Rev 1.10.00

Renesas RX63N 用サンプル(RTOS-NORTi 版)の説明

(EV-RX-xx＋MP-RX63N/FPGA|CPLD-xx 対応)

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，いかなる保証も行わない。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（RTOS 版）のプロジェクト名

サンプルプロジェクト名		
EVRX_NORTi	MCU 基板(MP-RX63N/FPGA|CPLD-*) 評価用基板(EV-RX-*)用サンプル	ソース公開 （有償）
EVRX_NORTi_USB	MCU 基板(MP-RX63N/FPGA|CPLD-*) 評価用基板(EV-RX-*) USB-Function 機能を追加したサンプル	実行ファイルのみ添付

統合開発環境	コンパイラー
HEW4(バージョン 4.09.01.007)	RXC(バージョン V.1.02.01.000）
CS+ for CC V3.00.00	CC-RX(バージョン V.2.02.00)

HEW4 の環境をベースにして、CS+は全てサブプロジェクトとしてソースおよびヘッダファイル全てを HEW4 のフォルダから登録しています。
最終生成物(*.mot/*.abs)は、Hew4(Debug)/CS+(DefaultBuild)に作成されます。

C ソースに#ifdef 等のマクロ定義している場合に使用します。	
RTOS	NORTi 使用時に定義
RXC_V1	NORTi 使用時に定義
ITF_LIB	USB-Function 使用時に定義
RXC	USB-Function 使用時に定義

サンプルプロジェクト別に必要なマクロ定義例				
EVRX_NORTi	RTOS	RXC_V1		
EVRX_NORTi_USB	RTOS	RXC_V1	ITF_LIB	RXC

２－１．「EVRX_NORTi」プロジェクトの説明

１）動作説明

・Tera Term からのコマンド指示により各デバイスを動作させる。

・各コマンド体系は後記にて説明します。

２）フォルダ構成とファイル名

Sample_RX¥EV-RX¥Hew4¥EVRX_NORTi				【Hew4】
	Debug	ビルドにより生成された実行ファイル等が格納される場所		
	NORTi	空		
		ReadMe.txt	インストール済み NORTi オリジナルからのインポート手順書	
	NORTi_smp	NETSMP	inc	NETSMP のインクルード
			aonettcp.c	TCP/IP ループバックテスト
			noncons.c	コンソール関係
			nondhcp.c	DHCP 処理
			nonedns.c	DNS サーバー
			nonping.c	PING 処理
			nonshel.c	コマンド shell
			nonsntp.c	SNTP 処理
			nontelc.c	TELNET サンプル
			nonteld.c	TELNET デーモン
			sntptime.c	SNTP 処理
			ReadMe.txt	オリジナル NETSMP の変更履歴
		SMP	inc	SMP のインクルード
			r4i63n.c	タイマーと割込みマネジャー
			nonecfg.c	TCP/IP サンプルコンフィグレーション
			nonethw.c	ETHER 割り込み登録
			nos63n.c	SIO 用シリアルドライバー
			phy63n.c	PHY コントロール
			rx63ne.c	LAN ドライバーの初期化
			ReadMe.txt	オリジナル SMP の変更履歴
	src_app	inc	src_app のインクルード用ディレクトリ	
		main_r.c	メイン処理	
		board.c	LED・SW 等の処理ソフト	
		bsc.c	BSC 初期化処理	
		cevrx63n.c	OS からの周辺 I/O の初期化	
		rtc.c	RTC の初期化と処理ソフト	
		sfram.c	FRAM の初期化と read/write 処理	
		lvc.c	LVC の初期化と LVC 電源断検出のサンプル	
		command.c	コマンド処理	
	src_eva	inc	src_eva のインクルード用ディレクトリ	
		e2p.c	EEPROM の read/write 処理	
		riic.c	RIIC の初期化と read/write 処理	
		rcan.c	RCAN の初期化と read/write 処理	
		sci0.c	SCI0 の初期化と read/write 処理	
		usb_func.c	ITF_USBLib の使用サンプル	

	src_evb	inc	src_evb のインクルード用ディレクトリ
		c_lcd_fpga.c	キャラクタ LCD 表示処理
		m_lcd_fpga.c	モノクロ LCD 表示処理
		pwm1.c	DC モータ(PWM 出力)制御処理
	src_sys	inc	src_sys のインクルード用ディレクトリ
		cpg.c	CPG 設定の基本ソフト
		idcode.c	IDCODE 管理テーブル
		initrx.c	ルネサス C 初期設定ルーチン
		ofs.c	OFS 管理テーブル
		profile.c	プロファイル処理
		sbrk.c	メモリアロケーション
		vect63n.c	リセットベクターテーブル

