

Renesas R5F565N9(RX65N-1MB)用サンプル

(e2studio RX65N_gnu_udp_client_AES_uselib)の説明

(e2studio Version:2022-07 / Azure RTOS Version 6.2.1 rel-rx-2.0.0)

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 |
|------------------|--|---|
| Azure_sample_gnu | <p>有線 LAN 接続した DHCP と UDP 通信のサンプル</p> <p>セキュリティ API Crypto ドライバー AES-CBC を使用したサンプル (暗号・復号)</p> | <p>RX65N_gnu_udp_client_AES_uselib</p> <p>Azure RTOS モードで動作</p> <p>NetX DHCP Client (dhcp_client)</p> <p>UDP 通信(Client) (nx_udp_socket_.....)</p> <p>暗号・復号(AES-CBC) (NX_CRYPT0_ENCRYPT) (NX_CRYPT0_DECRYPT)</p> |

| |
|--|
| 統合開発環境 |
| Renesas e2studio(Version 2022-07) |
| Azure RTOS (Version 6.2.1 rel-rx 2.0.0) |
| GCC for Renesas RX(Version 8.3.0.202305) |

| |
|--------------------------|
| ハード環境 |
| AP-RX65N-0A(アルファプロジェクト製) |

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。
- ②PC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

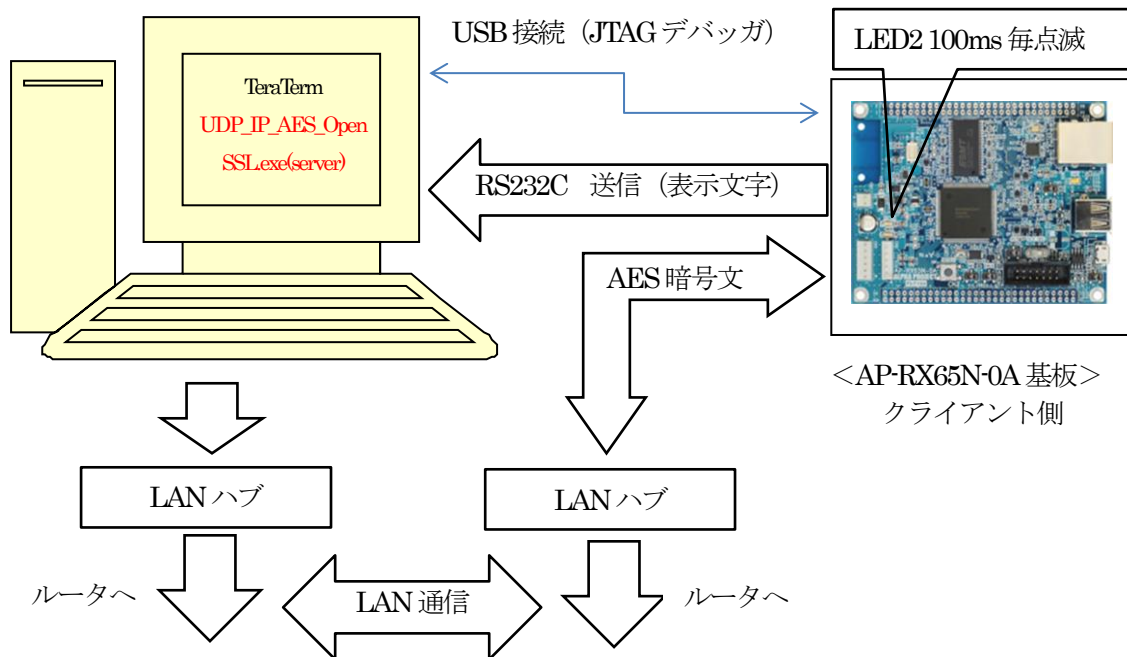
の仕様にする。

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 の処理＞

【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。

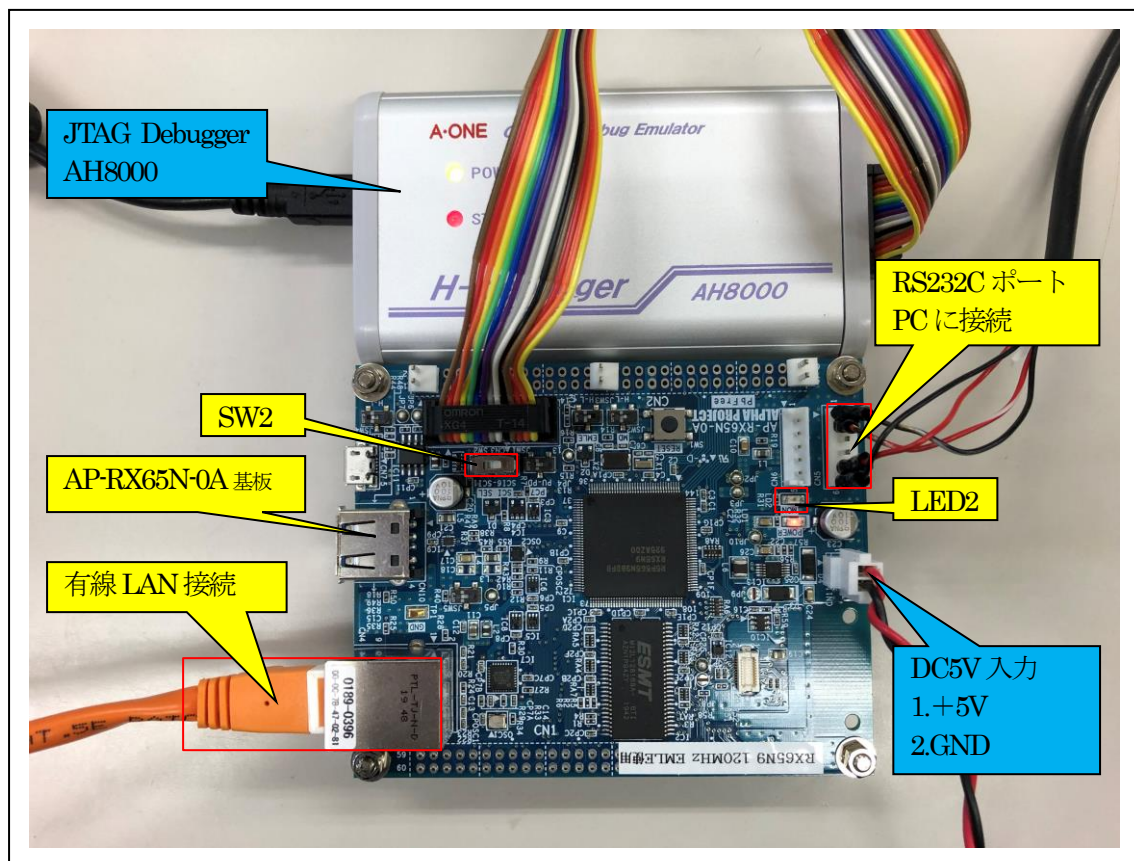
＜RX65N_gnu_udp_client_AES_uselib の処理＞

【Plain mode】

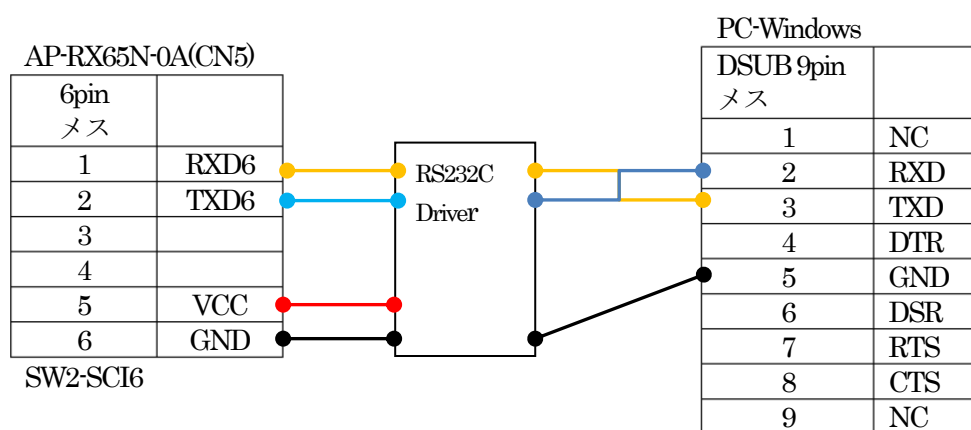
1. 平文を送信する。
2. 受信した平文を TeraTerm に表示する。

【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。



- ①Windows-PC 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. 「RX65N_gnu_udp_client_AES_uselib」 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】

