy_Import.pdf](#)」の3-2項を参照して下さい。

<WIRE_DHCP_TCP 実行画面>



```

COM1 - 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)
<tcp_thread waiting to get IP address>
<interface status check>
<Start WIRE NetX DHCP>
<Wait DHCP State Change>
<Start WIRE NetX DHCP>
<Wait DHCP BOUND>

Successfully assigned by DHCP server(PRI)
[MAC ADDRESS]: 54-10-EC-BB-81-89
[IP ADDRESS ]: 192.168.21.45
[Subnet mask]: 255.255.255.0
[MTU size   ]: 1500
[Socket port]: 50001
[DEST IP.adr]: 192.168.21.51
[DEST port  ]: 50000

<Start NetX DHCP TCP Socket S5D9[client]>
<Set IP address>
<TCP socket create>
<Recv packet_allocate.>
<Crypto Driver Open>

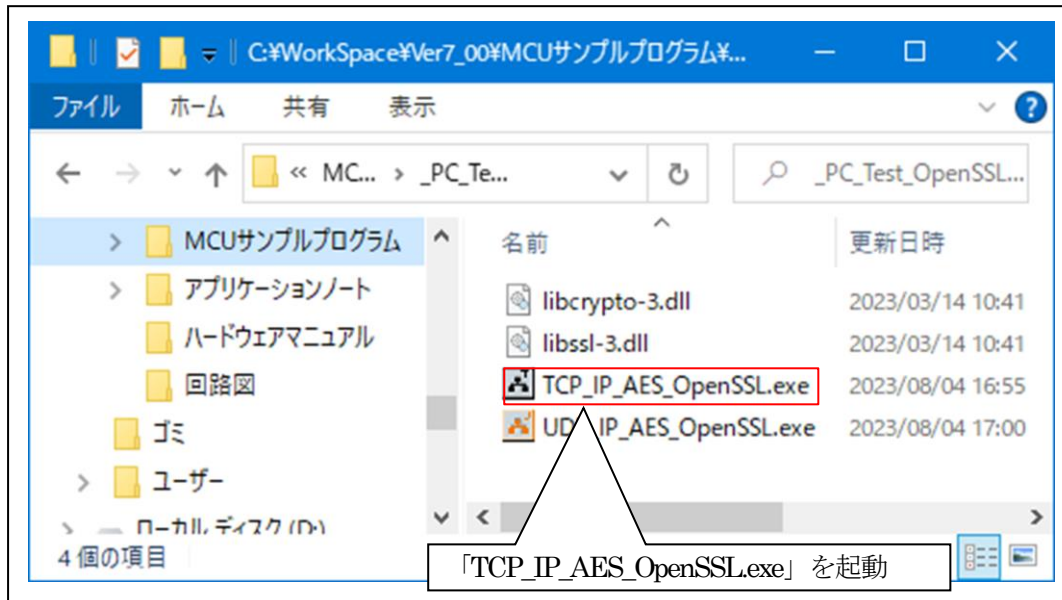
Synergy IoT AES security sample program. for MP-S5D9 Ver.1.00
Copyright (C) 2023 aone Corporation. All rights reserved.

tcp>
  
```

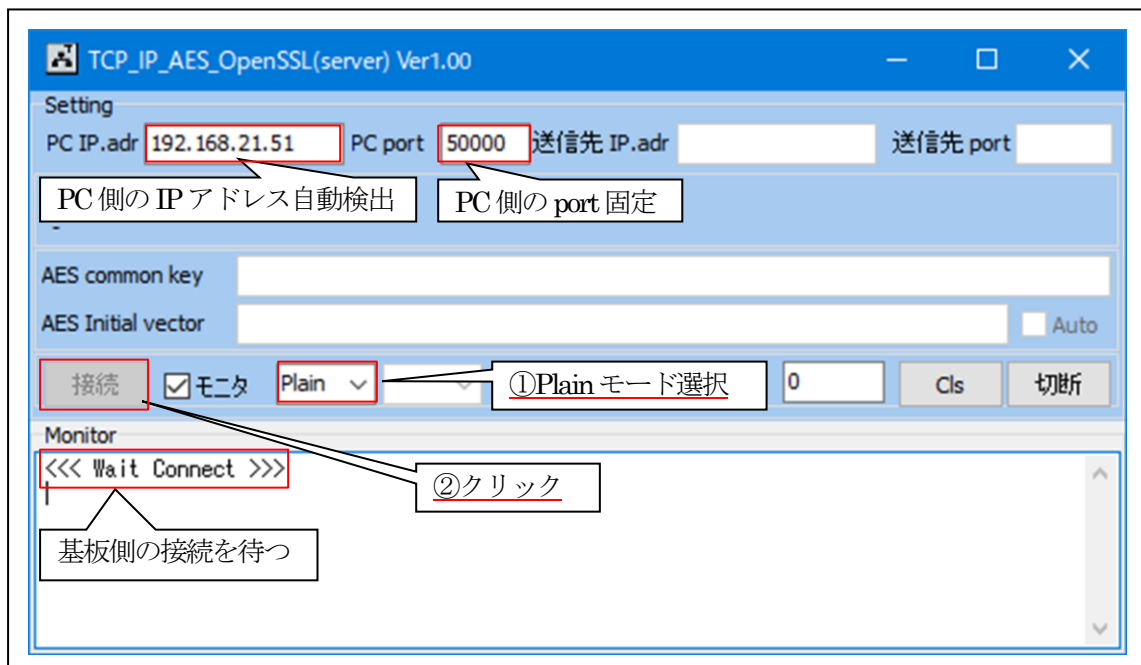
7-3. Windows PC 側のテスト用プログラムで動作確認 (PLAIN (平文) モード)

- 1) 「TCP_IP_AES_OpenSSL」を起動する。

プログラム場所【ご購入 CD¥MCU サンプルプログラム¥_PC_Test_OpenSSL】を Read/Write 可能な適当なフォルダに Copy してからお使い下さい。



- 2) 「TCP_IP_AES_OpenSSL」の各項目を設定して「基板」側からの「接続」を待つ。



3) 「基板」側の各項目の確認と設定。

①PC側のIPアドレスを設定する。

ex) sendip 192.168.21.51<Enter>

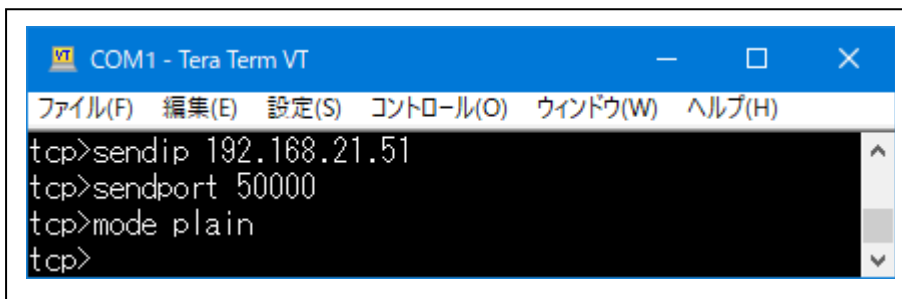
②PC側のポート番号を設定する。

ex) sendport 50000<Enter>

③「plain」(平文)モードを指定する。

ex) mode plain<Enter>

④実行画面



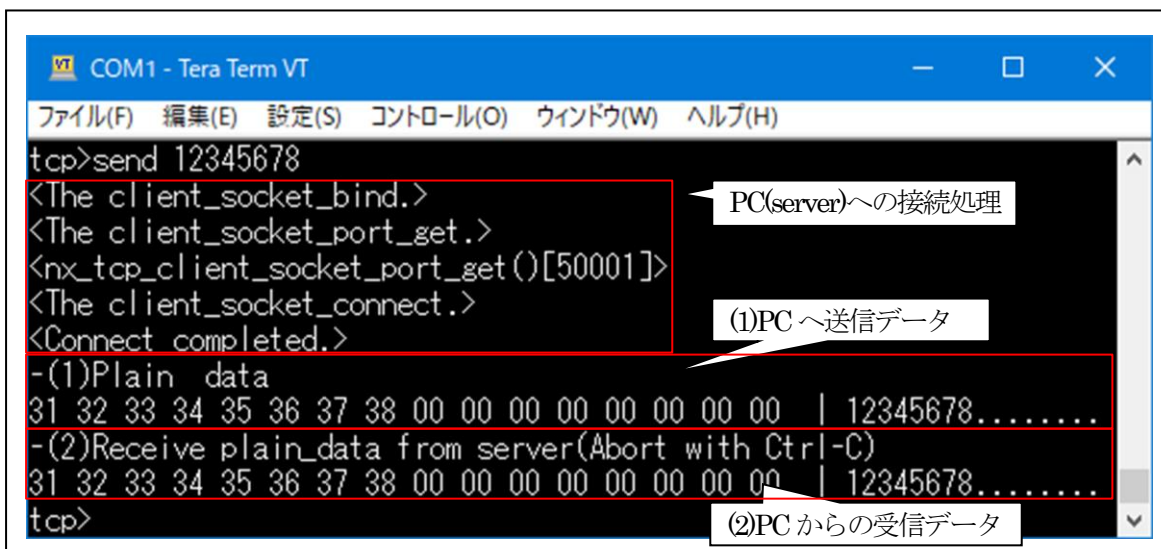
```

COM1 - 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)
tcp>sendip 192.168.21.51
tcp>sendport 50000
tcp>mode plain
tcp>
  
```

4) 「基板」側から「PC」(server)側へ平文テキストを送信する。

①テキスト文を送信する。(未接続の場合は、接続を実行)

ex) send 12345678



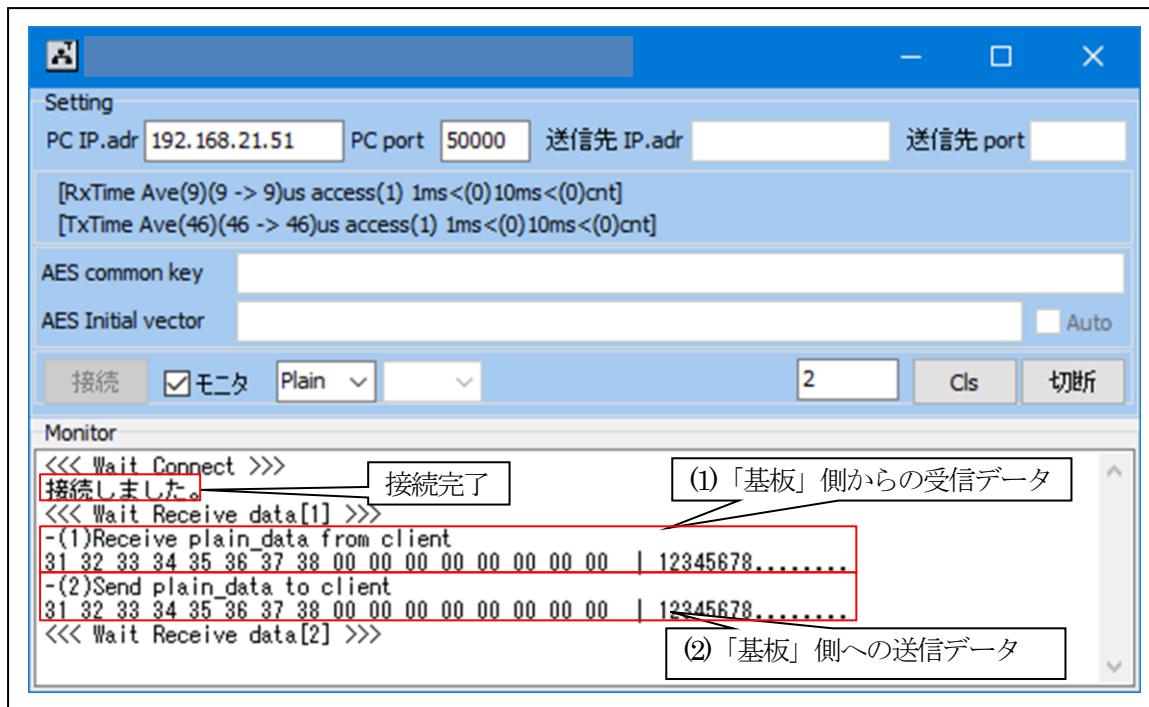
```

COM1 - 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)
tcp>send 12345678
<The client_socket_bind.>
<The client_socket_port_get.>
<nx_tcp_client_socket_port_get()[50001]>
<The client_socket_connect.>
<Connect completed.>
-(1)Plain data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
-(2)Receive plain_data from server(Abort with Ctrl-C)
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
tcp>
  