Sample_RX¥EV-RX¥Cube¥ EVRX_NORTi			【CS+】
	DefaultBuild	ビルドにより生成された実行ファイル等が格納される場所	
	lnk_NORTi	空	Hew4¥ EVRX_NORTi¥NORTi にリンク
	lnk_NORTi_smp	空	Hew4¥EVRX_NORTi¥NORTi_smp にリンク
	lnk_app	空	Hew4¥ EVRX_NORTi ¥src_app にリンク
	lnk_eva	空	Hew4¥ EVRX_NORTi ¥src_eva にリンク
	lnk_evb	空	Hew4¥ EVRX_NORTi ¥src_evb にリンク
	lnk_sys	空	Hew4¥ EVRX_NORTi ¥src_sys にリンク

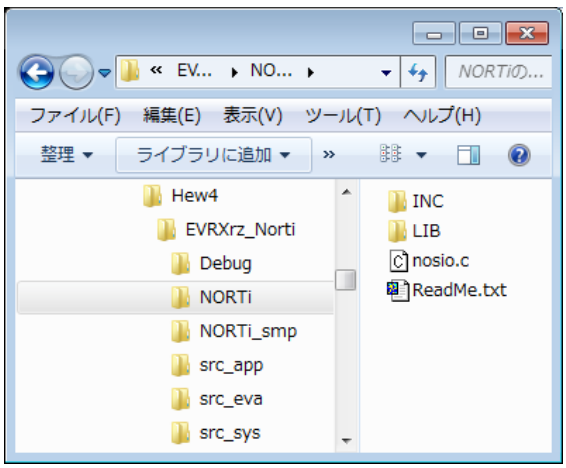
3）インストール済み NORTi オリジナル（有償）からサンプルにインポートする手順

a．サンプル「EVRX_NORTi」を PC 機の適当なフォルダに全 Copy します。または、「Sample_RX.zip」の圧縮ファイルを適当なフォルダで解凍します。

b．2）表の黄色部は、空ディレクトリになっていますので、インストール済み NORTi オリジナルから必要なファイルを Copy します。

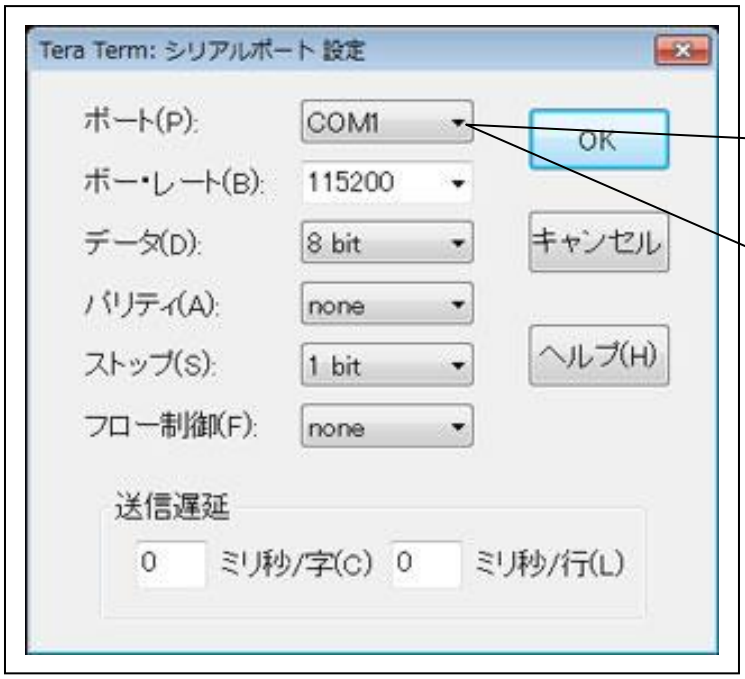
NORTi オリジナル		サンプル(EVRX_NORT
NORTi¥INC	→	NORTi¥INC
NORTi¥LIB	→	NORTi¥LIB
NORTi¥SRC¥nosio.c	→	NORTi¥nosio.c