| | | | |
|--------------------|--------------------------------------|--|-----------------------------------|
| <Azure_sample_gnu> | | | |
| | <rx_gnu_filex_lib> | AzureRTOS FileX ライブラリ作成用ホルダ | |
| | <rx_gnu_netxduo_addons_lib> | AzureRTOS NetX DuoAddons ライブラリ作成用ホルダ | |
| | <rx_gnu_netxduo_lib> | AzureRTOS NetX Duo ライブラリ作成用ホルダ | |
| | <rx_gnu_threadx_lib> | AzureRTOS ThredX ライブラリ作成用ホルダ | |
| | <RX65N_gnu_udp_client_AES_uselib> | DHCP / UDP 通信 / AES-CBC(暗号・復号) サンプルプロジェクト | |
| | <HardwareDebug> | RX65N_gnu_udp_client_AES_uselib.elf | ELF ファイル、JTAG で使用 |
| | | RX65N_gnu_udp_client_AES_uselib.map | MAP ファイル、アドレス情報 |
| | | RX65N_gnu_udp_client_AES_uselib.mot | モトローラーHEX ファイル |
| | | その他 | 自動生成ファイル |
| | <lib> | <filex> | FileX (全C ソースはビルド除外) |
| | | <netxduo> | NetX Duo (全C ソースはビルド除外) |
| | | <netxduo_addons> | NetX Duo Addons (全C ソースはビルド除外) |
| | | <threadx> | ThreadX (全C ソースはビルド除外) |
| | | librx_gnu_filex_lib.a | FileX ライブラリ |
| | | librx_gnu_netxduo_addons_lib.a | NetX DuoAddons ライブラリ |
| | | librx_gnu_netxduo_lib.a | NetX ライブラリ |
| | | librx_gnu_threadx_lib.a | ThredX ライブラリ |
| | <src> | <rtos_config> | スマートコンフィグレータにより作成 |
| | | <smc_gen> | スマートコンフィグレータにより作成 |
| | | <rtos_skeleton> | |
| | | dhcp_fixed_entry.c | DHCP スレッド処理のソース |
| | | udp_aes_thread_entry.c | UDP スレッド処理のソース |
| | | <Azure_aes> | AES-CBC |
| | | aes.c aes.h | 暗号・復号処理のソース |
| | | 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 |
| | | nx_driver_rx_fit.c | Renesas RX FIT driver: Control |
| | | nx_driver_rx_fit.h | nx_driver_rx_fit.c のヘッダー |
| | | sample_netx_duo_ping.c | NetX 等初期化サンプルソース |
| | <trash> | 過去変更した<src>等のごみ箱 | |
| | RX65N_gnu_udp_client_AES_uselib.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 |
|-----------------------------------|---|---|
| NX_ENABLE_DHCP | 0 | DHCP Client Disable ◎ソースコードに直接 IP アドレスを記述 sample_netx_duo_ping.c : status = nx_ip_create(&ip_0, "NetX IP Instance 0", #if(NX_ENABLE_DHCP == 1) IP_ADDRESS(0,0,0,0), IP_ADDRESS(255,255,255,0), #else IP_ADDRESS(192,168,21,8), // 固定 IP アドレス IP_ADDRESS(255,255,255,0), // サブネットマスク #endif &pool_0, nx_driver_rx_fit, (UCHAR*)ip_thread_stack, sizeof(ip_thread_stack), 1); |
| | 1 | DHCP Client Enable |
| TX_INCLUDE_USER_DEFINE_FILE | | 「tx_user.h」を有効にする |
| NX_INCLUDE_USER_DEFINE_FILE | | 「nx_user.h」を有効にする |
| FX_INCLUDE_USER_DEFINE_FILE | | 「fx_user.h」を有効にする |
| NXD_MQTT_CLOUD_ENABLE | | MQTT メッセージングプロトコルを有効にする |
| NX_SECURE_ENABLE | | MQTT クライアントは TLS サポート付きで構築される |
| NX_ENABLE_EXTENDED_NOTIFY_SUPPORT | | 多くのコールバックフックを有効にする |
| NX_ENABLE_IP_PACKET_FILTER | | IP パケットを有効にする |
| FLATCC_NO_ASSERT | | FLATCC をアサートしない |
| NX_AZURE_IOT_LOG_LEVEL | 0 | NX_AZURE ログ関数を使用しない |
| | 1 | LogError(...)を使用する |
| | 2 | LogError(...)/LogInfo(...)を使用する |
| | 3 | LogError(...)/LogInfo(...)/LogDebug(...)を使用する |

5-3. サンプルの動作説明（基板側 AP-RX65N-0A）

1) DHCP 無効時（NX_ENABLE_DHCP=0）

<DHCP FIXED Thread>

Term 画面

- < 1 > 「"Driver INITIALIZE_DONE..."」
- < 2 > 「"<Wait for IP address acquisition>..."」
- < 3 > 「"LINK Status Check..."」
- < 4 > 「"Timer Creat..."」

<成功画面> IP アドレス確立により、基板上の LED2 を 100msec 毎に点滅

```

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)
Driver INITIALIZE_DONE...
<Wait for IP address acquisition >
LINK Status Check...
Timer Creat...

Successfully assigned by FIXED address(PRI)
[MAC ADDRESS]: 00-0c-7b-47-02-81
[MY IP.adr ]: 192.168.21.8
[SUBNET mask]: 255.255.255.0
[MTU size ]: 1514
[TCP port ]: 50001
[UDP port ]: 50002
<UDP socket create>
<UDP socket bind>
<Start NetX UDP AES AP-RX65N-0A[client]>

AES_mode[CBC] KeyLength[128] DEST_IP[192.168.21.50]
Press any key...[Press 'C' key to change AES mode...]
  
```

固定 IP アドレス

UDP ポート番号

<失敗> IP アドレス未確立により、基板上の LED2 を 500msec 毎に点滅

```

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)
Driver INITIALIZE_DONE...
<Wait for IP address acquisition >
LINK Status Check...
Timer Creat...
NX_NOT_SUCCESSFUL...
  