```

Annotations in the image:

- PC(server)への接続処理 (points to the connection sequence)
- (1)PCへ送信データ (points to the outgoing data line)
- (2)PCからの受信データ (points to the incoming data line)

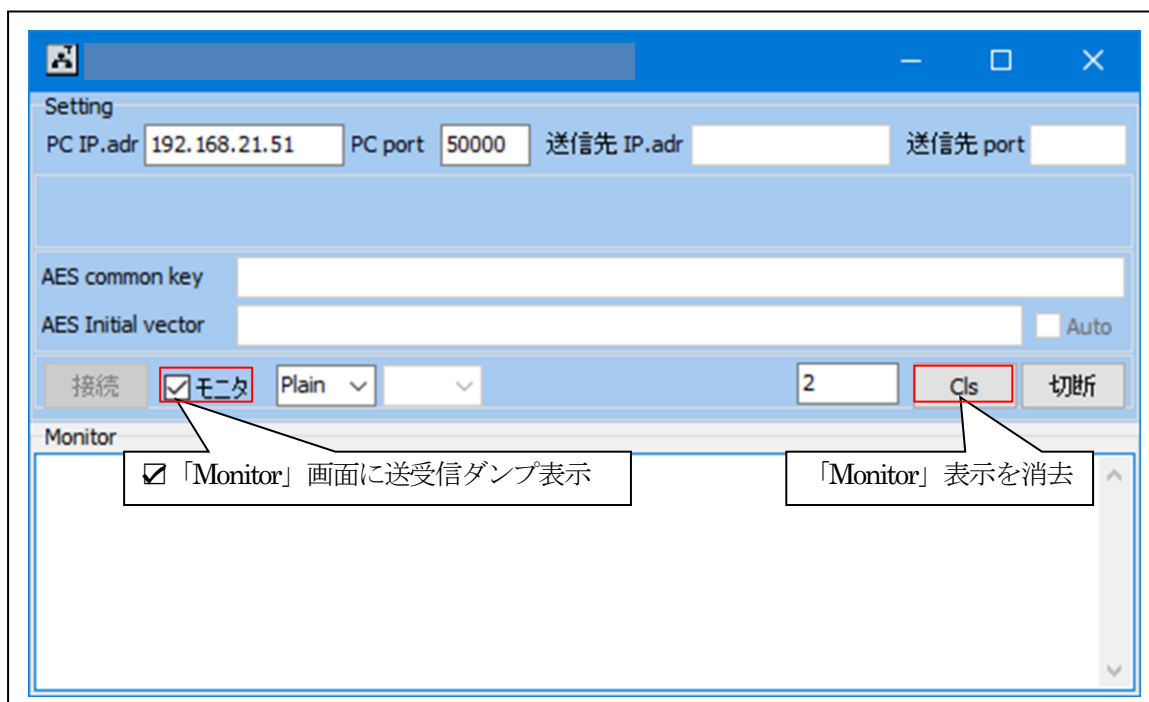
5) 「TCP_IP_AES_OpenSSL」側の送受信を確認する。



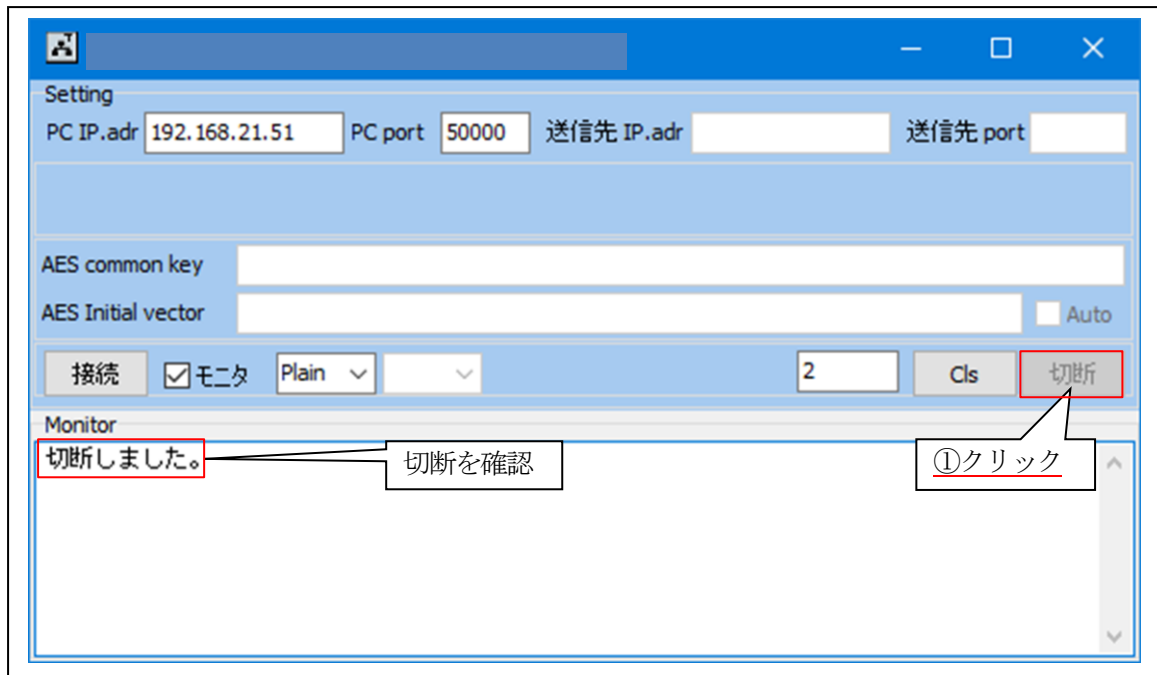
【Error 表示】

- ・受信時の接続失敗 「"error!! client_socket_connect()[error code]"」
- ・受信異常 「"error!! recvfrom()[error code]"」
- ・送信異常 「"error!! sendto()[error code]"」

6) 「TCP_IP_AES_OpenSSL」その他の操作（各モード共通）



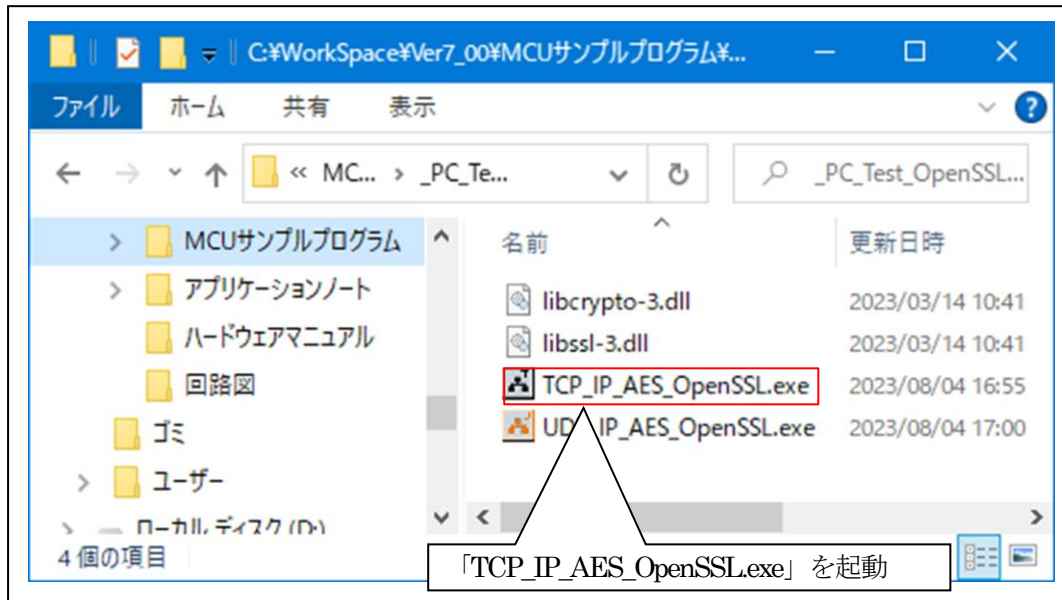
7) TCP_IP-Portを「切断」する。(各モード共通)



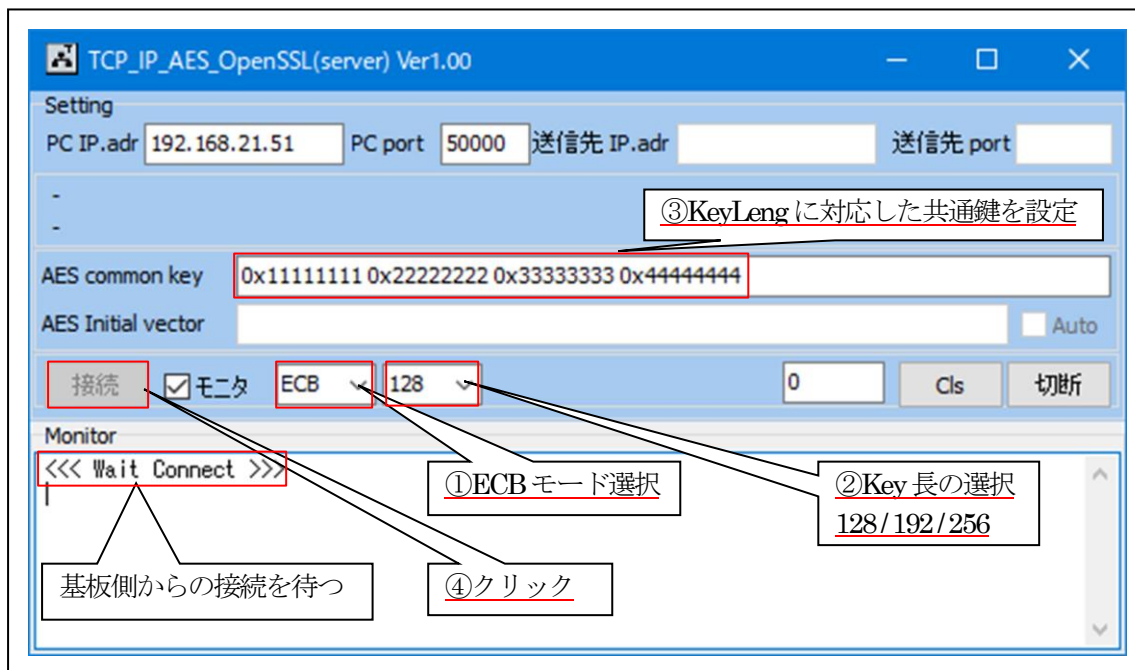
7-4. Windows PC 側のテスト用プログラムで動作確認 (AES-ECB モード)

- 1) 「TCP_IP_AES_OpenSSL」を起動する。

プログラム場所【ご購入 CD¥MCU サンプルプログラム¥_PC_Test_OpenSSL】を Read/Write 可能な適当なフォルダに Copy してからお使い下さい。



