Rev 1.00.00

Renesas R5F565NE(CK-RX65N)用サンプル

(e2studio RX65N_gnu_dhcp_tcp_aes_gt202_uselib)の説明

(e2studio Version:2024-04 / 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

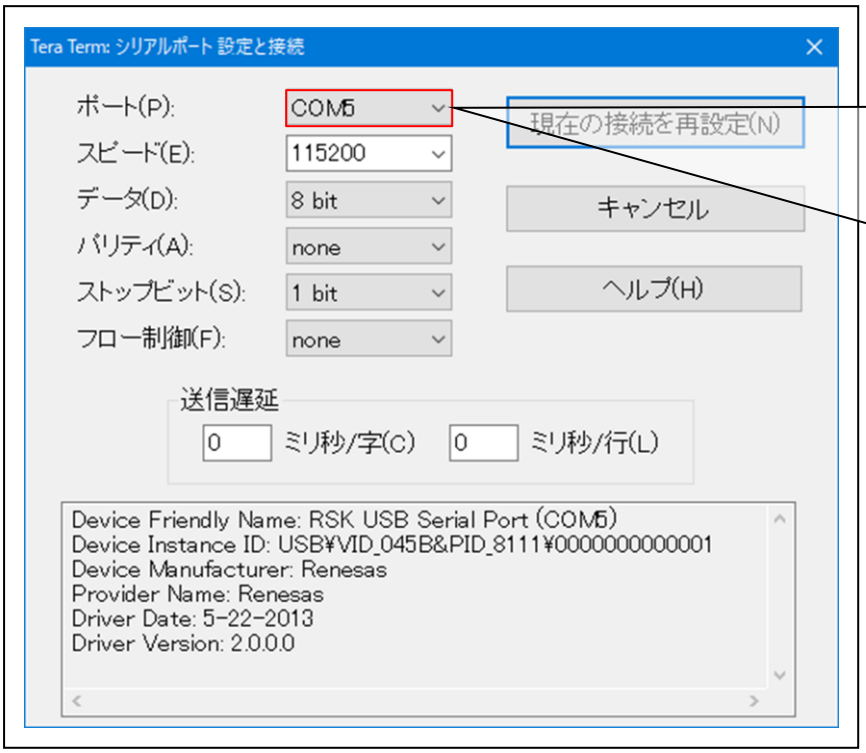
ワークスペース名	概要	プロジェクト名
Azure_sample_gnu_wifi_CK	無線 WiFi-module(GT202-QCA4002) を使用した DHCP と TCP 通信のサンプル セキュリティ API Crypto ドライバー AES-CBC を使用したサンプル（暗号・復号）	RX65N_gnu_dhcp_tcp_aes_gt202_uselib Azure RTOS モードで動作 NetX DHCP Client (dhcp_client) TCP 通信(Client) (nx_tcp_socket_.....) 暗号・復号(AES-CBC) (NX_CRYPTO_ENCRYPT) (NX_CRYPTO_DECRYPT)

統合開発環境
Renesas e2studio(Version 2024-04)
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
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。
②PC にインストールし実行する
③シリアルポートの設定