```

失敗

2) DHCP 有効時 (NX_ENABLE_DHCP=1)

<DHCP FIXED Thread>

Term 画面

- < 1 > 「Driver INITIALIZE_DONE...」
- < 2 > 「<Wait for IP address acquisition>..」
- < 3 > 「LINK Status Check...」
- < 4 > 「Timer Creat...」
- < 5 > 「DHCP In Progress...」
- < 6 > 「Wait until address resolved...」

<成功画面>IP アドレス取得により、基板上の LED2 を 100msec 毎に点滅

```
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)
Driver INITIALIZE_DONE...
<Wait for IP address acquisition >
LINK Status Check...
Timer Creat...
DHCP In Progress...
Wait until address resolved...

Successfully assigned by DHCP address(PRI)
[MAC ADDRESS]: 00-0c-7B-47-02-81
[MY IP.adr ]: 192.168.21.8
[SUBNET mask]: 255.255.255.0
[MTU size ]: 1514
[TCP port ]: 50001
[UDP port ]: 50002
<UDP socket create>
<UDP socket bind>
<Start NetX UDP AES AP-RX65N-0A[client]>

AES_mode[CBC] KeyLength[128] DEST_IP[192.168.21.50]
Press any key...[Press 'C' key to change AES mode...]
```

<失敗画面>IP アドレス未取得により、基板上の LED2 を 500msec 毎に点滅

```
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)
Driver INITIALIZE_DONE...
<Wait for IP address acquisition >
LINK Status Check...
Timer Creat...
DHCP In Progress...
Wait until address resolved.
NX_NOT_SUCCESSFUL...
```


3) UDP/IP 送受信
<UDPAESThread>

```

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)
Driver INITIALIZE_DONE...
<Wait for IP address acquisition >
LINK Status Check...
Timer Creat...
DHCP In Progress...
Wait until address resolved...

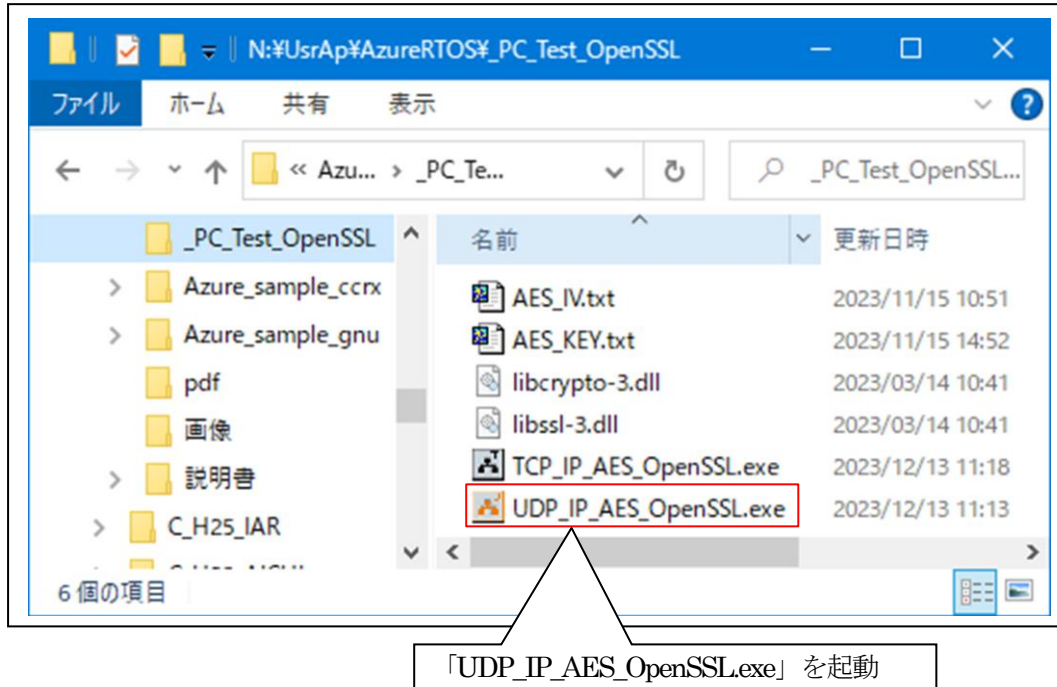
Successfully assigned by DHCP address(PRI)
[MAC ADDRESS]: 00-0c-7b-47-02-81
[MY IP.adr ]: 192.168.21.8
[SUBNET mask]: 255.255.255.0
[MTU size ]: 1514
[TCP port ]: 50001
[UDP port ]: 50002
<UDP socket create>
<UDP socket bind>
<Start NetX UDP AES AP-RX65N-0A[client]>
AES_mode[CBC] KeyLength[128] DEST_IP[192.168.21.50]
Press any key...[Press 'C' key to change AES mode...]
  