- 2) 「TCP_IP_AES_OpenSSL」の各項目を設定して「基板」側からの「接続」を待つ。



3) 「基板」側の各項目の確認と設定。

①PC側のIPアドレスを設定する。

ex) sendip 192.168.21.51<↵

②PC側のポート番号を設定する。

ex) sendport 50000<↵

③「ECB」(AES-ECB)モードを指定する。

ex) mode ecb<↵

④「Key Leng」を指定する。(128 / 192 / 256)

PC側の「TCP_IP_AES_OpenSSL」と同じKey長にする。

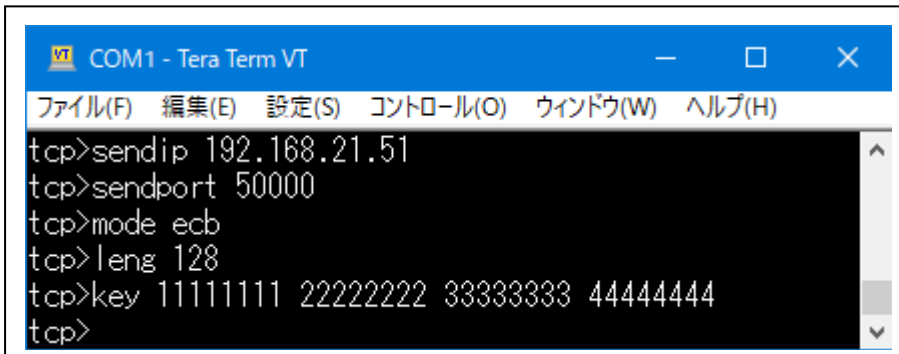
ex) leng 128<↵

⑤「共通鍵」を指定する。(HEX 数値)

PC側の「TCP_IP_AES_OpenSSL」と同等の共通鍵(AES common key)にする。

ex) key 11111111 22222222 33333333 44444444<↵

⑥実行画面



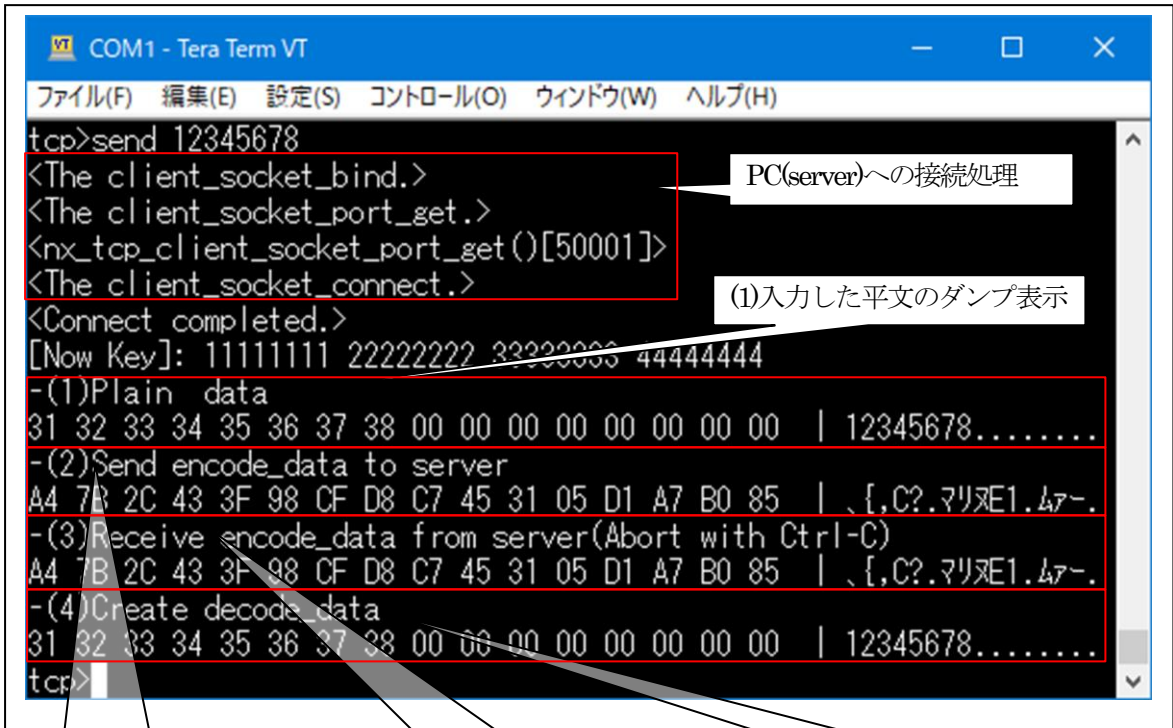
```

COM1 - 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)
tcp>sendip 192.168.21.51
tcp>sendport 50000
tcp>mode ecb
tcp>leng 128
tcp>key 11111111 22222222 33333333 44444444
tcp>
  
```

4) 「基板」側から「PC」(server)側へ ECB 暗号テキストを送信する。

①テキスト文を送信する。(未接続の場合は、接続を実行)

ex) send 12345678



```

COM1 - 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)
tcp>send 12345678
<The client_socket_bind.>
<The client_socket_port_get.>
<nx_tcp_client_socket_port_get()[50001]>
<The client_socket_connect.>
<Connect completed.>
[Now Key]: 11111111 22222222 33333333 44444444
-(1)Plain data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
-(2)Send encode_data to server
A4 7B 2C 43 3F 98 CF D8 C7 45 31 05 D1 A7 B0 85 | \[,C?.マリアE1.ムア-.
-(3)Receive encode_data from server(Abort with Ctrl-C)
A4 7B 2C 43 3F 98 CF D8 C7 45 31 05 D1 A7 B0 85 | \[,C?.マリアE1.ムア-.
-(4)Create decode_data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
tcp>
  
```

PC(server)への接続処理

(1)入力した平文のダンプ表示

(2)入力した平文を暗号化 (エンコード) して PC (Server) に送信したデータのダンプ表示

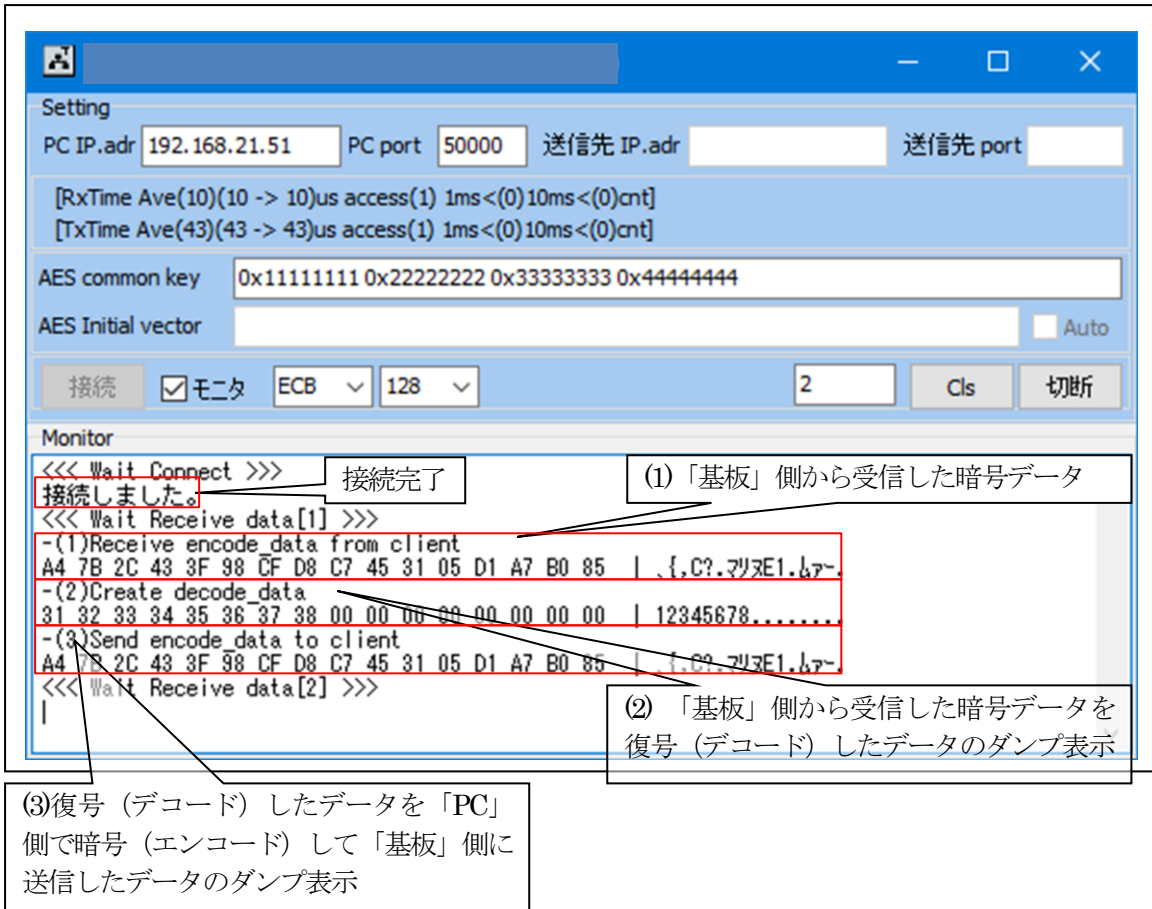
(3)PC (server) が受信した暗号文を復号 (デコード) した文章を PC (server) で再度暗号化 (エンコード) したデータを受信したダンプ表示

(4)受信した PC からの暗号文を復号 (デコード) したデータのダンプ表示

② 「(1)Plain data」と「(4)Create decode_data」が同等のダンプ表示の場合、基板側と PC 側が同等な復号処理 (デコード) であることの証明になります。

③ 「(2)Send encode_data to server」と「(3)Receive encode_data from server(Abort with Ctrl-C)」が同等のダンプ表示の場合、基板側と PC 側が同等な暗号処理 (エンコード) であることの証明になります。

5) 「TCP_IP_AES_OpenSSL」側の送受信を確認する。



The screenshot shows the application's settings and a monitor window. The settings include PC IP address (192.168.21.51), PC port (50000), and an AES common key. The monitor window displays the following log entries:

```

  <<< Wait Connect >>>
  接続しました。
  <<< Wait Receive data[1] >>>
  -(1)Receive encode_data from client
  A4 7B 2C 43 3F 98 CF D8 C7 45 31 05 D1 A7 B0 85 | ,C?.?/?E1.6ア-
  -(2)Create decode_data
  31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 | 12345678.....
  -(3)Send encode_data to client
  A4 7B 2C 43 3F 98 CF D8 C7 45 31 05 D1 A7 B0 85 | ,C?.?/?E1.6ア-
  <<< Wait Receive data[2] >>>
  |
  
```

Annotations in the image explain the logs:

- (1) 「基板」側から受信した暗号データ
- (2) 「基板」側から受信した暗号データを復号 (デコード) したデータのダンプ表示
- (3) 復号 (デコード) したデータを「PC」側で暗号 (エンコード) して「基板」側に送信したデータのダンプ表示

- ① 「(1)Receive encode_data from client」と「(3)Send encode_data to client」が同等のダンプ表示の場合、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ります。
- ② 「(2)Create decode_data」と「基板」側 TeraTerm 画面の「(4)Create decode_data」が同等のダンプ表示の場合、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ります。