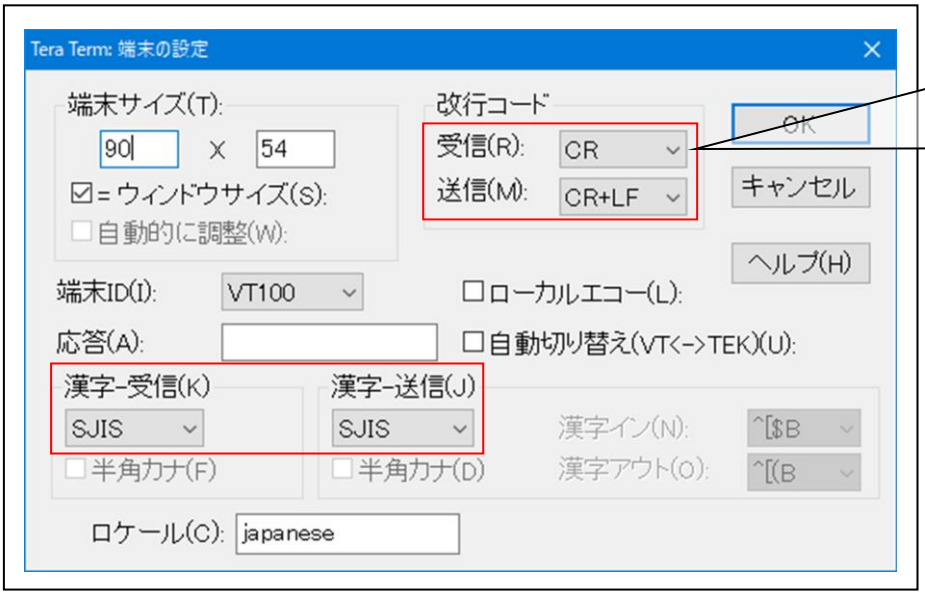


COM 番号は、
PC 側でシリアル通信可能な番号を指定する。

115200BPS
8bit
none
1bit
none

の仕様にする。

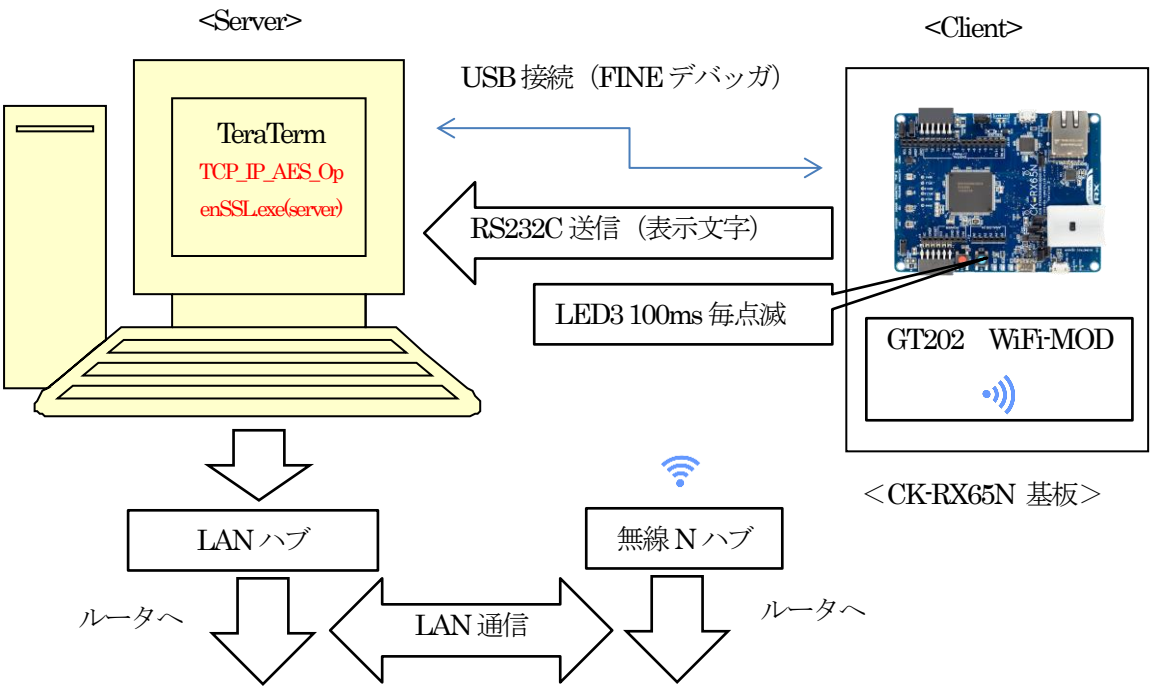
④端末の設定



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

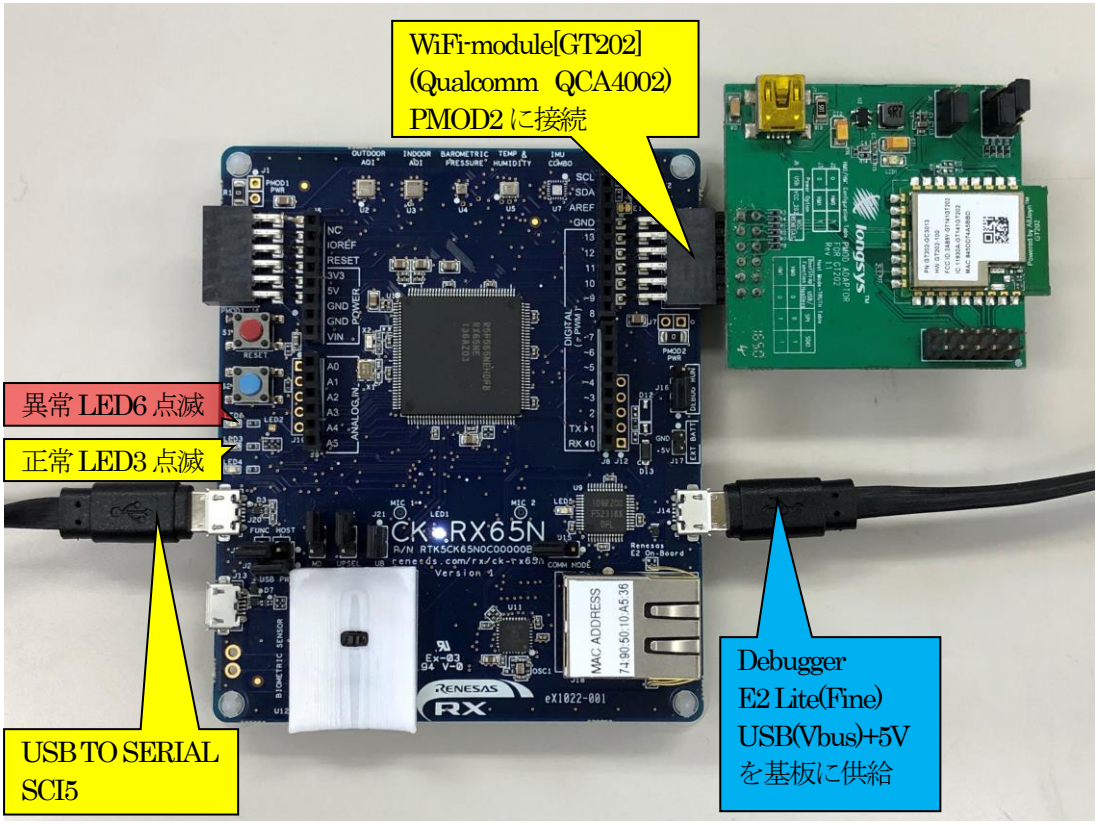


＜TCP_IP_AES_OpenSSLexe＞
【Plain mode】
1．平文を受信する。
2．受信した平文をそのまま送信する。

【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_tcp_aes_gt202_uselib＞
【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 m ode

５．「RX65N_gnu_dhcp_tcp_aes_gt202_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_wifi_CK>		
<rx_gnu_filex_lib>	AzureRTOS FileX ライブラリ作成用ホルダ	
<rx_gnu_netxduo_addons_lib>	AzureRTOS NetX Duo Addons 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_dhcp_tcp_aes_gt202_uselib>	DHCP / TCP 通信 /AES-CBC(暗号・復号) サンプルプロジェクト	
<HardwareDebug>	RX65N_gnu_dhcp_tcp_gt202_uselib.elf	ELF ファイル、JTAG で使用
	RX65N_gnu_dhcp_tcp_aes_gt202_uselib.map	MAP ファイル、アドレス情報
	RX65N_gnu_dhcp_tcp_aes_gt202_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)
	<thredx>	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 Duo Addons ライブラリ
	librx_gnu_netxduo_lib.a	NetX ライブラリ
	ibrx_gnu_threadx_lib.a	ThredX ライブラリ
<src>	<Azure_aes> 　|aes.c　|aes.h	AES-CBC 暗号・復号処理のソース
	<driver>	
	|<r_irq_rx>	FSP IRQ ドライバ（変更）
	|<r_sci_rx>	FSP SPI ドライバ（変更）
	<rtos_config>	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
	|tcp_aes_thread_entry.c	TCP スレッド処理のソース
	<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
		sample_netx_duo_ping.c	NetX 等初期化サンプルソース
		sf_wifi_nsal_api.c	NetX WiFi-API ソース
		sf_wifi_nsal_api.h	sf_wifi_nsal_api.c のヘッダ
		linker_script.ld	リンカスクリプトファイル
	RX65N_gnu_dhcp_tcp_aes_gt202_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
A_PRINTF_ENABLED	0	処理進行内容の表示を無効にする
	1	処理進行内容の表示を有効にする
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(
		&g_ip0,
		"NetX IP Instance 0",
		#if (NX_ENABLE_DHCP == 1)
		IP_ADDRESS(0,0,0,0),
		IP_ADDRESS(255,255,255,0),
		#else
		IP_ADDRESS(192,168,21,54), //固定 IP アドレス
		IP_ADDRESS(255,255,255,0), //サブネットマスク
		#endif
		&g_pool0,
		nsal_netx_driver,
		(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 CK-RX65N）