```

| 表示項目 | 表示内容 | 説明 |
|--------------------------|-------------------|---|
| AES_mode[x] | PLAIN 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; 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 ビット長の指定 ◎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 |
| DEST_IP[xxx.xxx.xxx.xxx] | 固定 | 送信先(PC 側)IP アドレス ◎define にて指定 udp_aes_thread_entry.c : #define DEST_IP IP_ADDRESS(192,168,xx,xx) |

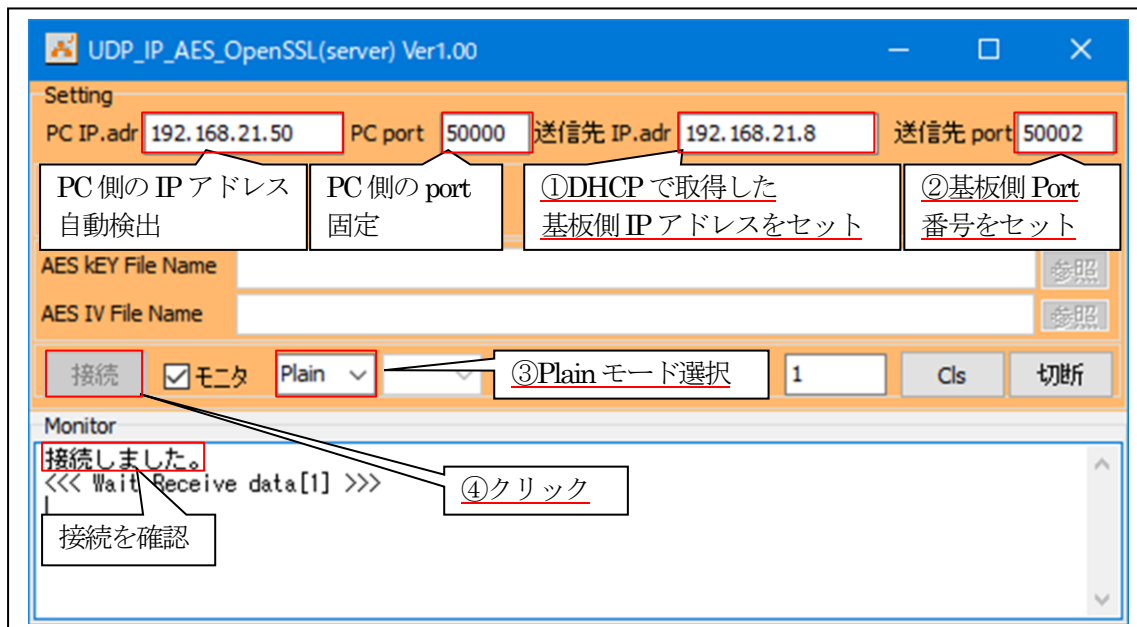
5-4. Windows PC 側のテストプログラムで動作確認 (PLAIN (平文) モード)