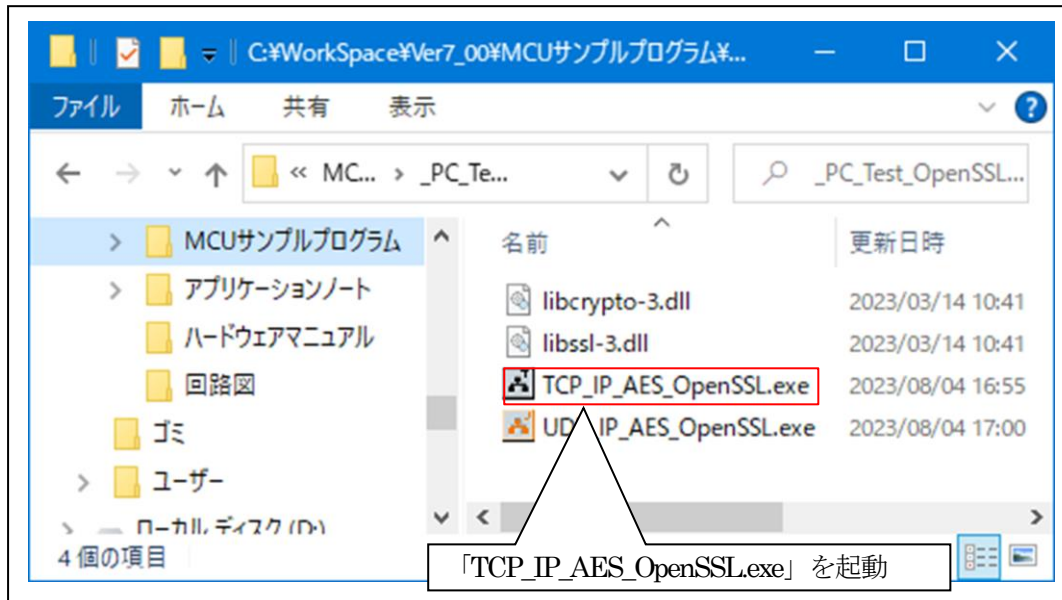
【Error 表示】

- ・受信時の接続失敗 「"error!! client_socket_connect()[error code]"
- ・受信異常 「"error!! rcvfrom()[error code]"
- ・送信異常 「"error!! sendto()[error code]"

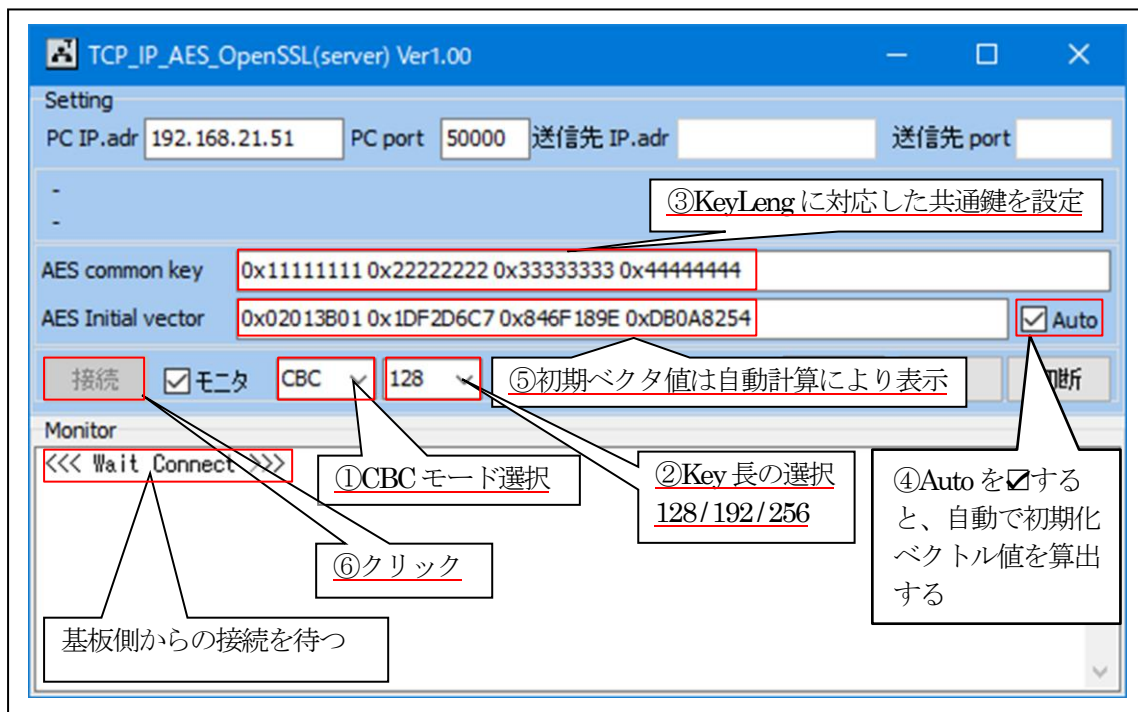
7-5. Windows PC 側のテスト用プログラムで動作確認 (AES-CBC モード)

1) 「TCP_IP_AES_OpenSSL」を起動する。

プログラム場所【ご購入 CD¥MCU サンプルプログラム¥_PC_Test_OpenSSL】を Read/Write 可能な適当なフォルダに Copy してからお使い下さい。



2) 「TCP_IP_AES_OpenSSL」の各項目を設定して「基板」側からの「接続」を待つ。



3) 「基板」側の各項目の確認と設定。

①PC側のIPアドレスを設定する。

ex) sendip 192.168.21.51<↵

②PC側のポート番号を設定する。

ex) sendport 50000<↵

③「CBC」(AES-CBC)モードを指定する。

ex) mode cbc<↵

④「Key Leng」を指定する。(128 / 192 / 256)

PC側の「TCP_IP_AES_OpenSSL」と同じKey長にする。

ex) leng 128<↵

⑤「共通鍵」を指定する。(HEX 数値)

PC側の「TCP_IP_AES_OpenSSL」と同等の共通鍵(AES common key)にする。

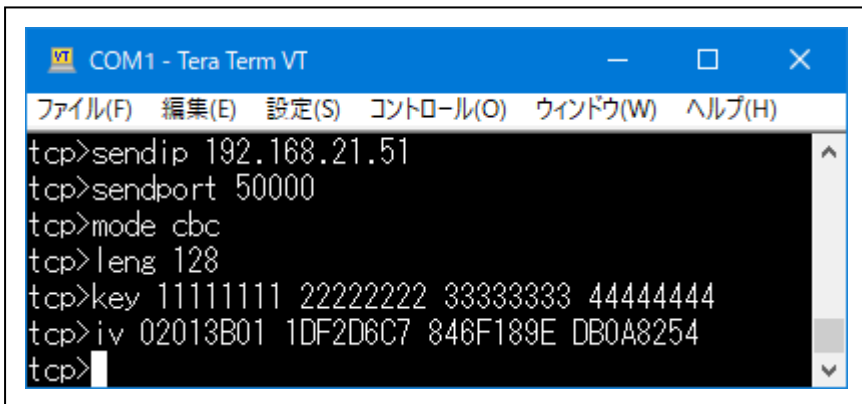
ex) key 11111111 22222222 33333333 44444444<↵

⑥「初期ベクタ値」を指定する。(HEX 数値)

PC側の「TCP_IP_AES_OpenSSL」と同等の初期ベクタ値(AES Initial vector)にする。

ex) iv 02013B01 1DF2D6C7 846F189E DB0A8254<↵

⑦実行画面



```

COM1 - 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)
tcp>sendip 192.168.21.51
tcp>sendport 50000
tcp>mode cbc
tcp>leng 128
tcp>key 11111111 22222222 33333333 44444444
tcp>iv 02013B01 1DF2D6C7 846F189E DB0A8254
tcp>
  
```

4) 「基板」側から「PC」(server)側へ CBC 暗号テキストを送信する。

①テキスト文を送信する。(未接続の場合は、接続を実行)

ex) send 12345678

COM1 - 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)

```
tcp>send 12345678
<The client_socket_bind.>
<The client_socket_port_get.>
<nx_tcp_client_socket_port_get()[50001]>
<The client_socket_connect.>
<Connect completed.>
[Now Key]: 11111111 22222222 33333333 44444444
[Now IV ]: 02013B01 1DF2D6C7 846F189E DB0A8254
-(1)Plain data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
-(2)Send encode_data to server
40 0B 96 58 87 17 01 92 3D 9F 40 BB 73 3F 7A D3 | @..X....=@#s?zF
-(3)Receive encode_data from server(Abort with Ctrl-C)
40 0B 96 58 87 17 01 92 3D 9F 40 BB 73 3F 7A D3 | @..X....=@#s?zF
-(4)Create decode_data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
tcp>
```

PC(server)への接続処理

(1)入力した平文のダンプ表示

(2)入力した平文を暗号化 (エンコード) して PC (Server) に送信したデータのダンプ表示

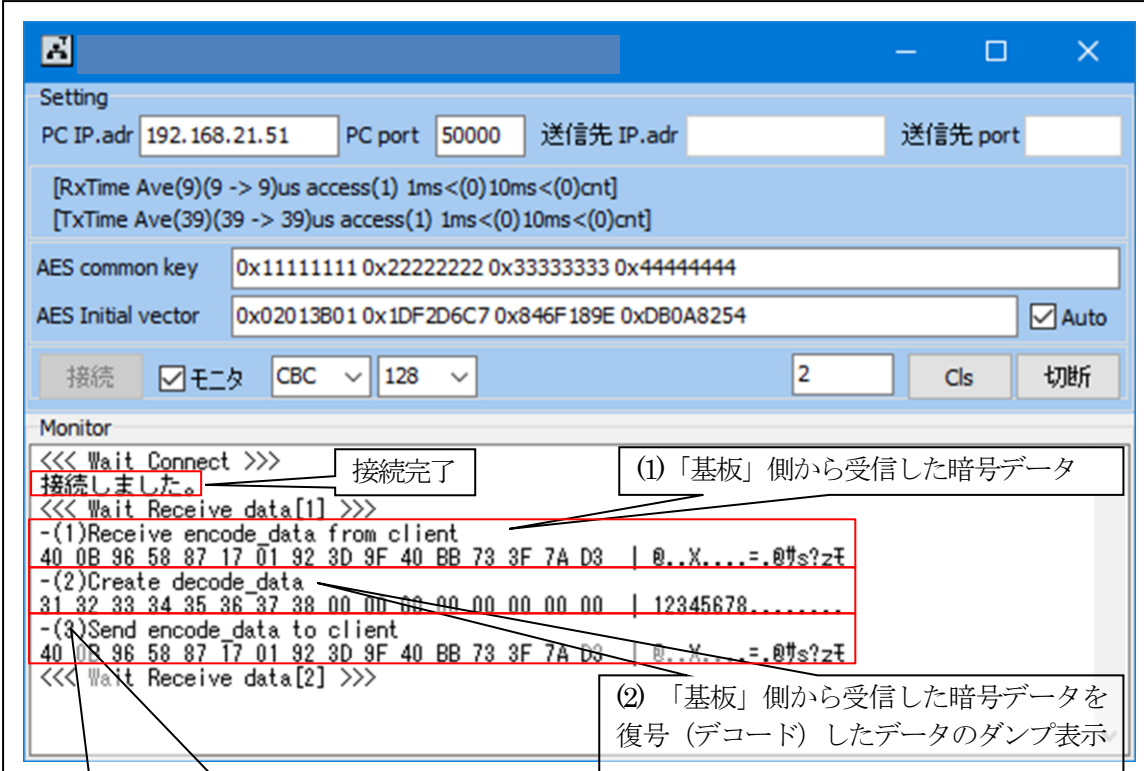
(3)PC (server) が受信した暗号文を復号 (デコード) した文章を PC (server) で再度暗号化 (エンコード) したデータを受信したダンプ表示