上記のように NORTi オリジナルから、サンプル「EVRX_NORTi¥NORTi」の空ディレクトリに Copy して下さい。



c．ReadMe.txt に Copy 後の変更内容が記述してありますのでソース変更をして下さい。

4）コマンド実行を指示するため「TeraTerm Pro」をインストールする。
①「teraterm-4.80.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

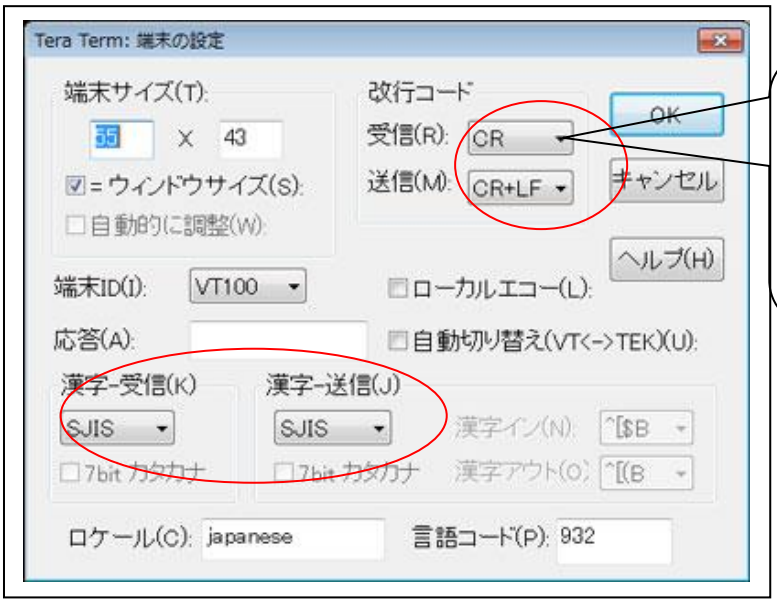


COM 番号は、PC 側でシリアル通信可能な番号を指定する。

115200BPS
8bit
none
1bit
none

の仕様にする。

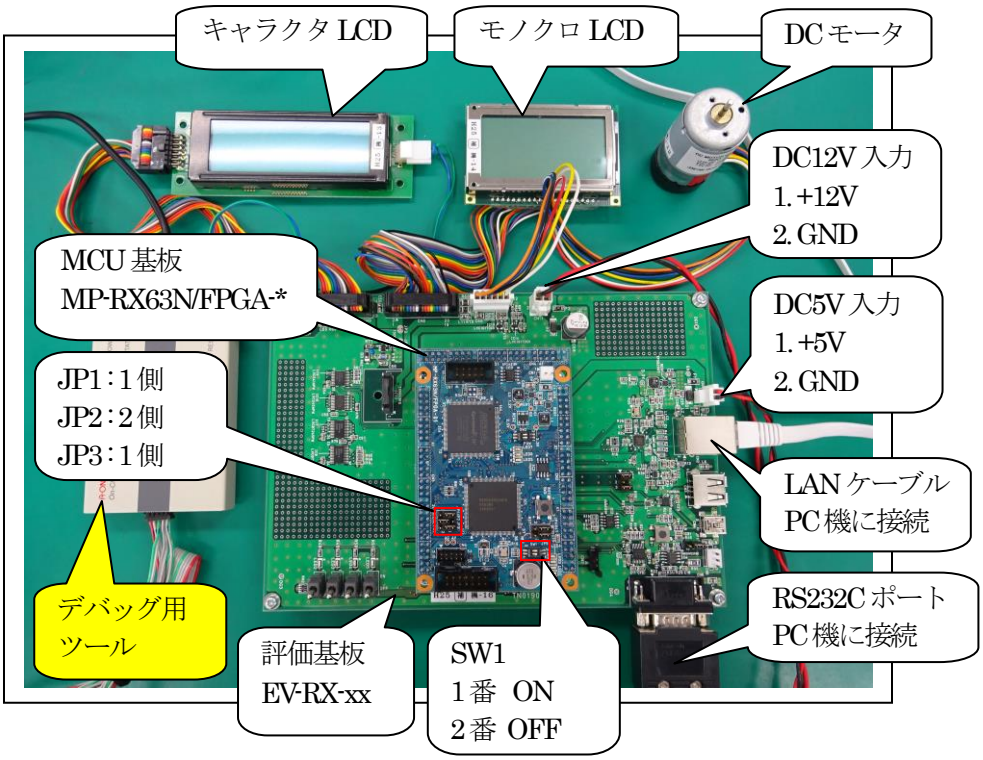
④端末の設定



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。

赤丸の設定にする。

5）動作構成



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　ICUSB232FTN」を推奨
③自作する場合は、下記の配線になります。