１）DHCP 無効時（NX_ENABLE_DHCP＝0）

＜DHCP FIXED Thread＞

Term 画面

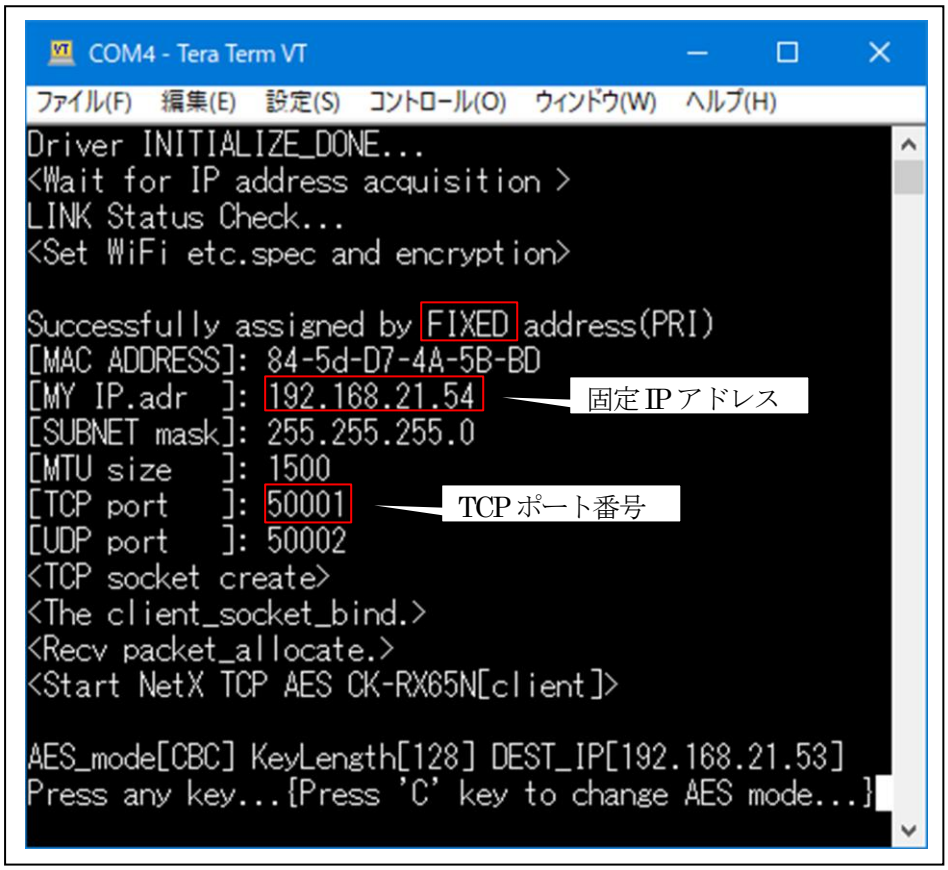
＜１＞「"Driver INITIALIZE_DONE..."」

＜２＞「"<Wait for IP address acquisition>.."」

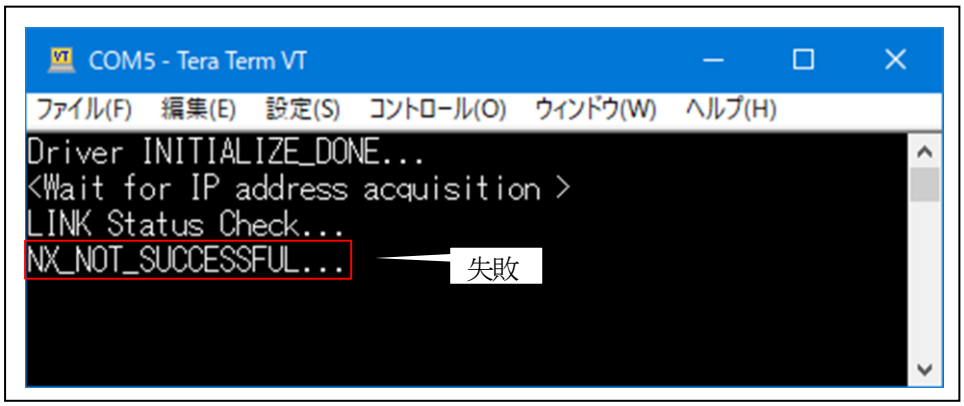
＜３＞「"LINK Status Check..."」

＜４＞「"<Set Wi Fi ect.spec and emcryption>"」

＜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



＜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



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>.."」

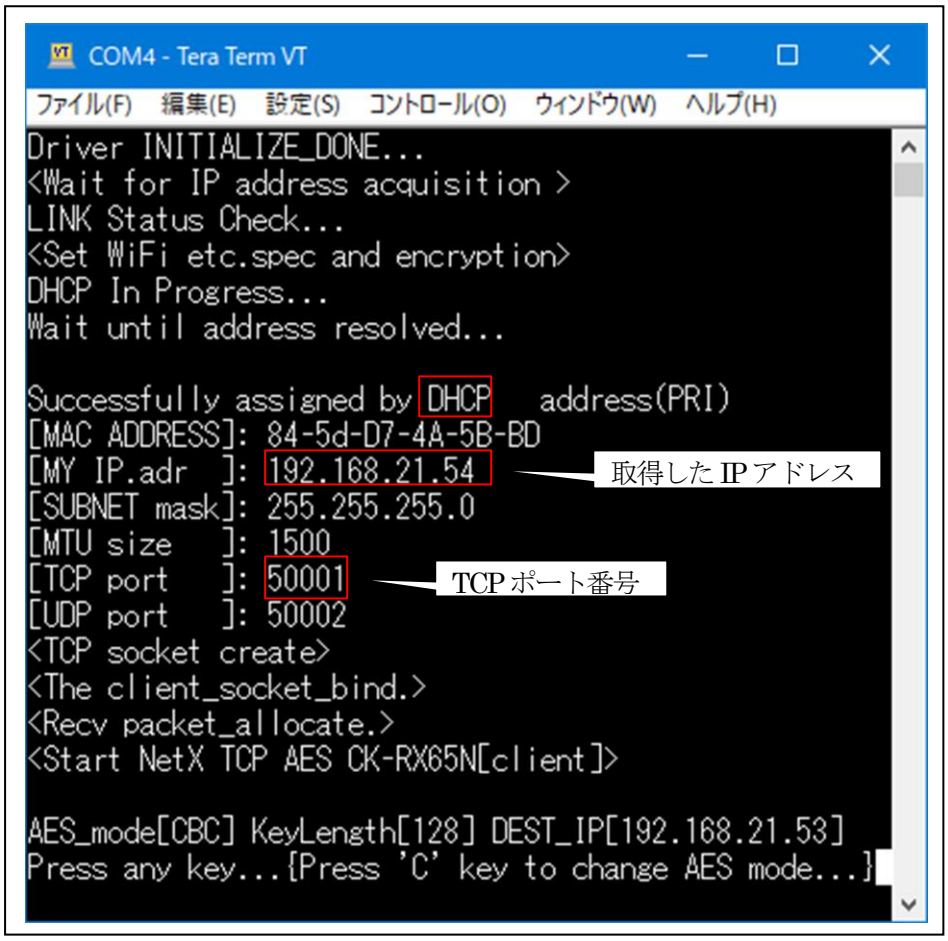
＜３＞「"LINK Status Check..."」

＜４＞「"<Set Wi Fi ect.spec and emcryption>"」

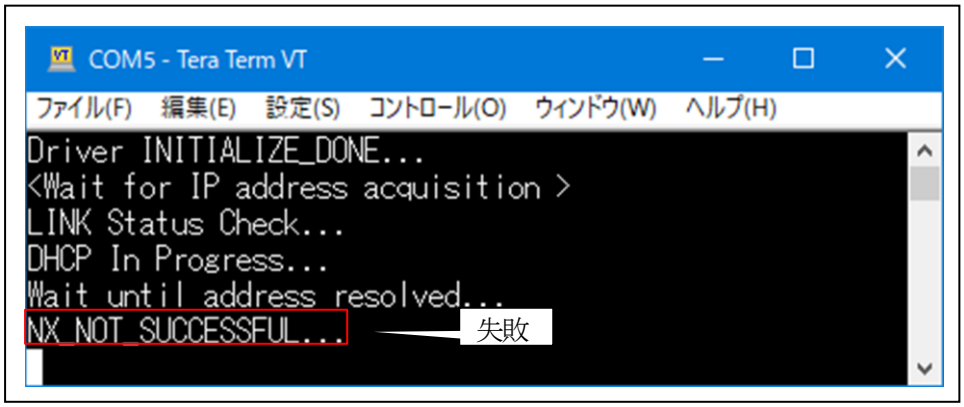
＜５＞「" DHCP In Progress..."」

＜６＞「" Wait until address resolved..."」