(4)受信した PC からの暗号文を復号 (デコード) したデータのダンプ表示

② 「(1)Plain data」と「(4)Create decode_data」が同等のダンプ表示の場合、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ります。

③ 「(2)Send encode_data to server」と「(3)Receive encode_data from server(Abort with Ctrl-C)」が同等のダンプ表示の場合、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ります。

5) 「TCP_IP_AES_OpenSSL」側の送受信を確認する。



The screenshot shows the 'Setting' window with the following configuration:

- PC IP.adr: 192.168.21.51
- PC port: 50000
- 送信先 IP.adr: [Empty]
- 送信先 port: [Empty]
- [RxTime Ave(9)(9 -> 9)us access(1) 1ms<(0)10ms<(0)cnt]
- [TxTime Ave(39)(39 -> 39)us access(1) 1ms<(0)10ms<(0)cnt]
- AES common key: 0x11111111 0x22222222 0x33333333 0x44444444
- AES Initial vector: 0x02013B01 0x1DF2D6C7 0x846F189E 0xDB0A8254 Auto
- 接続: モニタ CBC 128 2 Cls 切断

The 'Monitor' window displays the following log output with annotations:

```

  <<< Wait Connect >>>
  接続しました。
  接続完了
  (1) 「基板」側から受信した暗号データ
  <<< Wait Receive data[1] >>>
  -(1)Receive encode_data from client
  40 0B 96 58 87 17 01 92 3D 9F 40 BB 73 3F 7A D3 | @..X.....=.0#s?z#
  -(2)Create decode_data
  31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 | 12345678.....
  -(3)Send encode_data to client
  40 0B 96 58 87 17 01 92 3D 9F 40 BB 73 3F 7A D3 | @..X.....=.0#s?z#
  <<< Wait Receive data[2] >>>
  (2) 「基板」側から受信した暗号データを復号 (デコード) したデータのダンプ表示
  (3)復号 (デコード) したデータを「PC」側で暗号 (エンコード) して「基板」側に送信したデータのダンプ表示
  
```

- ① 「(1)Receive encode_data from client」と「(3)Send encode_data to client」が同等のダンプ表示の場合、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ります。
- ② 「(2)Create decode_data」と「基板」側 TeraTerm 画面の「(4)Create decode_data」が同等のダンプ表示の場合、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ります。

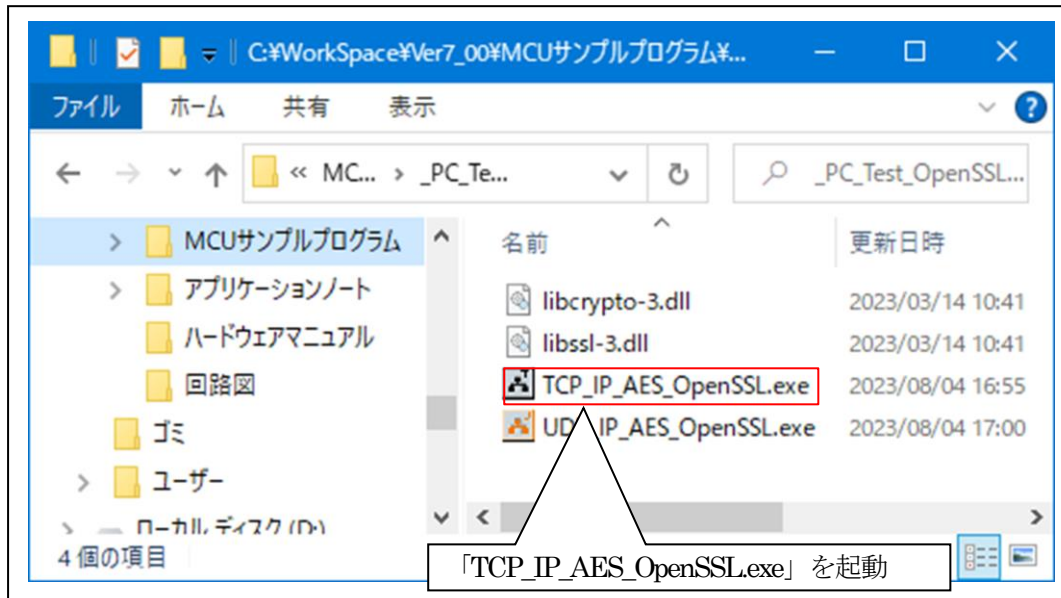
【Error 表示】

- ・受信時の接続失敗 「"error!! client_socket_connect()[error code]"
- ・受信異常 「"error!! rcvfrom()[error code]"
- ・送信異常 「"error!! sendto()[error code]"

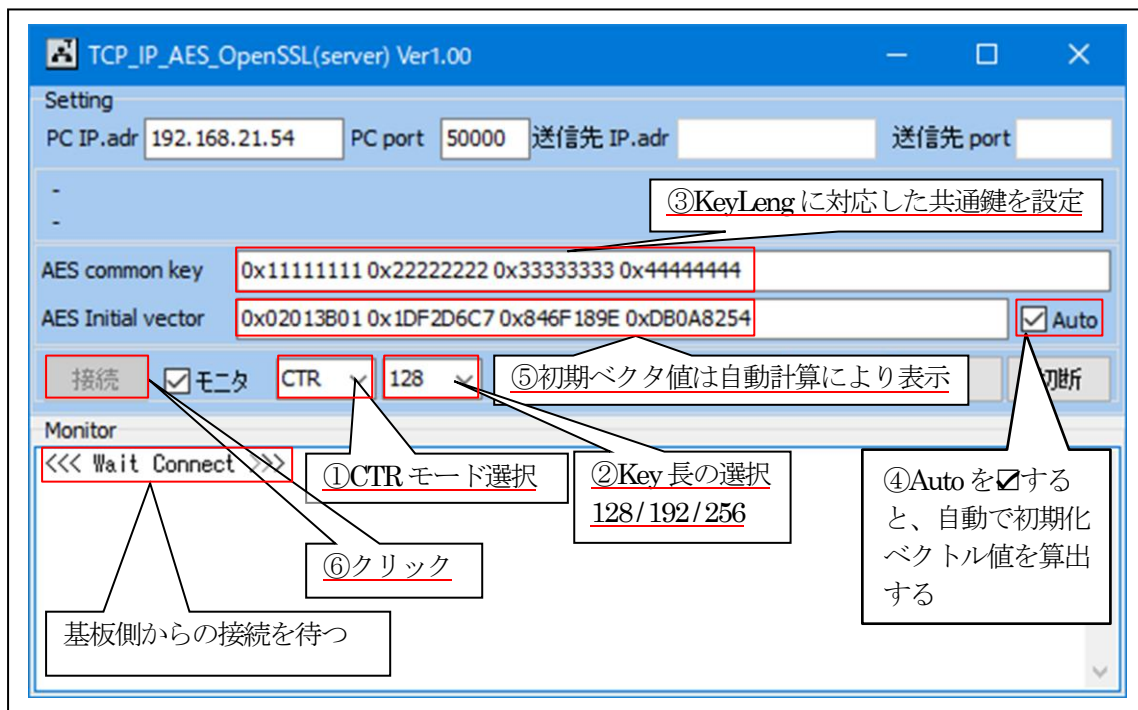
7-6. Windows PC 側のテスト用プログラムで動作確認 (AES-CTR モード)

- 1) 「TCP_IP_AES_OpenSSL」を起動する。

プログラム場所【ご購入 CD¥MCU サンプルプログラム¥_PC_Test_OpenSSL】を Read/Write 可能な適当なフォルダに Copy してからお使い下さい。



- 2) 「TCP_IP_AES_OpenSSL」の各項目を設定して「基板」側からの「接続」を待つ。



3) 「基板」側の各項目の確認と設定。

①PC側のIPアドレスを設定する。

ex) sendip 192.168.21.54<␣>

②PC側のポート番号を設定する。

ex) sendport 50000<␣>

③「CTR」(AES-CTR) モードを指定する。

ex) mode ctr<␣>

④「Key Leng」を指定する。(128 / 192 / 256)

PC側の「TCP_IP_AES_OpenSSL」と同じKey長にする。

ex) leng 128<␣>

⑤「共通鍵」を指定する。(HEX 数値)

PC側の「TCP_IP_AES_OpenSSL」と同等の共通鍵(AES common key)にする。

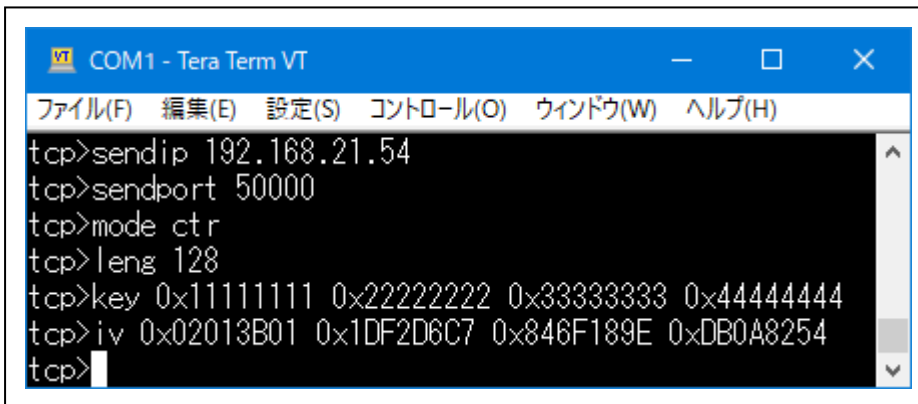
ex) key 11111111 22222222 33333333 44444444<␣>

⑥「初期ベクタ値」を指定する。(HEX 数値)

PC側の「TCP_IP_AES_OpenSSL」と同等の初期ベクタ値(AES Initial vector)にする。

ex) iv 02013B01 1DF2D6C7 846F189E DB0A8254<␣>

⑦実行画面



```

COM1 - 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)
tcp>sendip 192.168.21.54
tcp>sendport 50000
tcp>mode ctr
tcp>leng 128
tcp>key 0x11111111 0x22222222 0x33333333 0x44444444
tcp>iv 0x02013B01 0x1DF2D6C7 0x846F189E 0xDB0A8254
tcp>
  
```

4) 「基板」側から「PC」(server)側へCTR暗号テキストを送信する。

①テキスト文を送信する。(未接続の場合は、接続を実行)

ex) send 12345678

The screenshot shows a terminal window titled 'COM1 - Tera Term VT'. The user enters the command 'tcp>send 12345678'. The terminal output shows the following sequence of events:

- Connection establishment: '<The client_socket_bind.>', '<The client_socket_port_get.>', '<nx_tcp_client_socket_port_get()[50001]>', '<The client_socket_connect.>', '<Connect completed.>'.
- Key and IV display: '[Now Key]: 11111111 22222222 33333333 44444444', '[Now IV]: 02013B01 1DF2D6C7 846F189E DB0A8254'.
- Step (1): '-(1)Plain data' followed by a hex dump of the plaintext '12345678'.
- Step (2): '-(2)Send encode_data to server' followed by a hex dump of the encoded data.
- Step (3): '-(3)Receive encode_data from server(Abort with Ctrl-C)' followed by a hex dump of the received encoded data.
- Step (4): '-(4)Create decode data' followed by a hex dump of the decoded data, which matches the original plaintext '12345678'.