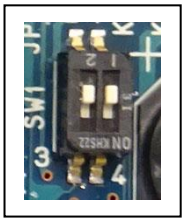
EV-RX-xx

DSUB 9pin オス	
1	NC
2	RXD
3	TXD
4	NC
5	GND
6	NC
7	RTS
8	CTS
9	NC

PC-Windows

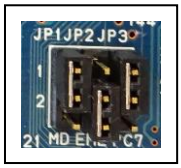
DSUB 9pin オス	
1	NC
2	RXD
3	TXD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	NC

④MCU 基板上の SW1 設定



SW1-1　ON
SW1-2　OFF

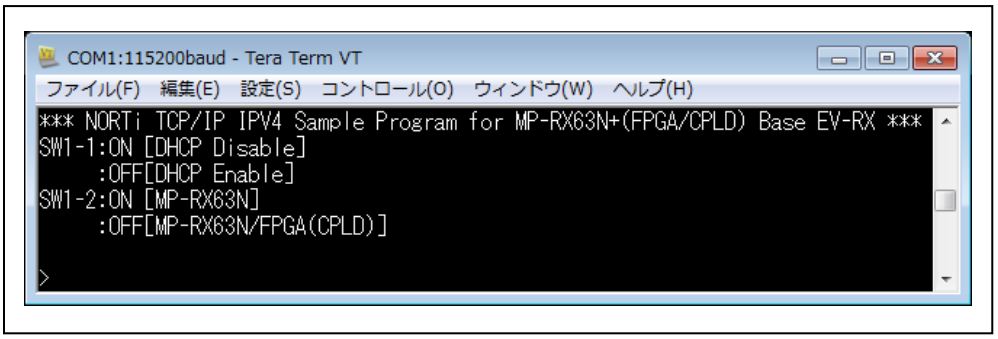
⑤MCU 基板上の JP 設定



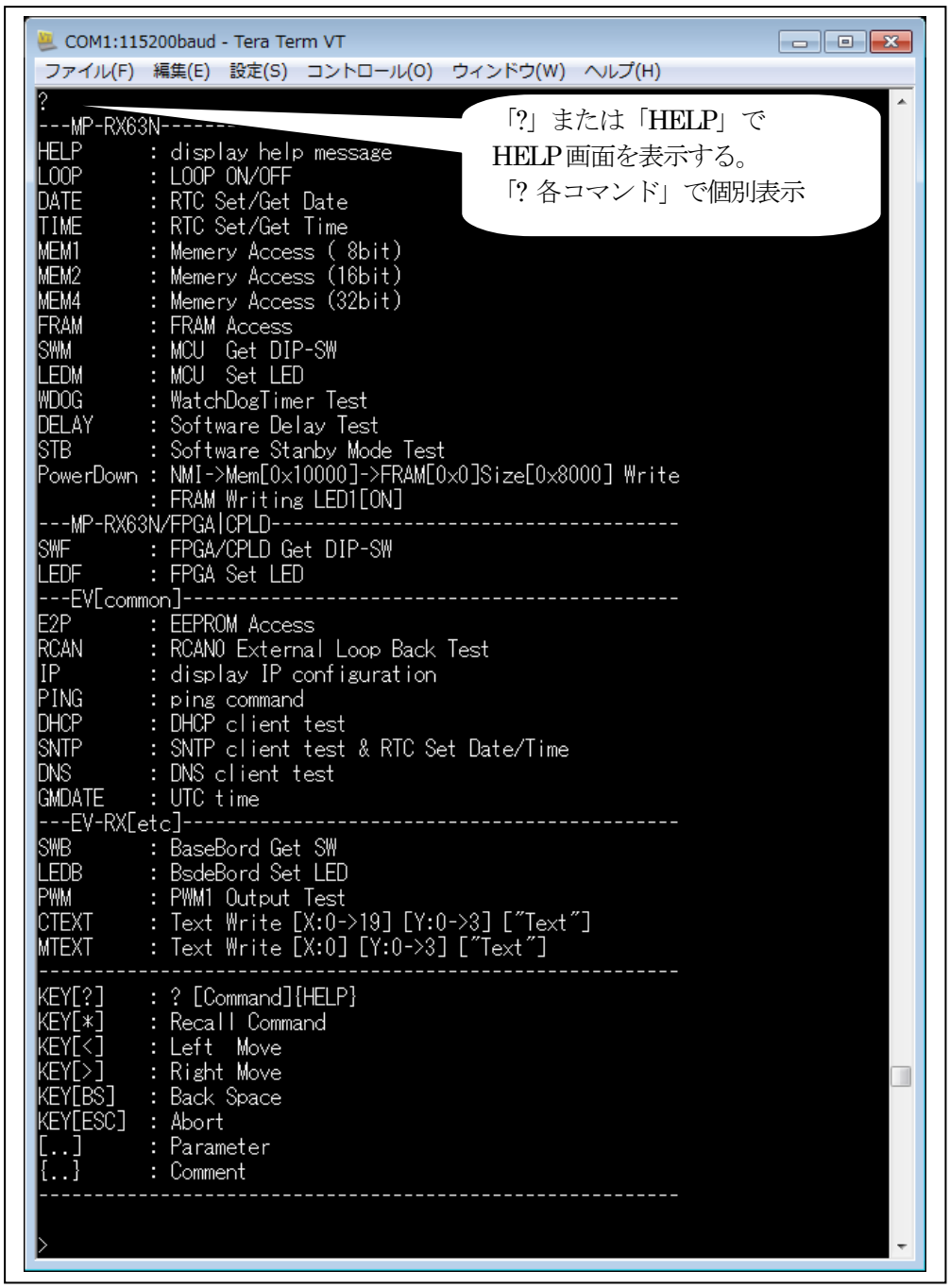
JP1：1 側（シングルチップ）
JP2：2 側（オンチップエミュレータ）
JP3：1 側（PC7 端子：LOW）