＜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

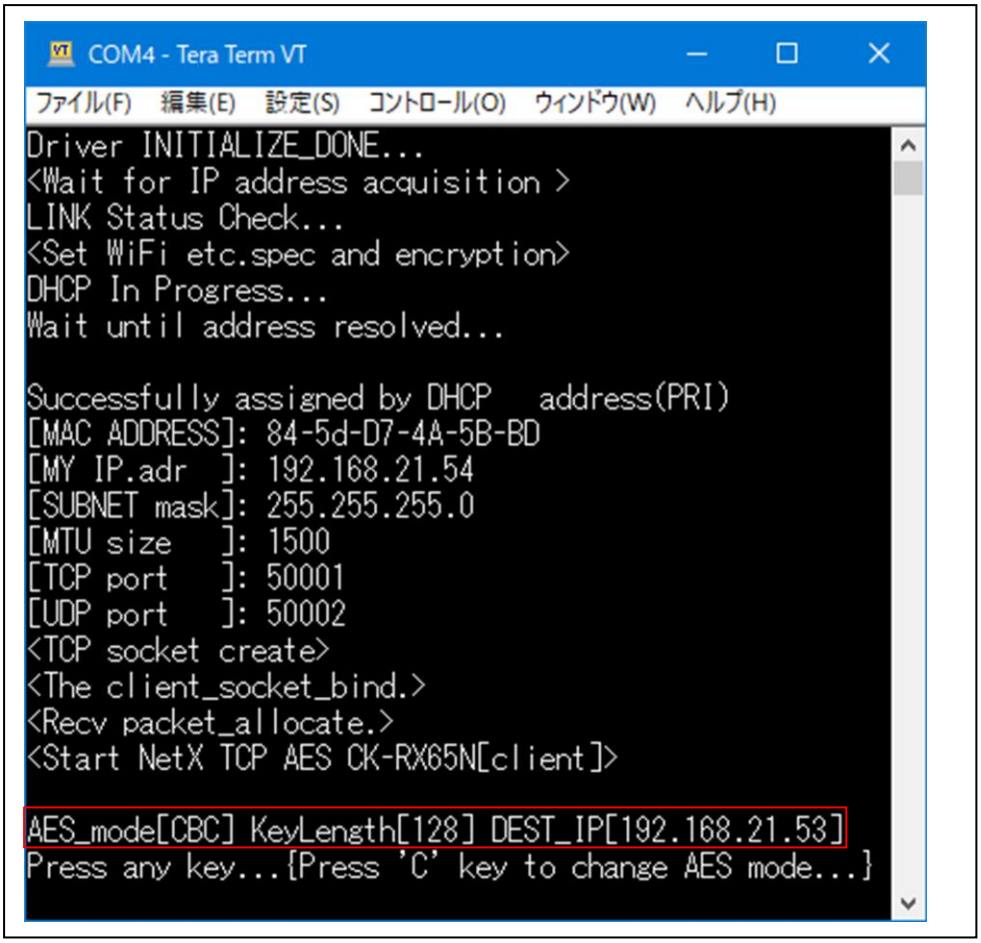


＜失敗画面＞IP アドレス未確立により、基板上の LED6（赤色）を 100msec 毎に点滅



3）TCP/IP 送受信

＜TCP AES Thread＞

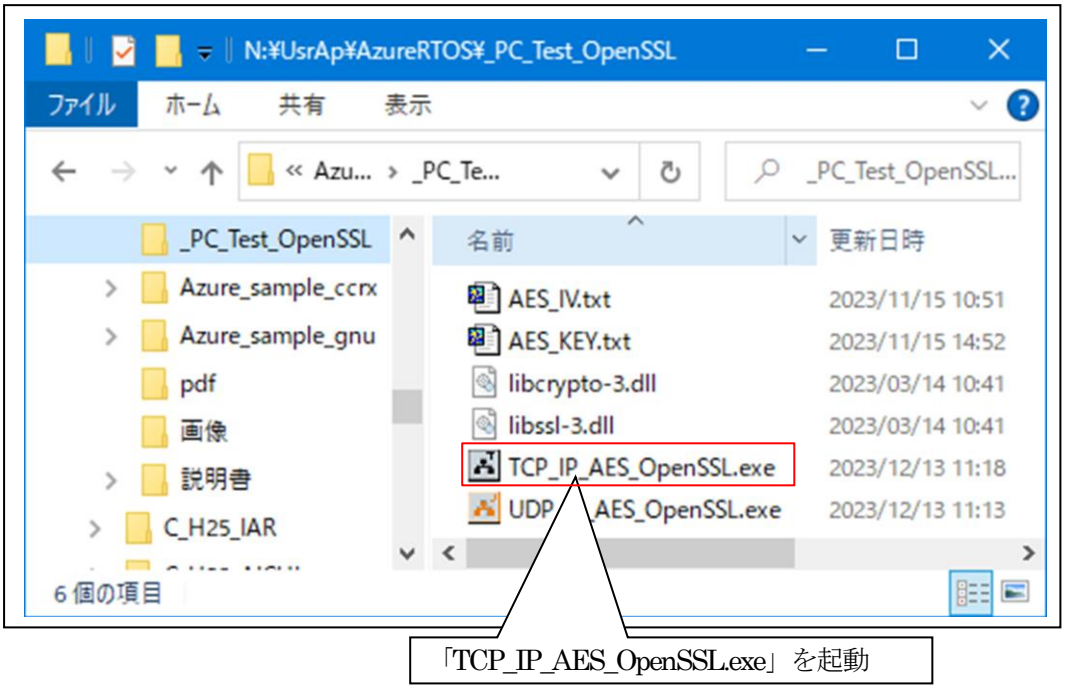


表示項目	表示内容	説明
AES_mode[x]	PLAIN CBC	送受信モードの指定 ◎変数のフラグにより指定 aes.c：int AES_crypto_mode; 0 ： PLAIN // 平文モード 1 ：CBC　//AES-CBC モード暗号・復号
KeyLength[x]	128 192 256	AES-CBC モード時の Key ビット時の指定 ◎変数の数値により指定 aes.c：int AES_crypto_bit; 128 ：Key ビット長が 128bit 192 ：Key ビット長が 192bit 256 ：Key ビット長が 256bit
DEST_IP[xxx.xxxx.xxxx.xxxx]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

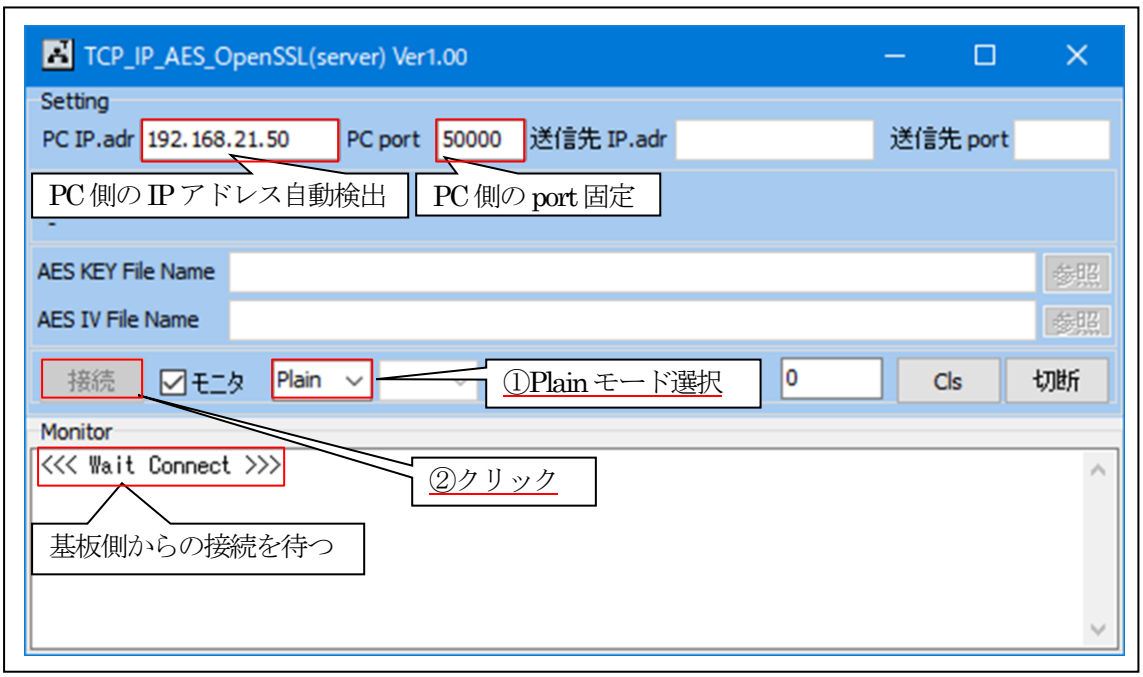
５－４．Windows PC 側のテストプログラムで動作確認（PLAIN（平文）モード）