- 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

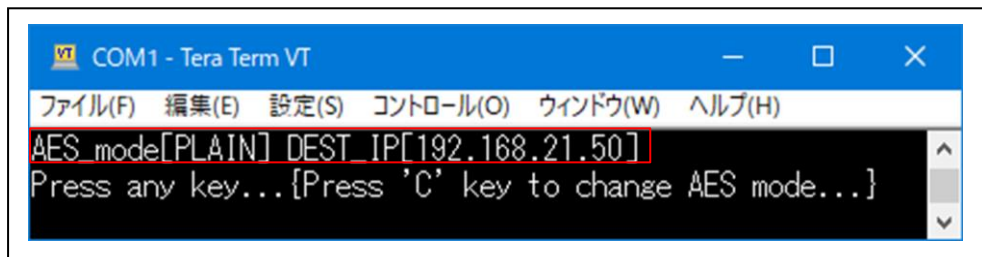


- 2) 「UDP_IP_AES_OpenSSL」の各項目を設定して接続する。

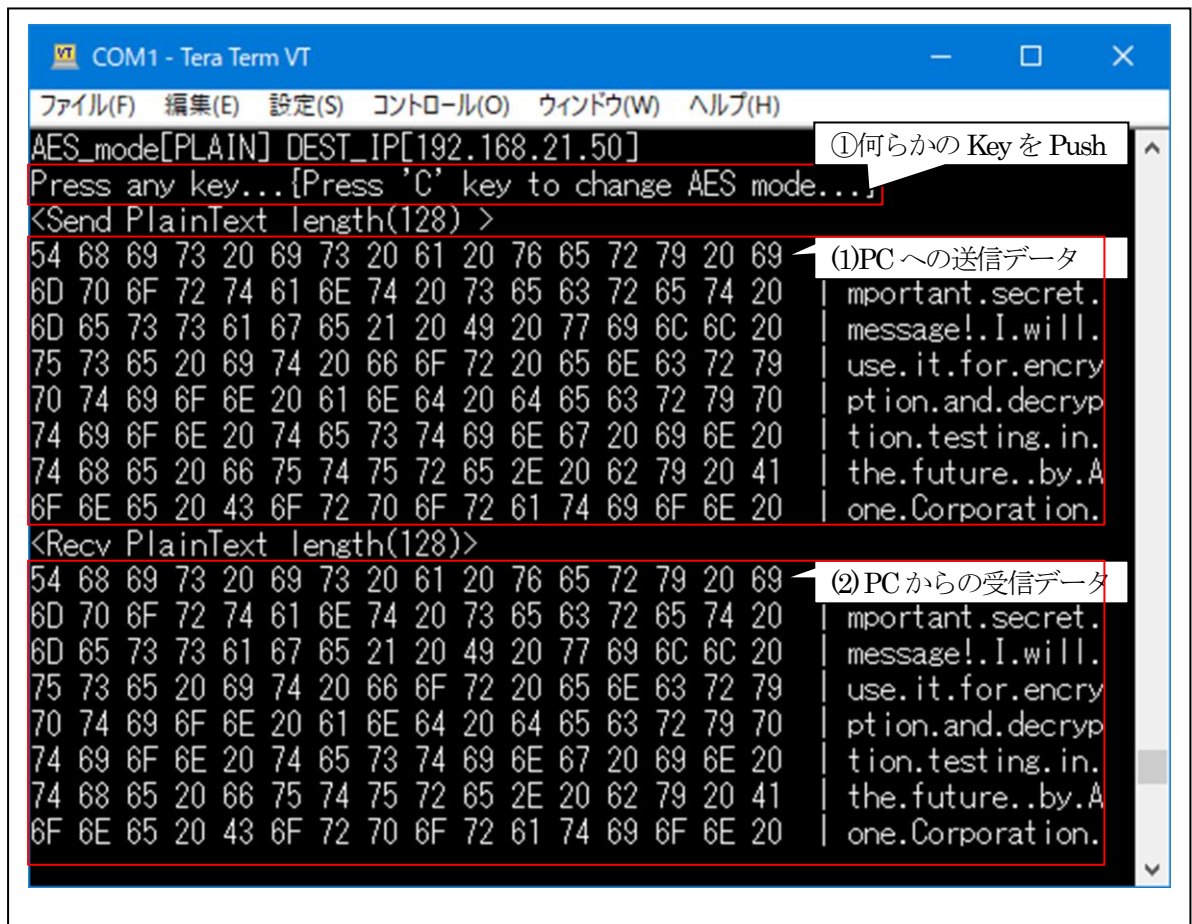


3) 基板側の各項目の確認と設定。

| 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 |
| DEST_IP[192.168.21.50] | 送信先(PC側)IPアドレス ◎defineにて指定 udp_aes_thread_entry.c : #define DEST_IP IP_ADDRESS(192,168,21,50) |



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 | 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 " }; |

6) 「UDP_IP_AES_OpenSSL」側の送受信を確認する。

UDP_IP_AES_OpenSSL(server) Ver1.00

Setting

PC IP.adr 192.168.21.50 PC port 50000 送信先 IP.adr 192.168.21.8 送信先 port 50002

[RxTime Ave(5)(5 -> 5)us access(1) 1ms<(0)10ms<(0)cnt]
[TxTime Ave(35)(35 -> 35)us access(1) 1ms<(0)10ms<(0)cnt]

AES key File Name 参照

AES IV File Name 参照

接続 ☒ モニタ Plain 2 Cls 切断