SW1-1-ON　起動時に DHCP サーバーへのアクセスを禁止する。
SW1-2-OFF　MP-RX63N/FPGA | CPLD-xx 用サンプルを指定

6）「EVRX_NORTi」プロジェクトのプログラムを MCU 基板にダウンロードして実行させます。



TeraTerm pro の画面にオープニングメッセージが表示されます。



２－１－１．各コマンドの説明

LOOP　コマンド

各コマンドを繰り返し実行させたい時に使用します。

LOOP ␣1⏎　//LOOP 指示　　␣記述はスペース表現とします。以下省略
LOOP ␣0⏎　//LOOP 解除　　⏎記述はリターン表現とします。以下省略

LOOP　1　にてコマンド処理を繰り返し実行している時に「ESC」キー入力で中断します。

DATE　コマンド

MCU 内蔵の RTC に年月日曜を設定します。

DATE ␣年␣月␣日␣曜日⏎　// DATE␣2015␣4␣5␣0　2015/4/5 日曜日
　　//曜日　0:日　1:月　2:火　3:水　4:木　5:金　6:土
DATE⏎　　// 現設定データを表示

TIME　コマンド

MCU 内蔵の RTC に時間を設定します。

TIME␣時␣分␣秒⏎　　// TIME␣9␣0␣0　　9 時 0 分 0 秒
TIME⏎　　//現設定データを表示

MEM1　コマンド

メモリーを 8bit アクセスで Read/Write/FILL/インクリメント FILL します。

MEM2　コマンド

メモリーを 16nit アクセスで Read/Write/FILL/インクリメント FILL します。

MEM4　コマンド

メモリーを 32bit アクセスで Read/Write/FILL/インクリメント FILL します。

MEM{1/2/4}␣{R/F/I/W}␣先頭アドレス␣サイズ␣{パターン}⏎

{READ}
MEM1␣R␣0x1_0000␣0x100⏎　　// 0x1_0000 から 0x100 要素分 8bit ダンプ
MEM2␣R␣0x1_0000␣0x100⏎　　// 0x1_0000 から 0x100 要素分 16bit ダンプ
MEM4␣R␣0x1_0000␣0x100⏎　　// 0x1_0000 から 0x100 要素分 32bit ダンプ
{FILL}
MEM1␣F␣0x1_0000␣0x100␣0⏎　　// 0x1_0000 から 0x100 要素分(0)8bitFILL
MEM2␣F␣0x1_0000␣0x100␣0⏎　　// 0x1_0000 から 0x100 要素分(0)16bitFILL
MEM4␣F␣0x1_0000␣0x100␣0⏎　　// 0x1_0000 から 0x100 要素分(0)32bitFILL
{Increment FILL}
MEM1␣I␣0x1_0000␣0x100␣0⏎　　// 0x1_0000 から 0x100 要素分(0++)8bitFILL
MEM2␣I␣0x1_0000␣0x100␣0⏎　　// 0x1_0000 から 0x100 要素分(0++)16bitFILL
MEM4␣I␣0x1_0000␣0x100␣0⏎　　// 0x1_0000 から 0x100 要素分(0++)32bitFILL
{WRITE}
MEM1␣W␣0x1_0000␣0x12⏎　　// 0x1_0000 に 0x12 を Write
MEM2␣W␣0x1_0000␣0x1234⏎　　// 0x1_0000 に 0x1234 を Write
MEM4␣W␣0x1_0000␣012345678⏎　　// 0x1_0000 に 0x12345678 を Write

{Read Only Memory アドレス}
・内蔵 RAM エリア　　{0x0 ～0xFFFF}