１）「TCP_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

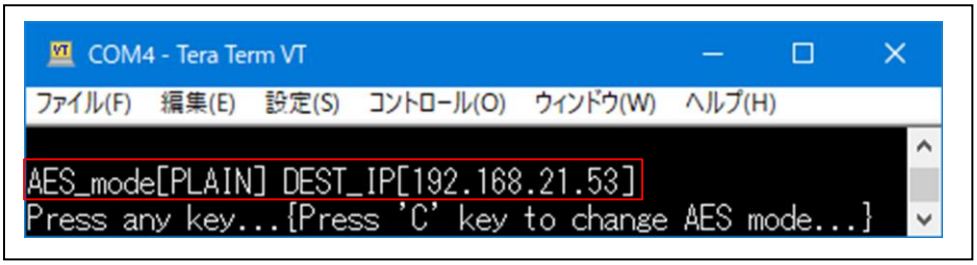


２）「TCP_IP_AES_OpenSSL」の各項目を設定して「基板」側からの「接続」を待つ。



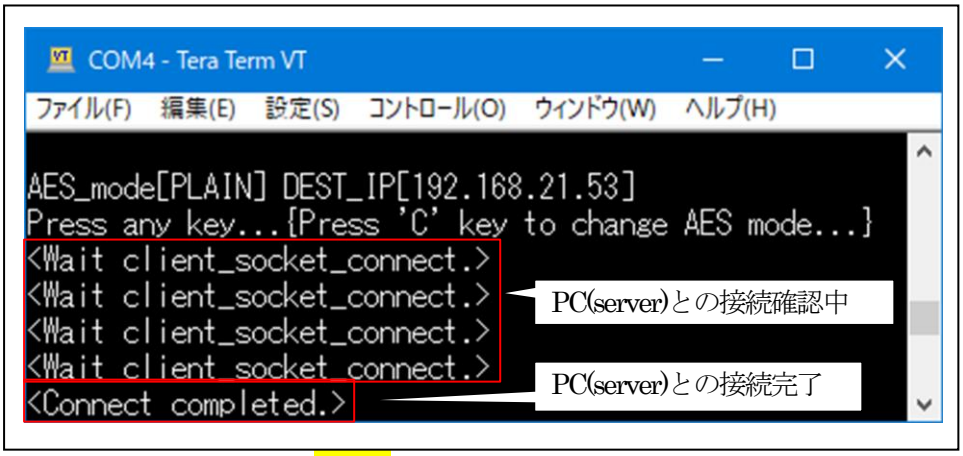
3）基板側の各項目の確認と設定。

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[192.168.21.53]	送信先(PC 側)IP アドレス ◎define にて指定 tcp_aes_thread_entry.c： #define DEST_IP IP_ADDRESS(192,168,21,53)