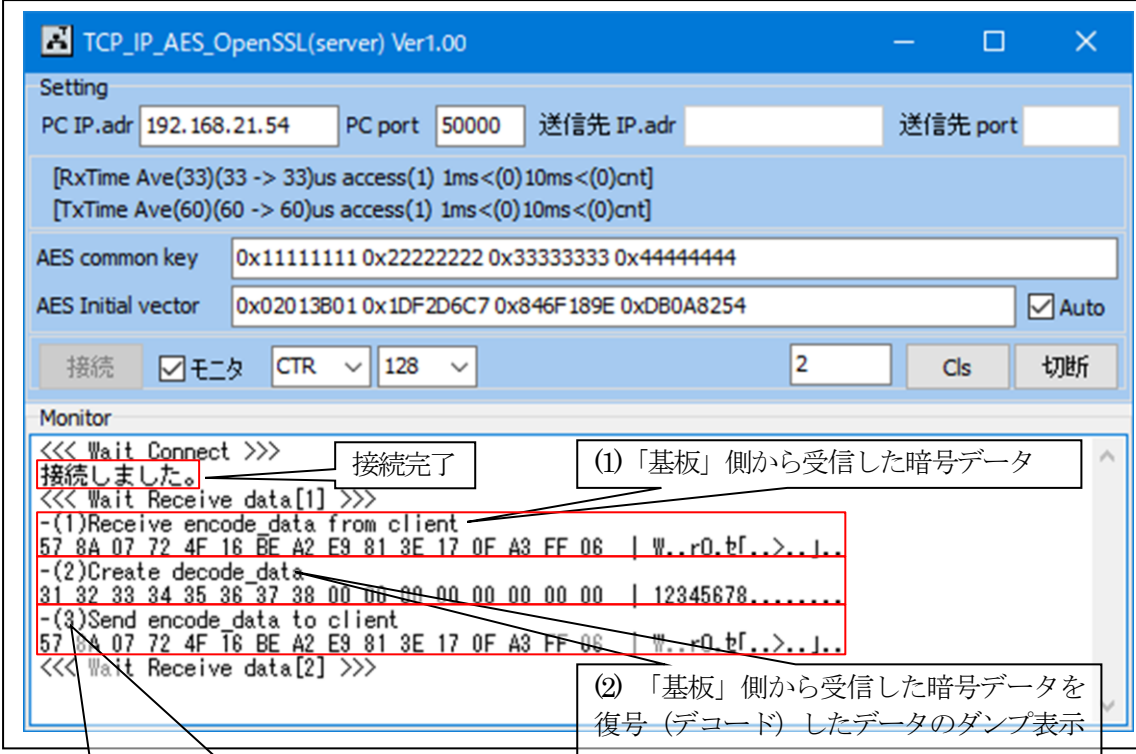
Annotations in the image explain these steps:

- Annotation 1: 'PC(server)への接続処理' (Connection processing to PC(server)).
- Annotation 2: '(1)入力した平文のダンプ表示' (Dump display of the input plaintext).
- Annotation 3: '(2)入力した平文を暗号化(エンコード)してPC(Server)に送信したデータのダンプ表示' (Dump display of data transmitted to PC(Server) after encoding the input plaintext).
- Annotation 4: '(3)PC(server)が受信した暗号文を復号(デコード)した文章をPC(server)で再度暗号化(エンコード)したデータを受信したダンプ表示' (Dump display of data received after re-encoding the decoded ciphertext on the PC(server)).
- Annotation 5: '(4)受信したPCからの暗号文を復号(デコード)したデータのダンプ表示' (Dump display of data received after decoding the ciphertext from the PC).

② 「(1)Plain data」と「(4)Create decode_data」が同等のダンプ表示の場合、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ります。

③ 「(2)Send encode_data to server」と「(3)Receive encode_data from server(Abort with Ctrl-C)」が同等のダンプ表示の場合、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ります。

5) 「TCP_IP_AES_OpenSSL」側の送受信を確認する。



The screenshot shows the 'TCP_IP_AES_OpenSSL(server) Ver1.00' application. The 'Setting' section includes fields for PC IP.adr (192.168.21.54), PC port (50000), and AES keys. The 'Monitor' window displays the following log output:

```

  <<< Wait Connect >>>
  接続しました。
  <<< Wait Receive data[1] >>>
  -(1)Receive encode_data from client
  57 8A 07 72 4F 16 BE A2 E9 81 3E 17 0F A3 FF 06 | W...r0.t[...]...
  -(2)Create decode_data
  31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 | 12345678.....
  -(3)Send encode_data to client
  57 8A 07 72 4F 16 BE A2 E9 81 3E 17 0F A3 FF 06 | W...r0.t[...]...
  <<< Wait Receive data[2] >>>
  
```

Annotations in the image point to specific log entries:

- ① 「基板」側から受信した暗号データ (points to the hex dump in step 1)
- ② 「基板」側から受信した暗号データを復号 (デコード) したデータのダンプ表示 (points to the hex dump in step 2)
- ③ 復号 (デコード) したデータを「PC」側で暗号 (エンコード) して「基板」側に送信したデータのダンプ表示 (points to the hex dump in step 3)

- ① 「(1)Receive encode_data from client」と「(3)Send encode_data to client」が同等のダンプ表示の場合、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ります。
- ② 「(2)Create decode_data」と「基板」側 TeraTerm 画面の「(4)Create decode_data」が同等のダンプ表示の場合、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ります。

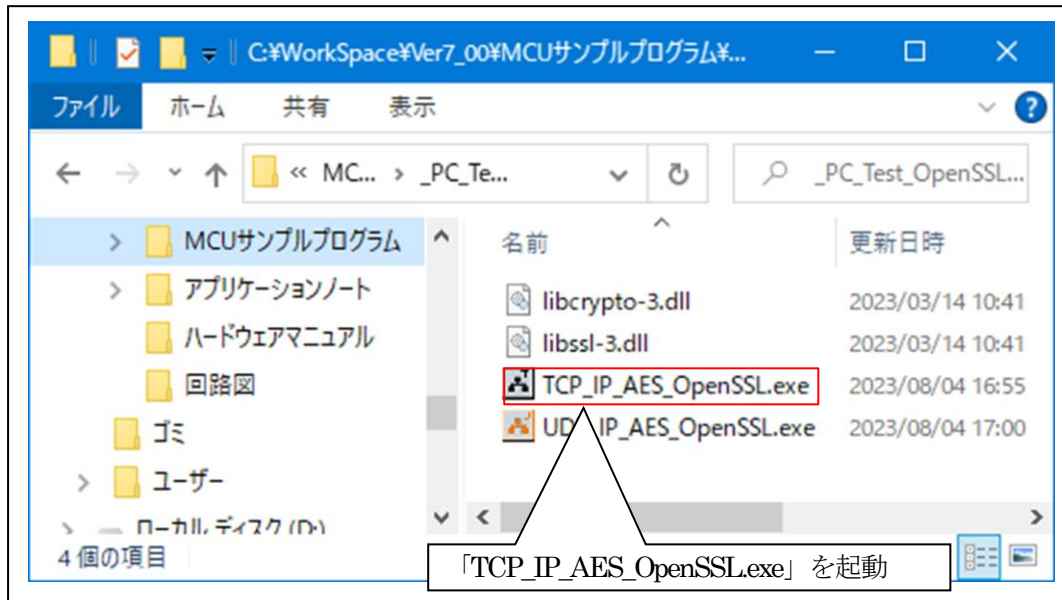
【Error 表示】

- ・受信時の接続失敗 「"error!! client_socket_connect()[error code]"」
- ・受信異常 「"error!! recvfrom()[error code]"」
- ・送信異常 「"error!! sendto()[error code]"」

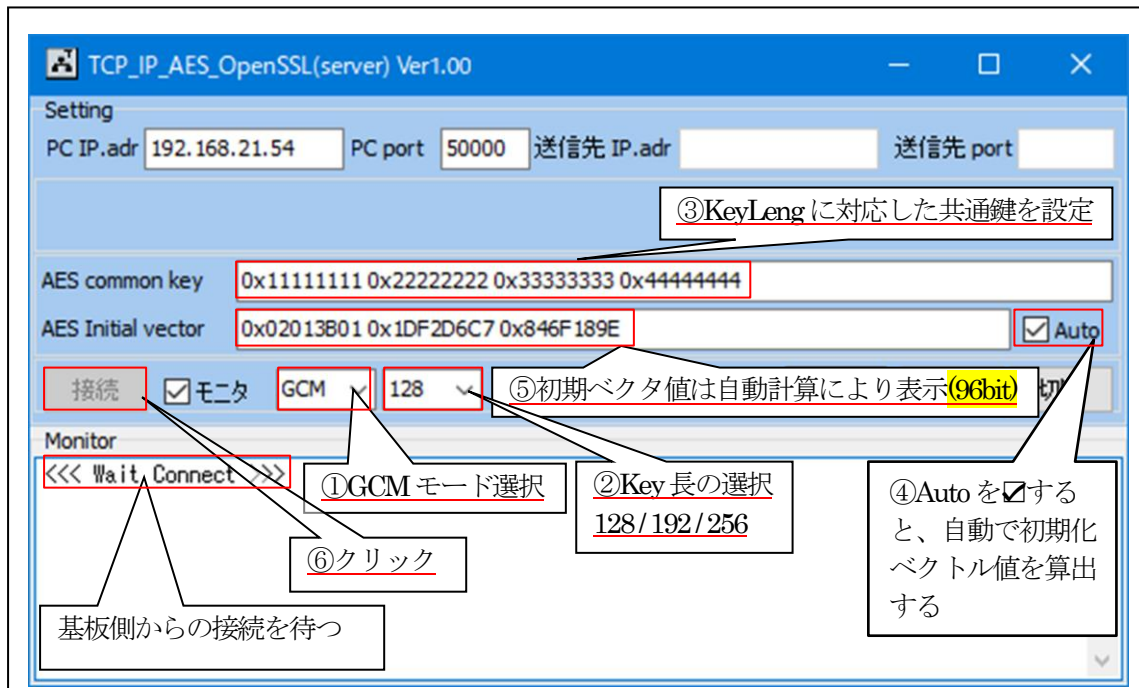
7-7. Windows PC 側のテスト用プログラムで動作確認 (AES-GCM モード)

1) 「TCP_IP_AES_OpenSSL」を起動する。

プログラム場所【ご購入 CD¥MCU サンプルプログラム¥_PC_Test_OpenSSL】を Read/Write 可能な適当なフォルダに Copy してからお使い下さい。



2) 「TCP_IP_AES_OpenSSL」の各項目を設定して「基板」側からの「接続」を待つ。



3) 「基板」側の各項目の確認と設定。

①PC側のIPアドレスを設定する。

ex) sendip 192.168.21.54<↵

②PC側のポート番号を設定する。

ex) sendport 50000<↵

③「初期ベクタ値」を指定する。

PC側の「UDP_IP_AES_OpenSSL」と同等の初期ベクタ値(AES Initial vector)にする。

ex) iv 0x02013B01 0x1DF2D6C7 0x846F189E 0x01000000<↵

AES-GCMのIVの96bit以降は、必ず、バイト表記で **0x00,0x00,0x00,0x01** にする必要があります。

④「GCM」(AES-GCM) モードを指定する。

ex) mode gcm<↵

⑤「Key Leng」を指定する。(128 / 192 / 256)