{Read/Write Memory アドレス}
・MCU 内蔵 RAM エリア　　{0x1_0000 ～0x1_FFFF}
・FPGA 側 I/O エリア　　{0x700_0000 ～0x700_00FF}
・FPGA 内蔵 RAM エリア　　{0x600_0000 ～0x600_3FFF}
・MCU 内蔵周辺モジュール　　{周辺モジュールの仕様による}

FRAM　コマンド

FRAM の内容を内蔵メモリーに Read します。また、内蔵 RAM の内容を FRAM に Write します。

{READ}
FRAM␣R␣FRAM アドレス␣Store アドレス␣サイズ
ex)
FRAM␣R␣0x0␣0x1_0000␣0x8000
FRAM アドレス(0x0)からサイズ(0x8000)分 Store アドレス(0x1_0000)に Read します。

{WRITE}
FRAM␣W␣FRAM アドレス␣Memory アドレス␣サイズ
ex)
FRAM␣W␣0x0␣0x1_0000␣0x8000
FRAM アドレス(0x0)に Memory アドレス(0x1_0000)からサイズ(0x8000)分 Write します。

・FRAM アドレス　　{0x0 ～0x7FFF}
・Store アドレス　　{0x1_0000 ～0x1_FFFF}
・Memory アドレス　　{0x0 ～0x1_FFFF}

SWM　コマンド

MCU 側が制御している DIP-SW1 の状態を表示します。

SWM⏎

ex)
MCU DIP-SW1_1[ON/OFF] SW1_2[ON/OFF]

LEDM　コマンド

MCU 側で制御している LED1/2/3 を点灯・消灯します。

LEDM␣{0/1}␣{0/1}␣{0/1}⏎　　// LEDM {LED1} {LED2} {LED3} 0:消灯　1:点灯

WDOG　コマンド
WDOG タイマーを起動させ MCU リセットさせます。
MCU リセット後は、電源を再立ち上げして下さい。

DELAY　コマンド
MCU 内部で利用している 1usec タイマーの精度を計るため LED1 を点滅させます。

DELAY␣{Time 値}usec⏎　　// DELAY 10⏎　10usec の精度

①LED1{time 値}　点灯
②LED1{time 値}　消灯
③LED1{time 値}　点灯
④LED1{10msec}　消灯

STB　コマンド
ソフトウェア・スタンバイ・モードに移行させます。
STB 後は、電源を再立ち上げして下さい。

Power Down(NMI 処理)
停電検出回路が有効になっている場合、電源 OFF 時に内蔵 RAM の内容を 32Kbyte 分 FRAM に Write します。

LED1 点灯
FRAM(0x0)から内蔵 RAM(0x1_0000)の内容を 32Kbyte 分 Write する。
LED1 消灯

LED1 の点灯時間を計測することにより書き込み時間を得ることができます。

SWF　コマンド
FPGA 側が制御している DIP-SW2 の状態を表示します。

SWF⏎

ex)
FPGA/CPLD DIP-SW2_1[ON/OFF] SW2_2[ON/OFF]

LEDF　コマンド
FPGA 側で制御している LED5/6 を点灯・消灯します。

LEDF␣{0/1}␣{0/1}⏎　　// LEDF {LED5}{LED6}　0:消灯　1:点灯

RCAN　コマンド
RCAN-2 の外部ループバックテスト機能を実行します。

RCAN⏎
ex)
<TX> 00000000 00 01 02 03 04 05 06 07 // 00->07 数字を送信　data++
<RX> 00000000 00 01 02 03 04 05 06 07 // 00->07 数字を受信