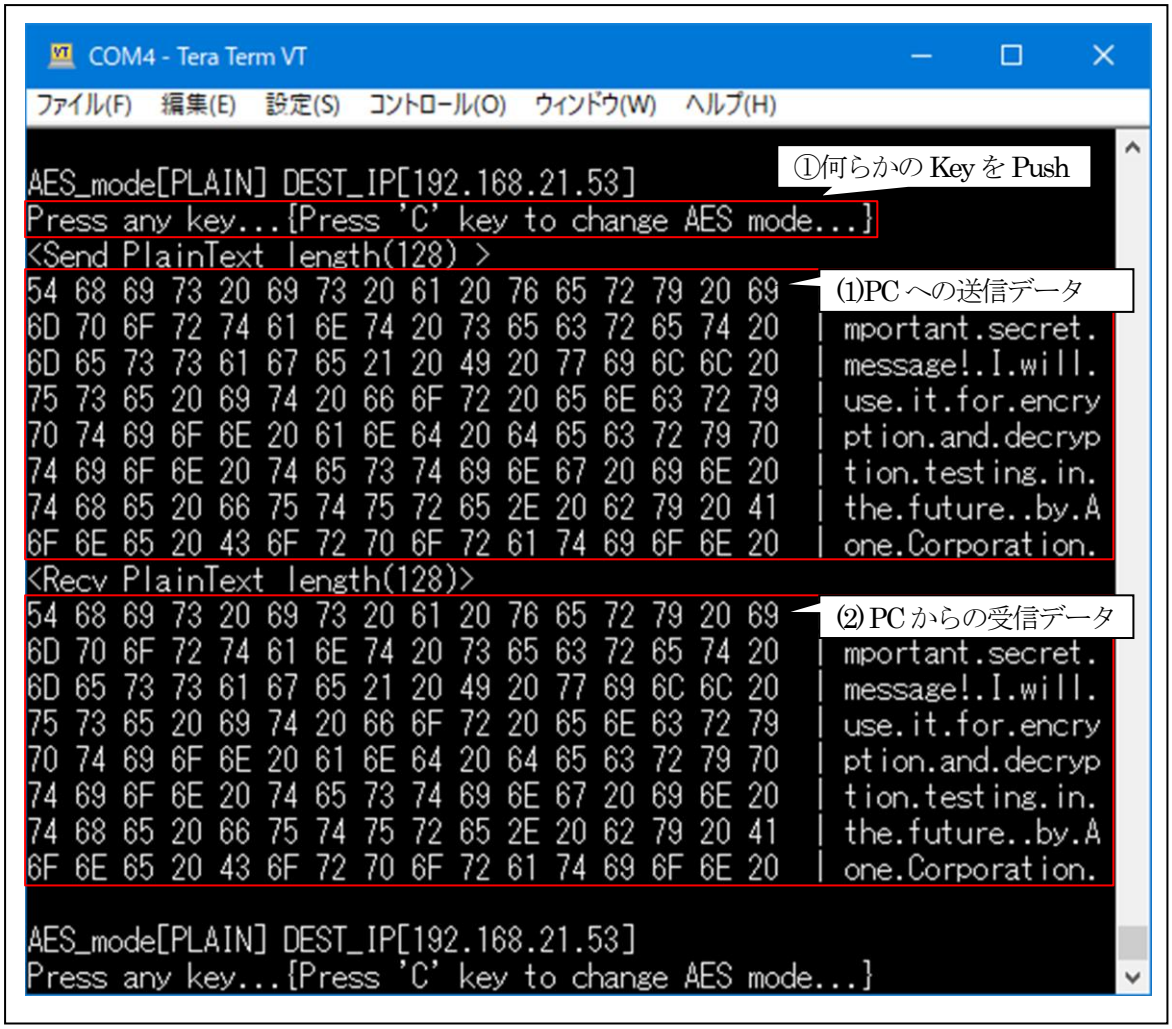
4）基板側から PC(server) 側へ平文テキストを送信する。

①disconection の場合は、PC(server)との Conection 処理を実行



☆接続が失敗した場合は「Ctrl+C」Key-Push で中断する。

②接続完了後、「基板」側から PC(server) 側へ平文テキストを送受信する

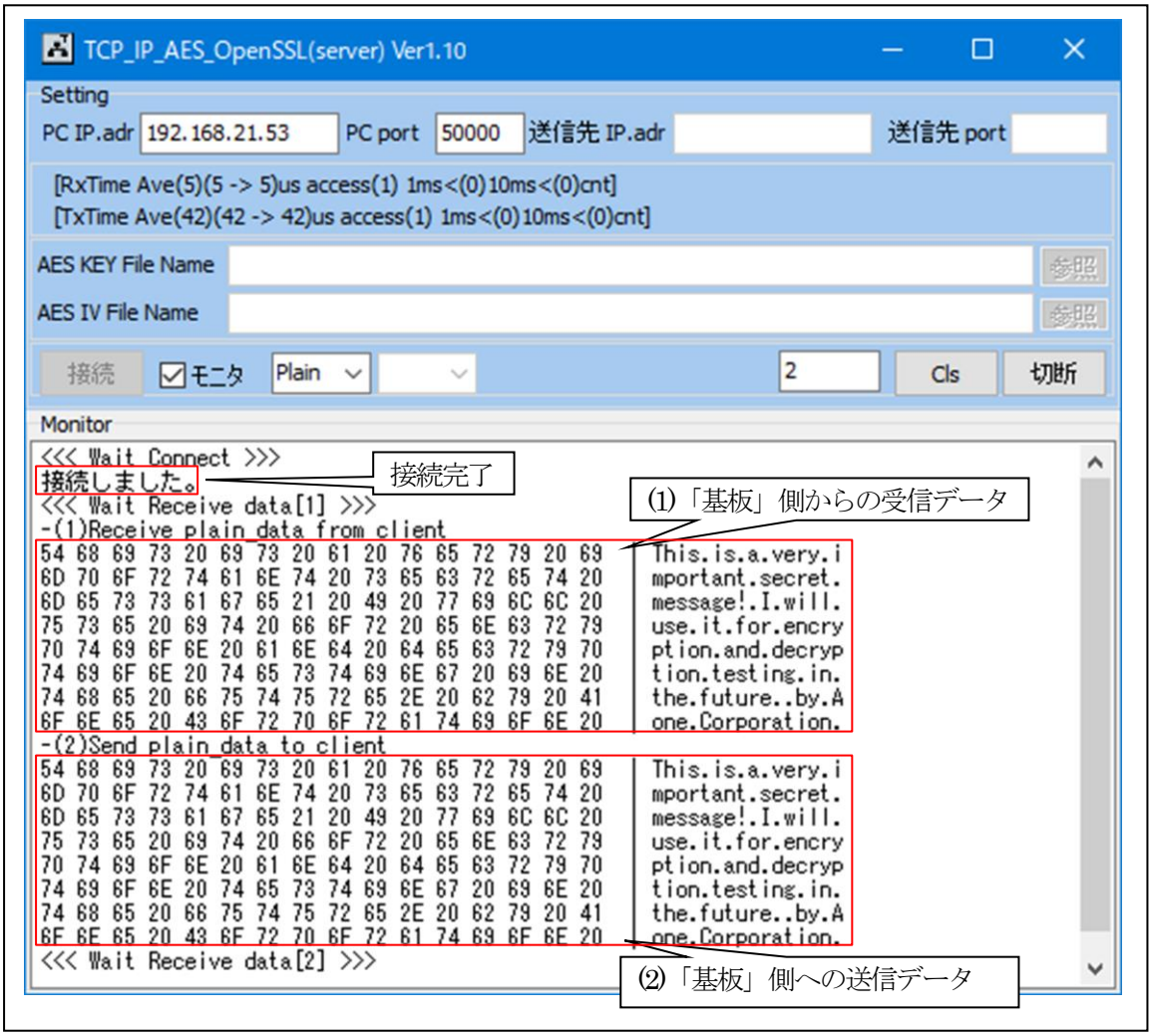


☆受信処理が失敗した場合は「Ctrl+C」Key-Push で中断する。

5）基板側の原文保存場所（各モード共通）

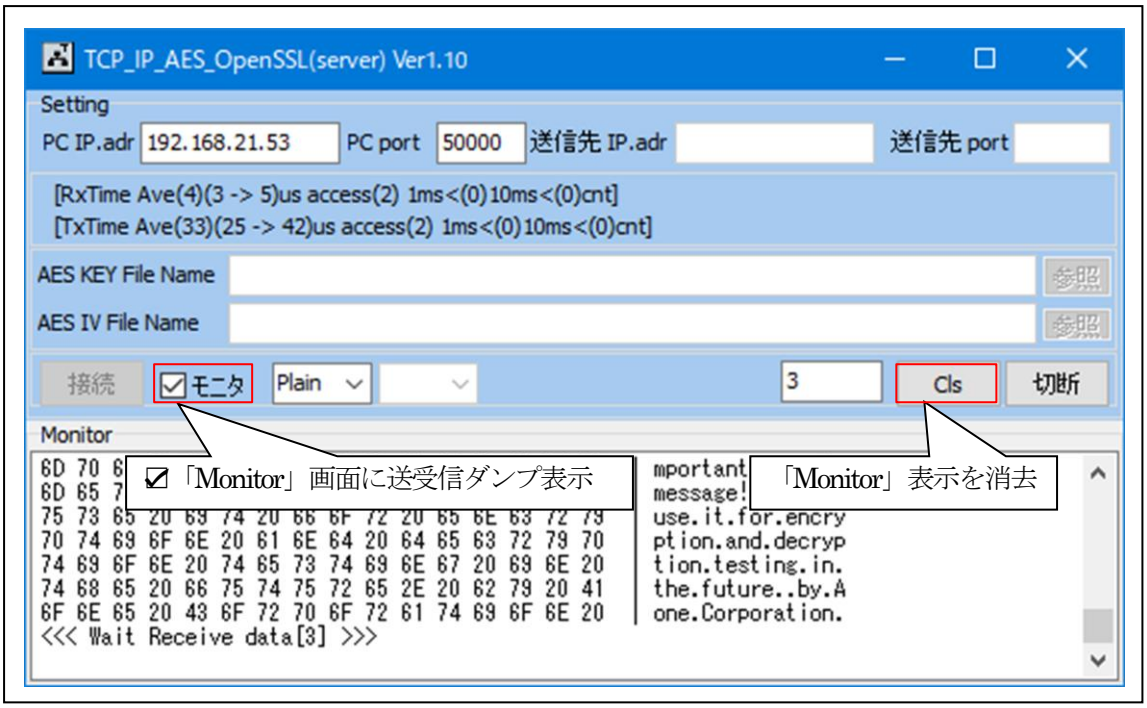
モジュール名	変数名
tcp_aes_thread_entry.c	static UCHAR *src_str = {　//テスト用原文 "This is a very important secret message! " "I will use it for encryption and decryption testing " "in the future. by Aone Corporation " };