Monitor

接続しました。
<<< Wait Receive data[1] >>>
-(1)Receive plain data from client

(1)「基板」側からの受信データ

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 This.is.a.very.important.secret.message!.I.will.use.it.for.encryption.and.decryption.testing.in.the.future..by.Aone.Corporation.

-(2)Send plain data to client

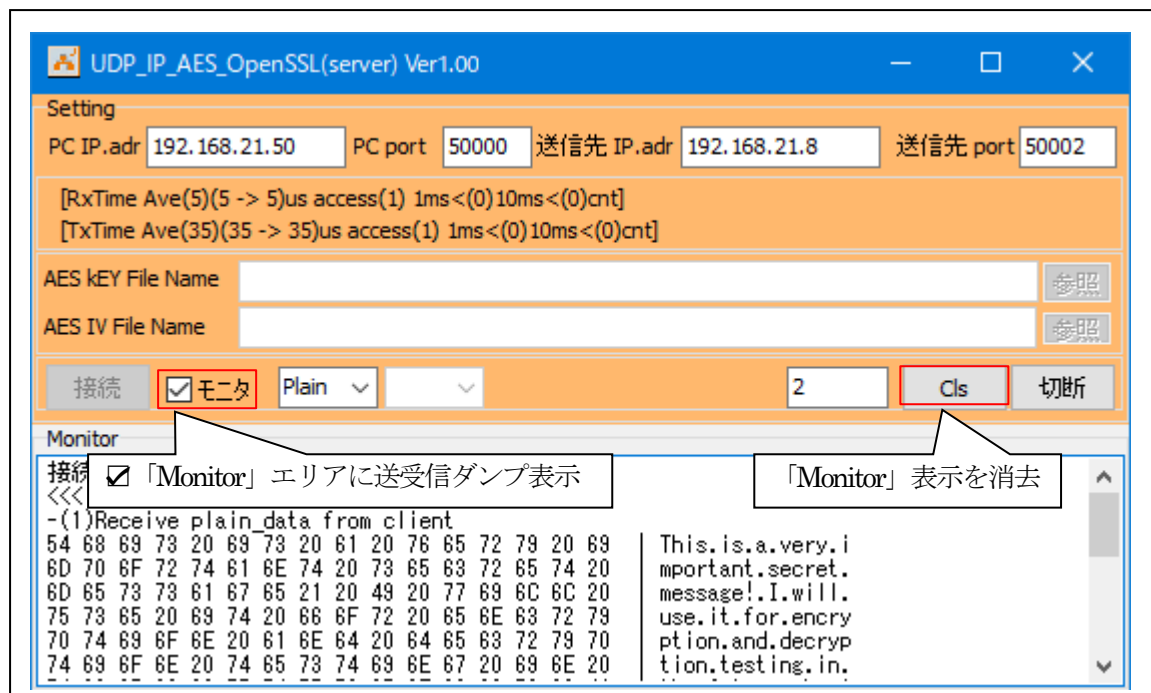
54 68 69 73 20 69 73 20 61 20 76 65 72 79 20 69 6D 70 6F 72 74 61 6E 74 20 73 65 63 72 65 74 20 6D 65 73 73 61 67 65 21 20 49 20 77 69 6C 6C 20 75 73 65 20 69 74 20 66 6F 72 20 65 6E 63 72 79 70 74 69 6F 6E 20 61 6E 64 20 64 65 63 72 79 70 74 69 6F 6E 20 74 65 73 74 69 6E 67 20 69 6E 20 74 68 65 20 66 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 This.is.a.very.important.secret.message!.I.will.use.it.for.encryption.and.decryption.testing.in.the.future..by.Aone.Corporation.