PC側の「UDP_IP_AES_OpenSSL」と同じKey長にする。

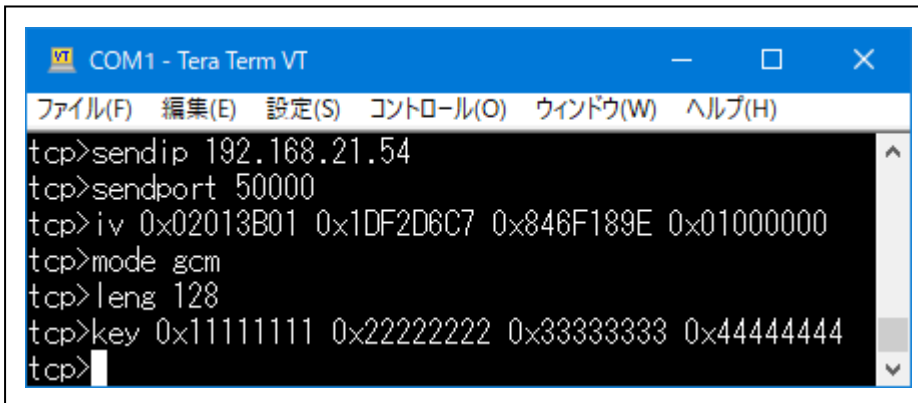
ex) leng 128<↵

⑥「共通鍵」を指定する。

PC側の「UDP_IP_AES_OpenSSL」と同等の共通鍵(AES common key)にする。

ex) key 0x11111111 0x22222222 0x33333333 0x44444444<↵

⑦実行画面



```

COM1 - 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)
tcp>sendip 192.168.21.54
tcp>sendport 50000
tcp>iv 0x02013B01 0x1DF2D6C7 0x846F189E 0x01000000
tcp>mode gcm
tcp>leng 128
tcp>key 0x11111111 0x22222222 0x33333333 0x44444444
tcp>
  
```

4) 「基板」側から「PC」(server)側へ GCM 暗号テキストを送信する。

①テキスト文を送信する。(未接続の場合は、接続を実行)

ex) send 12345678

```

COM1 - 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)
tcp>send 12345678
<The client_socket_bind.>
<The client_socket_port_get.>
<nx_tcp_client_socket_port_get()[50001]>
<The client_socket_connect.>
<Connect completed.>
[Now Key]: 0x11111111 0x22222222 0x33333333 0x44444444
[Now IV ]: 0x02013B01 0x1D52D6C7 0x846F189E 0x01000000
-(1)Plain data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
-(2)Send encode_data to server
DE CC 3F B1 32 8A BD 8B C5 87 CC 5E 44 A1 EB 84 | `??A2.ス.ナ.フ^D...
6E 82 3D F7 6D 24 6B 86 97 05 00 AC CE 46 CD A3 | n.三.m$k....ヤF^I
-(3)Receive encode_data from server(Abort with Ctrl-C)
DE CC 3F B1 32 8A BD 8B C5 87 CC 5E 44 A1 EB 84 | `??A2.ス.ナ.フ^D...
18 B2 49 68 88 F5 29 3B B5 B5 D1 96 C8 93 91 C3 | イlh8.);カムネテ
-(4)Create decode_data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
tcp>
    
```

PC(server)への接続処理

(1)入力した平文のダンプ表示

(2)入力した平文を暗号化 (エンコード) したデータと **認証用 TAG データ (16byte)**を PC (Server) に送信したデータのダンプ表示

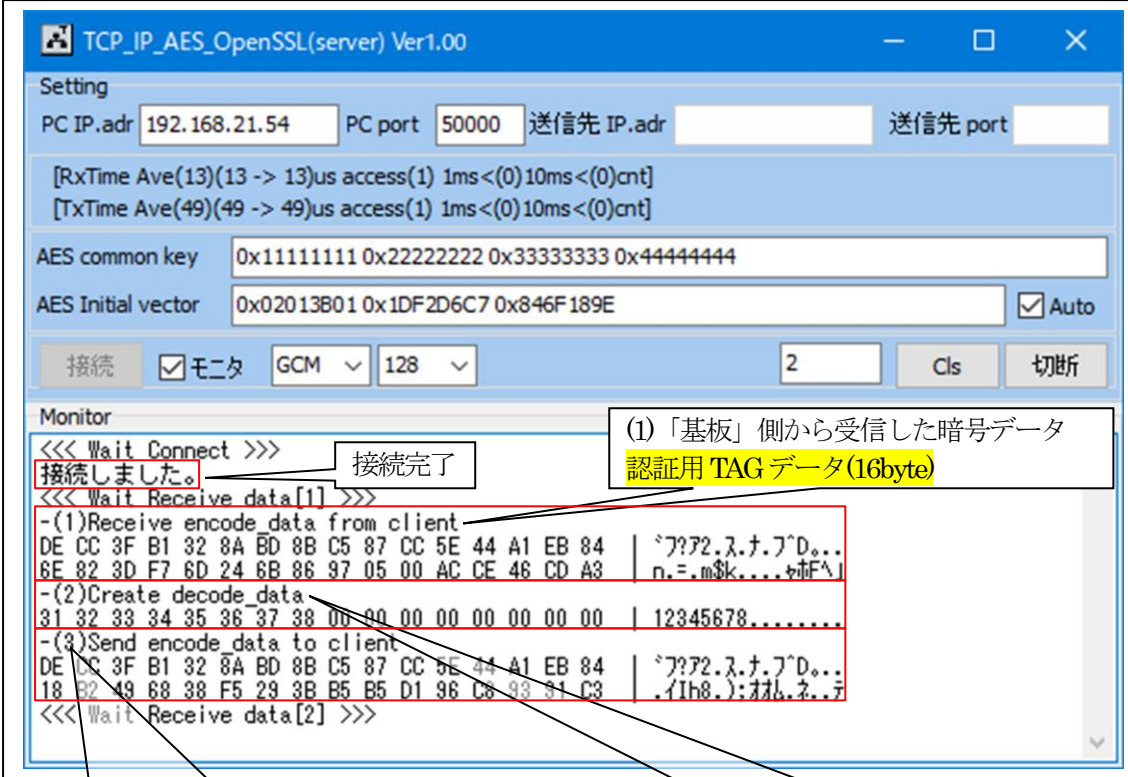
(3)PC (server) が受信した暗号文を復号 (デコード) したデータと **認証用 TAG データ (16byte)**を PC (server) で再度暗号化 (エンコード) したデータを受信したダンプ表示

(4)受信した PC からの暗号文を復号 (デコード) したデータのダンプ表示

② 「(1)Plain data」と 「(4)Create decode_data」が同等のダンプ表示の場合、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ります。

③ 「(2)Send encode_data to server」と 「(3)Receive encode_data from server(Abort with Ctrl-C)」が同等のダンプ表示の場合、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ります。

5) 「TCP_IP_AES_OpenSSL」側の送受信を確認する。



The screenshot shows the 'TCP_IP_AES_OpenSSL(server) Ver1.00' application window. The 'Setting' section includes PC IP address (192.168.21.54), PC port (50000), and AES common key/initial vector. The 'Monitor' section displays a log of operations:

```

  <<< Wait Connect >>>
  接続しました。
  <<< Wait Receive data[1] >>>
  -(1)Receive encode_data from client
  DE CC 3F B1 32 8A BD 8B C5 87 CC 5E 44 A1 EB 84 | '???.?.?.?D...
  6E 82 3D F7 6D 24 6B 86 97 05 00 AC CE 46 CD A3 | n.=.m$K....+F^J
  -(2)Create decode_data
  31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 | 12345678.....
  -(3)Send encode_data to client
  DE CC 3F B1 32 8A BD 8B C5 87 CC 5E 44 A1 EB 84 | '???.?.?.?D...
  18 82 49 68 38 F5 29 3B B5 B5 D1 96 C8 93 81 C3 | .(h8.):.h..s..?
  <<< Wait Receive data[2] >>>
  
```

Annotations in the image:

- (1) 「基板」側から受信した暗号データ 認証用 TAG データ(16byte)
- (2) 「基板」側から受信した暗号データを復号 (デコード) したデータのダンプ表示
- (3) 復号 (デコード) したデータを「PC」側で暗号 (エンコード) と認証用 TAG データ(16byte)を「基板」側に送信したデータのダンプ表示

- ① 「(1)Receive encode_data from client」と「(3)Send encode_data to client」が同等のダンプ表示の場合、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ります。
- ② 「(2)Create decode_data」と「基板」側 TeraTerm 画面の「(4)Create decode_data」が同等のダンプ表示の場合、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ります。

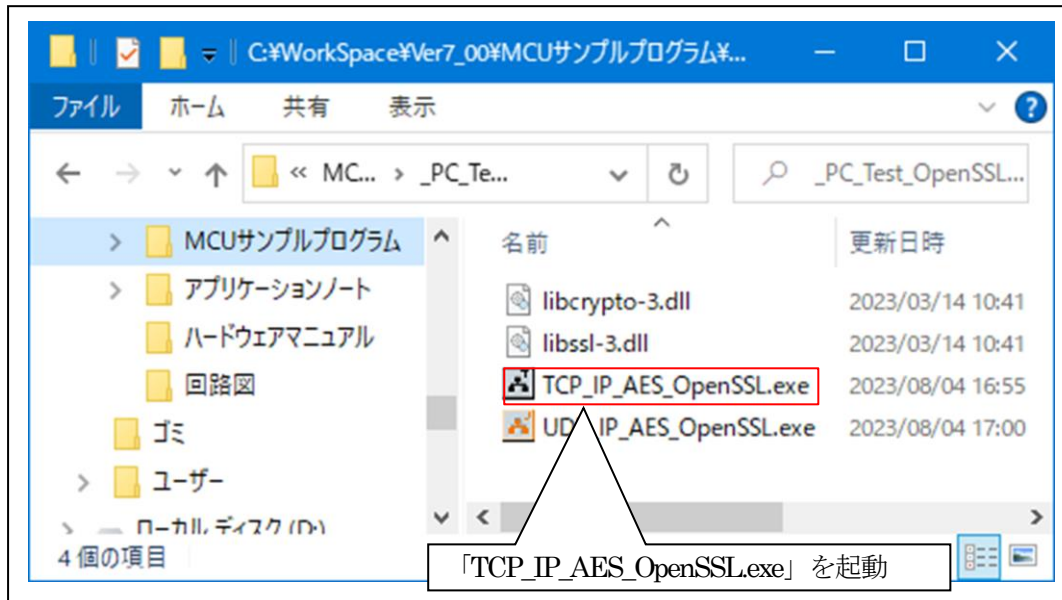
【Error 表示】

- ・受信時の接続失敗 「"error!! client_socket_connect()[error code]"
- ・受信異常 「"error!! recvfrom()[error code]"
- ・送信異常 「"error!! sendto()[error code]"

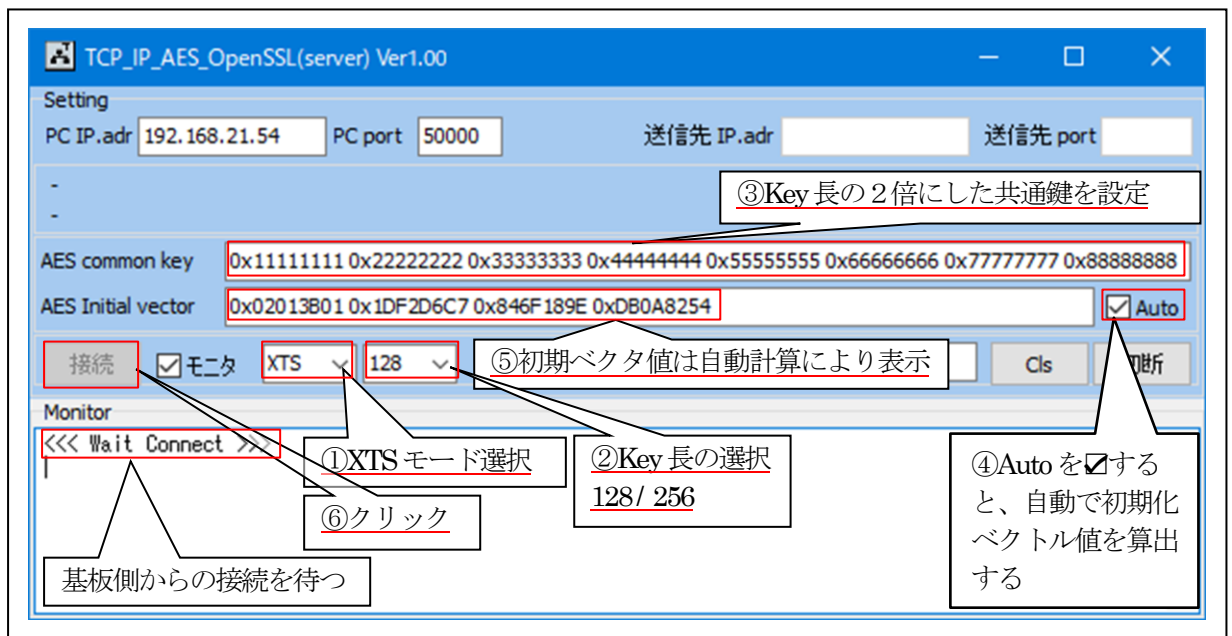
7-8. Windows PC 側のテスト用プログラムで動作確認 (AES-XTS モード)