6）「TCP_IP_AES_OpenSSL」側の送受信を確認する。

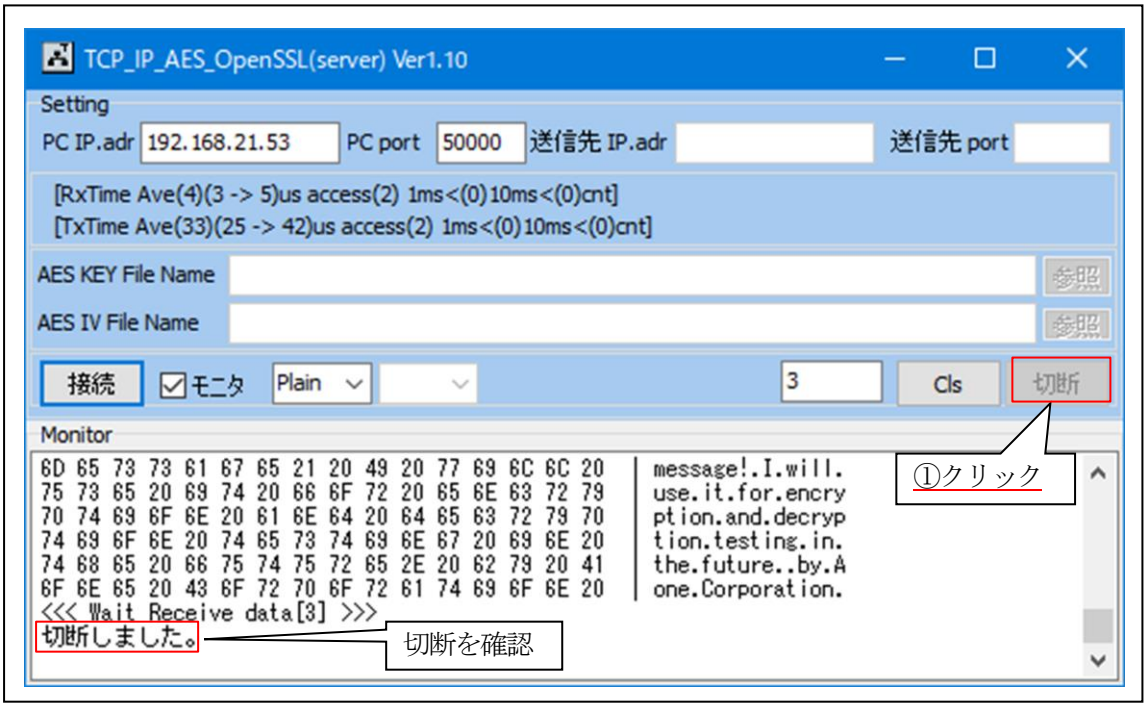


☆受信データをそのまま送信します。（ループバック）

7）「TCP_IP_AES_OpenSSL」その他の操作（各モード共通）



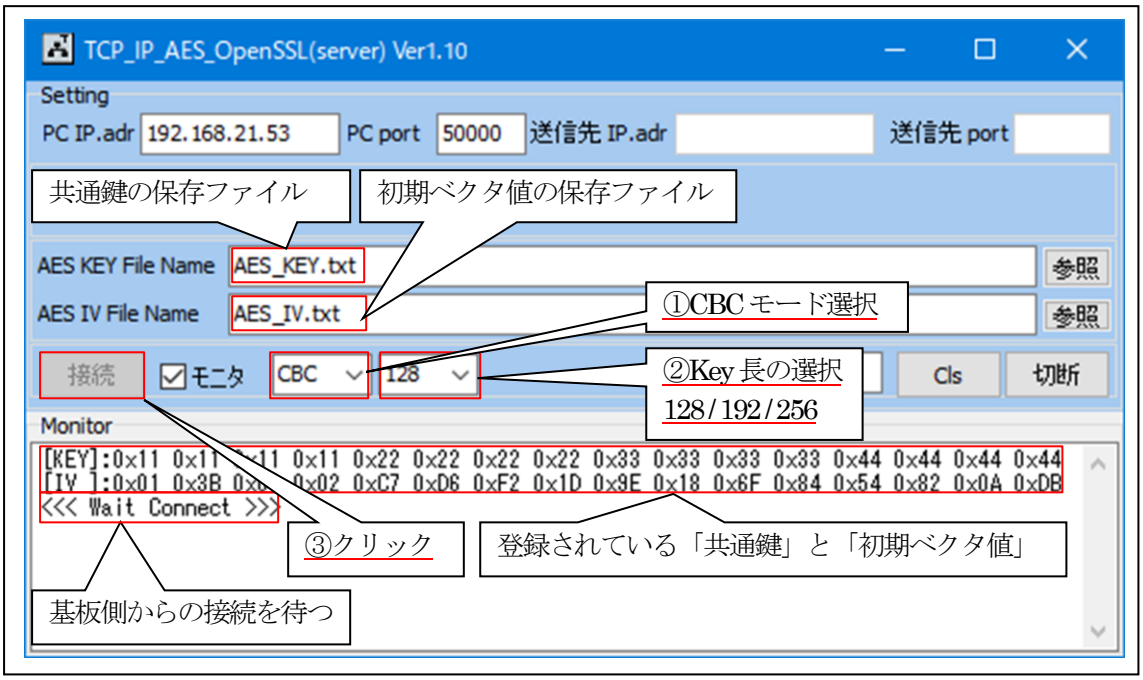
8）TCP_IP-Port を「切断」する。（各モード共通）



５－５．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.exe」を起動する。

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）「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、//のみの使用にして下さい。
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、//のみの使用にして下さい。
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。

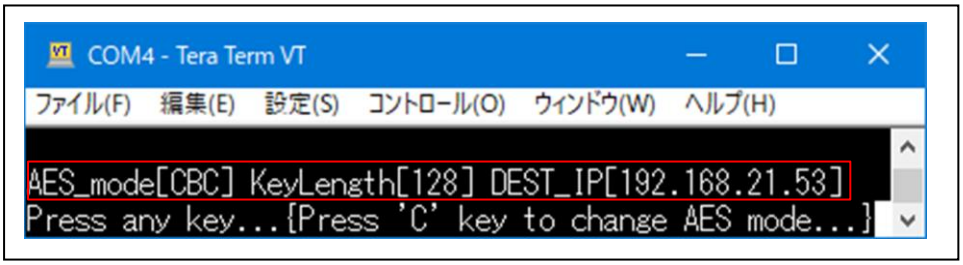
表示項目	説明
AES_mode[CBC]	送受信モードの指定 ◎変数のフラグにより指定 aes.c:：int AES_crypto_mode＝CBC; // 1=CBC
KeyLength[128]	AES-CBC モード時の Key ビット時の指定 ◎変数の数値により指定 aes.c:：int AES_crypto_bit＝128;
DEST_IP[192.168.21.53]	送信先(PC 側)IP アドレス ◎define にて指定 tcp_aes_thread_entry.c： #define DEST_IP IP_ADDRESS(192,168,21,53)