<<< Wait Receive data[2] >>>
|

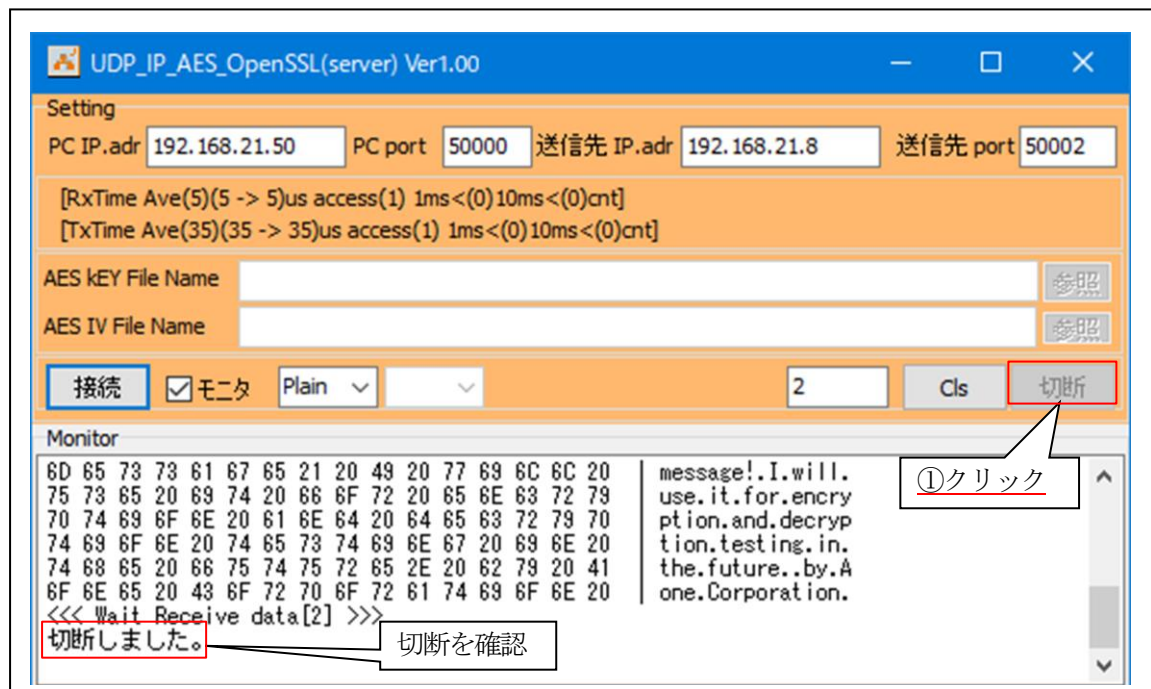
(2)「基板」側への送信データ

☆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)