1) 「TCP_IP_AES_OpenSSL」を起動する。

プログラム場所【ご購入 CD¥MCU サンプルプログラム¥_PC_Test_OpenSSL】を Read/Write 可能な適当なフォルダに Copy してからお使い下さい。



2) 「TCP_IP_AES_OpenSSL」の各項目を設定して「基板」側からの「接続」を待つ。



3) 「基板」側の各項目の確認と設定。

①PC側のIPアドレスを設定する。

ex) sendip 192.168.21.54<↵

②PC側のポート番号を設定する。

ex) sendport 50000<↵

③「XTS」(AES-XTS) モードを指定する。

ex) mode ctr<↵

④「Key Leng」を指定する。(128 / 256)

PC側の「TCP_IP_AES_OpenSSL」と同じKey長にする。

ex) leng 128<↵

⑤「共通鍵」を指定する。

PC側の「UDP_IP_AES_OpenSSL」と同等の共通鍵(AES common key)にする。

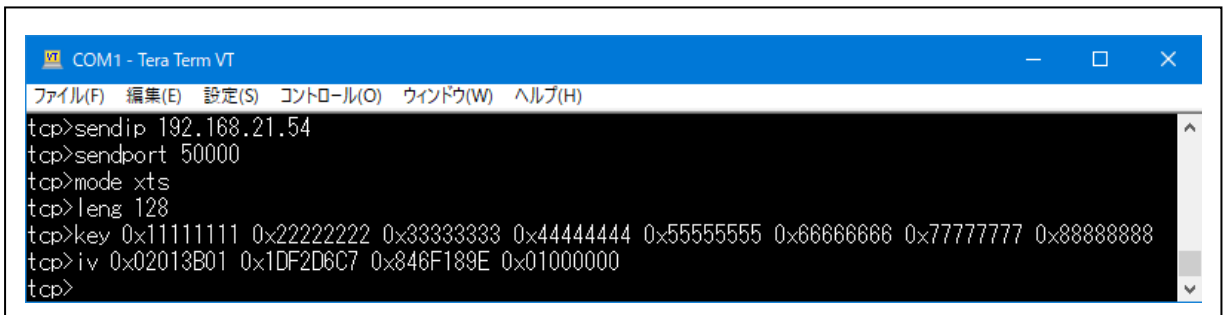
ex) key 0x11111111 0x22222222 0x33333333 0x44444444 0x55555555 0x66666666 0x77777777 0x88888888<↵

⑥「初期ベクタ値」を指定する。(HEX 数値)

PC側の「TCP_IP_AES_OpenSSL」と同等の初期ベクタ値(AES Initial vector)にする。

ex) iv 02013B01 1DF2D6C7 846F189E DB0A8254<↵

⑦実行画面



```

COM1 - 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)
tcp>sendip 192.168.21.54
tcp>sendport 50000
tcp>mode xts
tcp>leng 128
tcp>key 0x11111111 0x22222222 0x33333333 0x44444444 0x55555555 0x66666666 0x77777777 0x88888888
tcp>iv 0x02013B01 0x1DF2D6C7 0x846F189E 0x01000000
tcp>
  
```

4) 「基板」側から「PC」(server)側へXTS暗号テキストを送信する。

①テキスト文を送信する。(未接続の場合は、接続を実行)

ex) send 12345678

```

COM1 - 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)
tcp>send 12345678
<The client_socket_bind.>
<The client_socket_port_get.>
<nx_tcp_client_socket_port_get()[50001]>
<The client_socket_connect.>
<Connect completed.>
[Now Key]: 0x11111111 0x22222222 0x33333333 0x44444444
           0x55555555 0x66666666 0x77777777 0x88888888
[Now IV ]: 0x02013B01 0x15F2D6C7 0x846F189E 0xDB0A8254
-(1)Plain data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
-(2)Send encode_data to server
59 33 F4 A4 A9 82 23 A3 38 5C 69 28 60 52 91 19 | Y3..う.#J8¥i(`R..
-(3)Receive encode_data from server(Abort with Ctrl-C)
59 33 F4 A4 A9 82 23 A3 38 5C 69 28 60 52 91 19 | Y3..う.#J8¥i(`R..
-(4)Create decode_data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 00 | 12345678.....
tcp>
    
```

PC(server)への接続処理

(1)入力した平文のダンプ表示

(2)入力した平文を暗号化(エンコード)してPC(Server)に送信したデータのダンプ表示

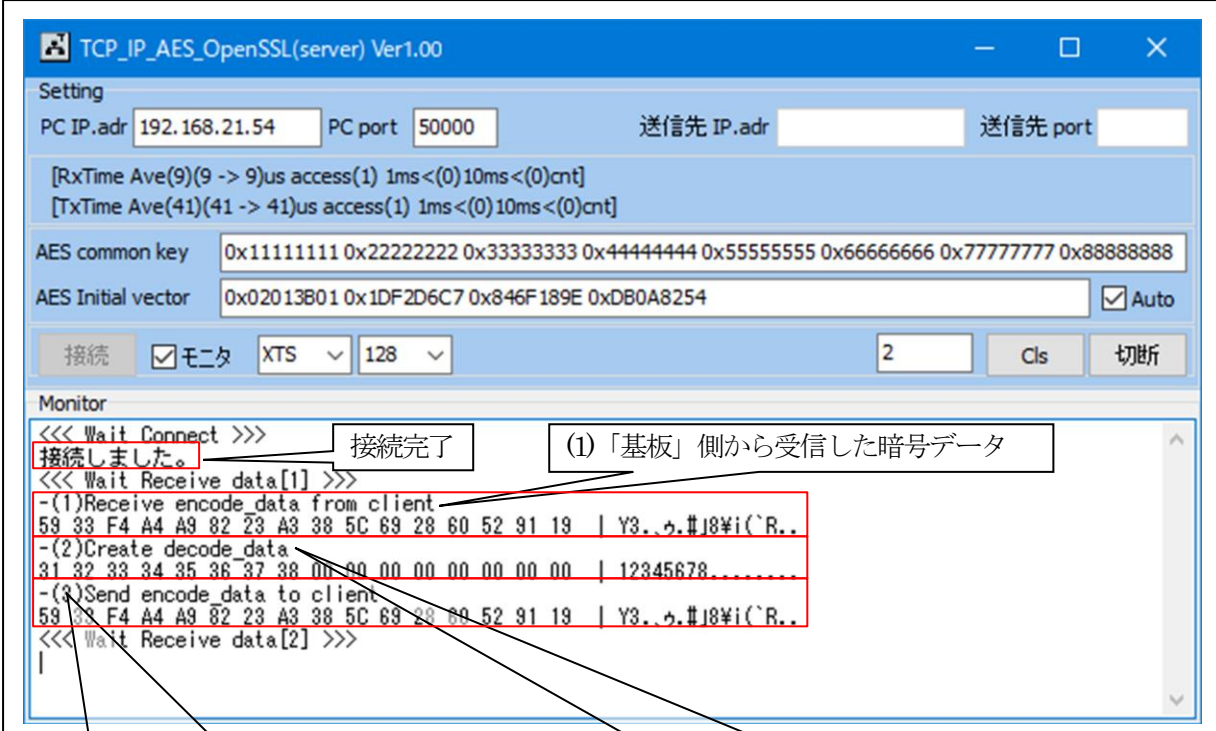
(3)PC(server)が受信した暗号文を復号(デコード)した文章をPC(server)で再度暗号化(エンコード)したデータを受信したダンプ表示

(4)受信したPCからの暗号文を復号(デコード)したデータのダンプ表示

② 「(1)Plain data」と「(4)Create decode_data」が同等のダンプ表示の場合、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ります。

③ 「(2)Send encode_data to server」と「(3)Receive encode_data from server(Abort with Ctrl-C)」が同等のダンプ表示の場合、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ります。

5) 「TCP_IP_AES_OpenSSL」側の送受信を確認する。



Setting

PC IP.adr 192.168.21.54 PC port 50000 送信先 IP.adr 送信先 port

[RxTime Ave(9)(9 -> 9)us access(1) 1ms<(0)10ms<(0)cnt]
[TxTime Ave(41)(41 -> 41)us access(1) 1ms<(0)10ms<(0)cnt]

AES common key 0x11111111 0x22222222 0x33333333 0x44444444 0x55555555 0x66666666 0x77777777 0x88888888
AES Initial vector 0x02013B01 0x1DF2D6C7 0x846F189E 0xDB0A8254 Auto

接続 モニタ XTS 128 2 Cls 切断

Monitor

```

<<< Wait Connect >>>
接続しました。
接続完了
(1)「基板」側から受信した暗号データ
<<< Wait Receive data[1] >>>
-(1)Receive encode_data from client
59 33 F4 A4 A9 82 23 A3 38 5C 69 28 60 52 91 19 | Y3..ふ.#18¥i(`R..
-(2)Create decode_data
31 32 33 34 35 36 37 38 00 00 00 00 00 00 00 | 12345678.....
-(3)Send encode_data to client
59 33 F4 A4 A9 82 23 A3 38 5C 69 28 60 52 91 19 | Y3..ふ.#18¥i(`R..
<<< Wait Receive data[2] >>>
  
```

(3)復号 (デコード) したデータを「PC」側で暗号 (エンコード) して「基板」側に送信したデータのダンプ表示

(2)「基板」側から受信した暗号データを復号 (デコード) したデータのダンプ表示

- ① 「(1)Receive encode_data from client」と「(3)Send encode_data to client」が同等のダンプ表示の場合、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ります。
- ② 「(2)Create decode_data」と「基板」側 TeraTerm 画面の「(4)Create decode_data」が同等のダンプ表示の場合、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ります。

【Error 表示】

- ・受信時の接続失敗 「"error!! client_socket_connect()[error code]"
- ・受信異常 「"error!! recvfrom()[error code]"
- ・送信異常 「"error!! sendto()[error code]"

7-6. デバッグの終了

☆詳細操作は「[e2studio_synergy_Import.pdf](#)」の3-3項を参照して下さい。

8. 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。

9. 商標

- ・e2studio 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。
- ・Renesas Synergy[™]および S3A7/S5D9/S7G2 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。

10. 参考文献

- ・「S3A7 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
- ・「S5D9 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
- ・「S7G2 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
- ・ルネサス エレクトロニクス株式会社提供のサンプル集
- ・「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
- ・「SSP vx.x.x User's Manual」 ルネサス エレクトロニクス株式会社
- ・「X-Ware Component Documents for Renesas Synergy[™]」 ルネサス エレクトロニクス株式会社
- ・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.robin-w.com>