共通鍵データの保存モジュールと変数【aes.c】

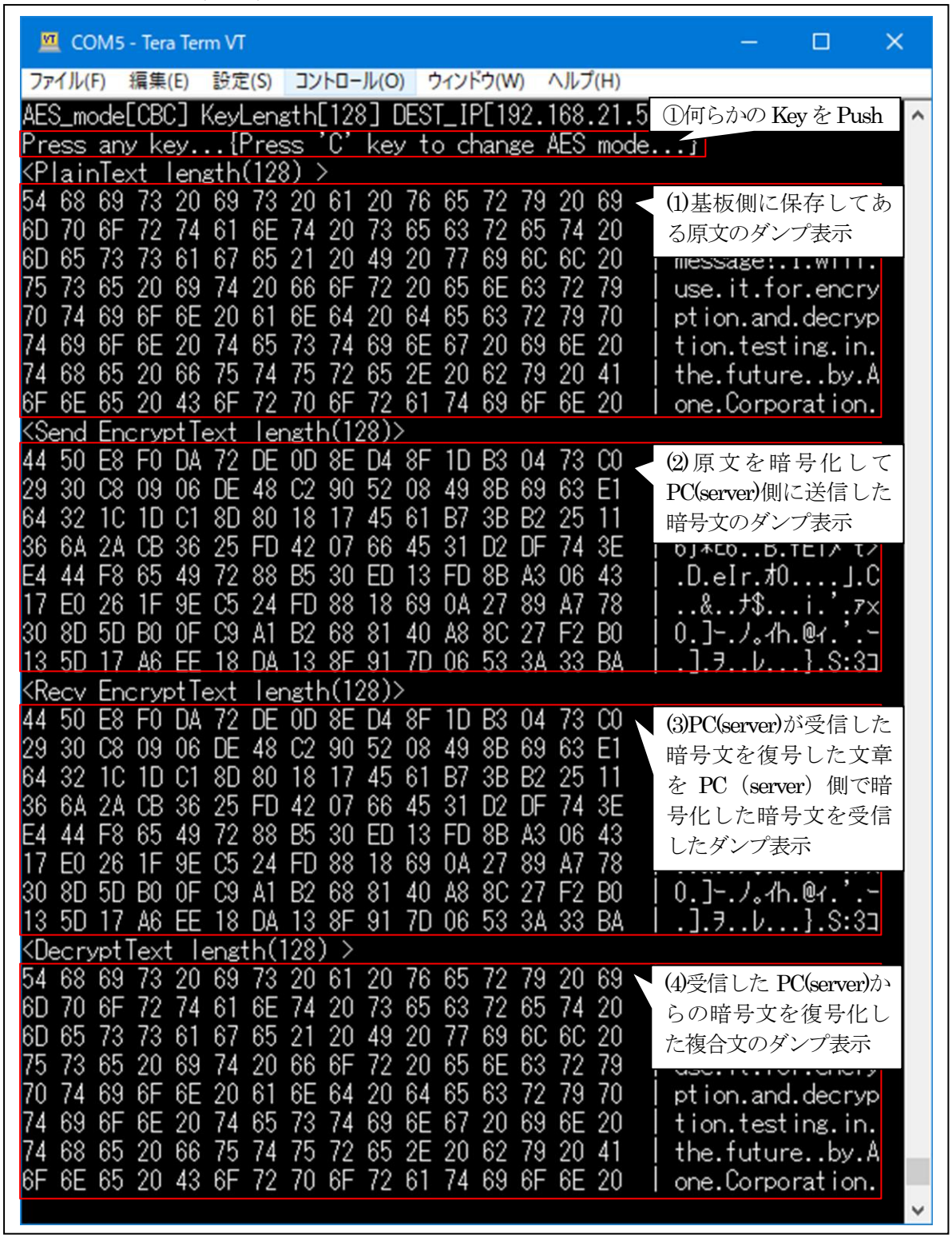
uint8_t AES_key[32] ={ // default aes_common_key 共通鍵 128bit｜192bit｜256bit
0x11,0x11,0x11,0x11,0x22,0x22,0x22,0x22,0x33,0x33,0x33,0x33,0x44,0x44,0x44,0x44, // 128bit
0x55,0x55,0x55,0x55,0x66,0x66,0x66,0x66, // ↑＋192bit
0x77,0x77,0x77,0x77,0x88,0x88,0x88,0x88, // ↑＋256bit
};

初期ベクタ値データの保存モジュールと変数【aes.c】

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
};



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

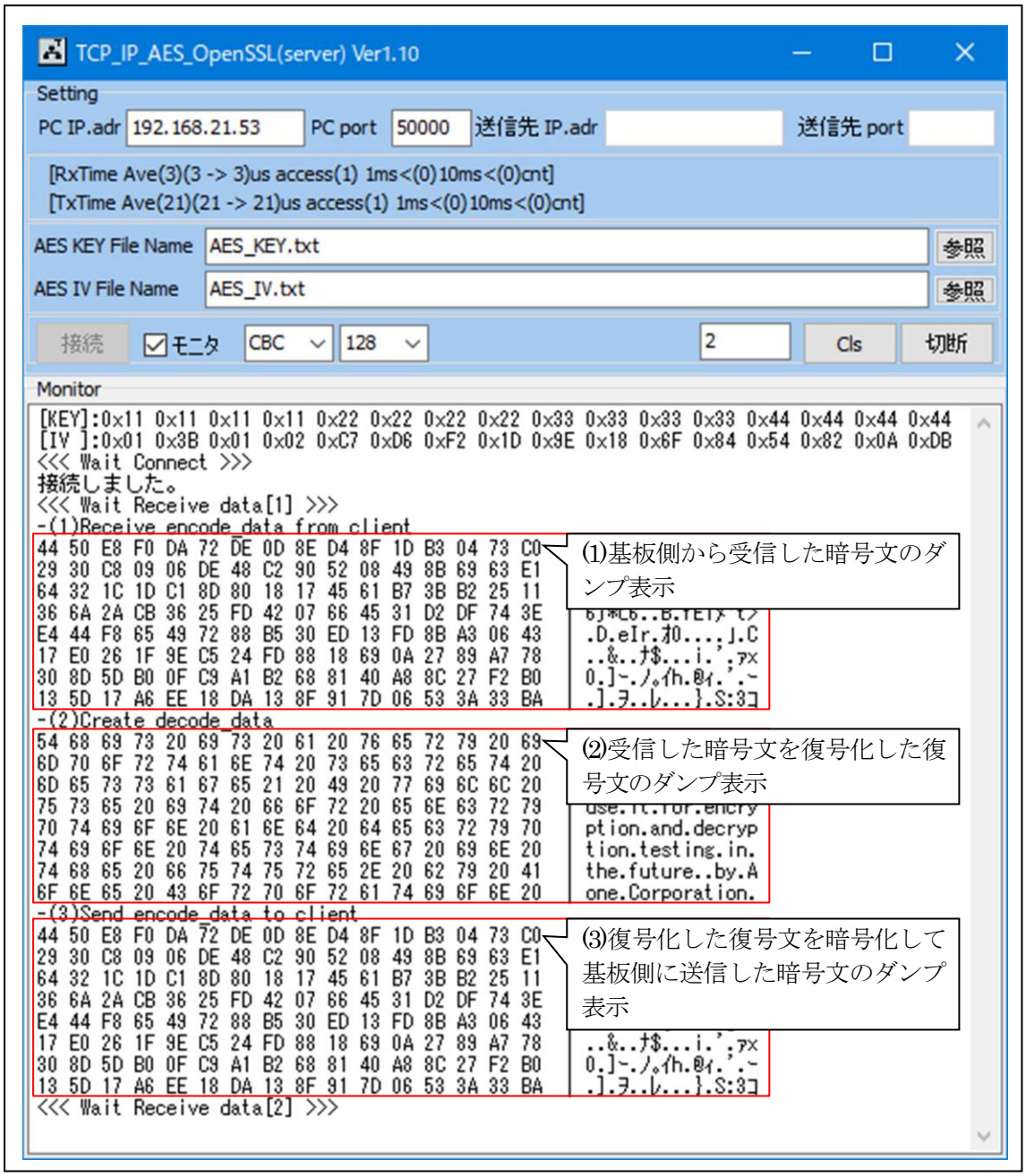


☆受信処理が失敗した場合は「Ctrl+C」Key-Push で中断する。

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 EncrptText」と「(3)Recv EncrptText」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）「TCP_IP_AES_OpenSSL」側の送受信を確認する。



①「⑴Receive encode_data from client」と「⑶Send 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「⑵Create decode_data」と「基板」側の「⑷DecryptText 」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は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
・「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.jp