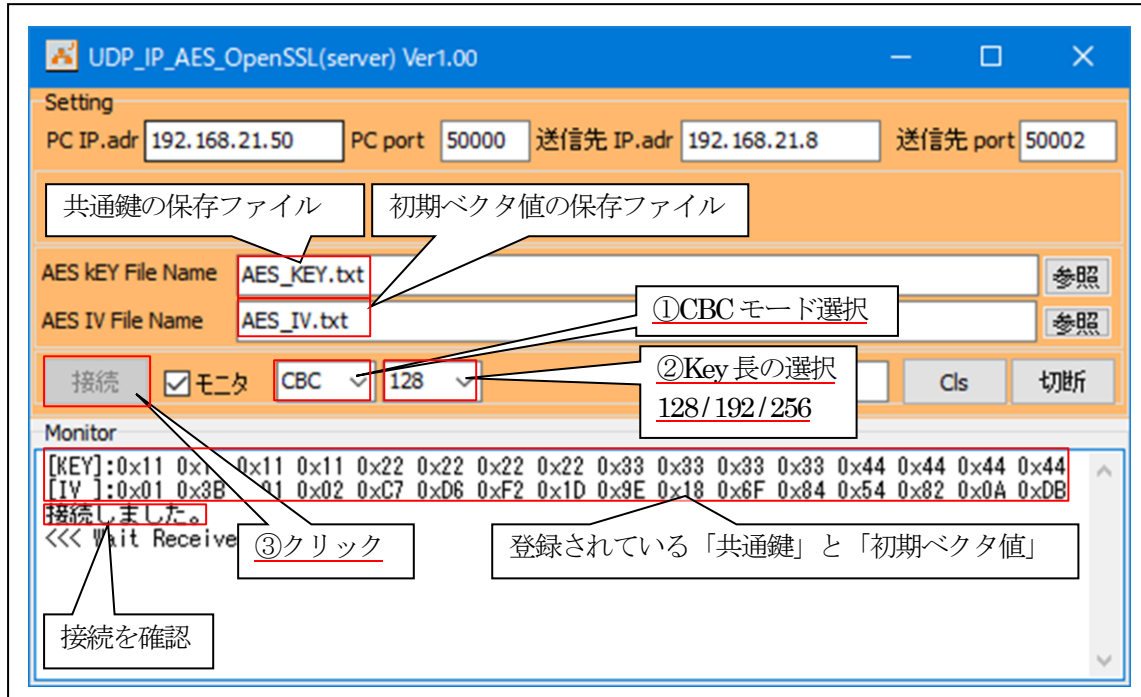


8) UDP_IP-Portを「切断」する。(各モード共通)



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 | |

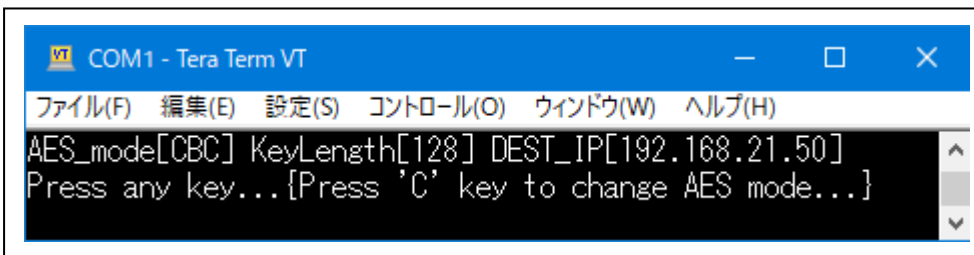
☆初期ベクタ値を変更する場合は、基板側と同等の初期ベクタ値を適当なエディタで変更する。

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; |
| DEST_IP[192.168.21.50] | 送信先(PC 側)IP アドレス ◎define にて指定 udp_aes_thread_entryc : #define DEST_IP IP_ADDRESS(192,168,21,50) |

| |
|---|
| 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。

```

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)

AES_mode[cbc] KeyLength[128] DEST_IP[192.168.21.50]
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 will use it for encryption and decryption 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 EE 18 DA 13 8F 91 7D 06 53 3A 33 BA
.D.eI.r.和....].C
..&..+$...i.'.アx
0.]-.ノ.ih.@i.'.-
.]ヲ..レ...}.S:3

<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:3

<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
use..it..for..encryp
tion..and..decryp
tion..testing..in..
the..future..by..A
one..Corporation..

```

☆受信処理が失敗した場合は「Ctrl+C」 Key-Push で中断する。

② 「(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。

③ 「(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。

UDP_IP_AES_OpenSSL(server) Ver1.00

Setting

PC IP.adr: 192.168.21.50 PC port: 50000 送信先 IP.adr: 192.168.21.8 送信先 port: 50002

[RxTime Ave(5)(5 -> 5)us access(1) 1ms<(0)10ms<(0)cnt]
[TxTime Ave(53)(53 -> 53)us access(1) 1ms<(0)10ms<(0)cnt]

AES key File Name: AES_KEY.txt 参照
AES IV File Name: AES_IV.txt 参照

接続 ☒ モニタ CBC 128 2 Cls 切断

Monitor

接続しました。
<<< Wait Receive data[1] >>>
-(1)Receive encode_data from client

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

(1)基板側から受信した暗号文のダンプ表示

-(2)Create decode_data

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

(2)受信した暗号文を復号化した復号文のダンプ表示

-(3)Send encode_data to client

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

(3)復号化した復号文を暗号化して基板側に送信した暗号文のダンプ表示

<<< Wait Receive data[2] >>>

- 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。
- ・本文章に関して、アルファプロジェクト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。
- ・AP-RX65N-0A は、株式会社アルファプロジェクトの商品名です。
- ・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
- ・「AzureRTOS」 マイクロソフト株式会社
- ・ルネサス エレクトロニクス株式会社提供のサンプル集
- ・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.in>