E2P　コマンド

EEPROM の Read/Write 処理をします。

E2P␣{R/W}␣EEPROM アドレス␣{メモリアドレス}␣サイズ⏎

{READ}
E2P␣R␣EEPROM アドレス␣サイズ⏎
ex)
E2P␣0x0␣0x100⏎　　// EEPROM の 0x0 番地から 0x100 サイズ分ダンプ表示

{WRITE}
E2P␣W␣EEPROM アドレス␣メモリアドレス␣サイズ
ex)
E2P␣W␣0x0␣0x2000_0000␣0x80　　// EEPROM の 0x0 番地に 0x1_0000 番地の内
　　// 容を 0x80 サイズ分 Write

この EEPROM は、MAC アドレス内蔵の EEPROM です。
EEPROM の(0x80~0xFF)は、ライトプロテクトになっていますので Write できません。
MAC アドレスは、【0xFA~0xFF】の 8 バイトに格納してあります。

{Read Only Memory アドレス}
・EEPROM エリア　　{0x80 ～0xFF}
・内蔵 RAM エリア　　{0x0 ～0x1_FFFF}

{Write Memory アドレス}
・EEPROM エリア　　{0x0 ～0x7F}
・内蔵 RAM エリア　　{0x1_0000 ～0x1_FFFF}

E2P コマンドには LAN 用データ設定コマンドが用意されています。

①E2P␣MAC　　{Read Only}
　MAC の表示

②E2P␣NAME　　{Read/Write}
　ネットワークインターフェイス名の表示と設定

　{Read}
　E2P␣NAME⏎

　{Write}
　E2P␣NAME␣{名前}⏎　　//　MAX 7 ASCII 文字 E2P␣NAME␣RZsamp1⏎

③E2P␣PORT　　{Read/Write}
PORT 番号の表示と設定

{Read}
E2P␣PORT⏎

{Write}
E2P␣PORT␣{xxxxx}⏎　　// E2P ␣PORT␣50000⏎

④E2P␣IP　　{Read/Write}
デフォルト IP アドレスの表示と設定

{Read}
E2P␣IP⏎

{Write}
E2P␣IP␣{xx.xx.xx.xx}⏎　　// E2P ␣IP␣192.168.21.12⏎

⑤E2P␣SUB　　{Read/Write}
サブネットマスクの表示と設定

{Read}
E2P␣SUB⏎

{Write}
E2P␣SUB␣{xx.xx.xx.xx}⏎ // E2P ␣SUB␣255.255.255.0⏎

⑥E2P␣GATE　　{Read/Write}
ゲートウェイアドレスの表示と設定

{Read}
E2P␣GATE⏎

{Write}
E2P␣GATE␣{xx.xx.xx.xx}⏎　　// E2P ␣GATE␣192.168.21.126⏎

⑦E2P␣DHCP　　{Read/Write}
DHCP アドレスの表示と設定

{Read}
E2P␣DHCP⏎

{Write}
E2P␣DHCP␣{xx.xx.xx.xx}⏎　　// E2P ␣GATE␣192.168.21.63⏎

⑧E2P␣DNS　　{Read/Write}
DNS アドレスの表示と設定

{Read}
E2P␣DNS⏎

{Write}
E2P␣DNS␣{xx.xx.xx.xx}⏎ // E2P ␣GATE␣192.168.21.126⏎

注記
使用するルータに設定されている IP アドレスを参照して正しいデータを設定して下さい。

IP　コマンド
DHCP 等により内部登録された、ネットワークインターフェイス名、IP アドレス、ポート番号を表示する。

IP⏎

ex)
RXsamp1　:　192.168.21.21　:　50000

PING　コマンド
PING コマンドを発行する。

PING␣xxx.xxx.xxx.xxx⏎

ex)
PING␣192.168.21.22⏎
32 bytes from 192.168.21.22: icmp_seq = 1, time = 2 ms
32 bytes from 192.168.21.22: icmp_seq = 2, time = 1 ms
32 bytes from 192.168.21.22: icmp_seq = 3, time = 1 ms

DHCP　コマンド
DHCP 処理を実行

DHCP⏎

ex)
Successfully assigned by DHCP server (RXsamp1)
[Ethernet Address　] : [00-1E-C0-F0-E8-7F]
[My IP Address　　] : [192.168.21.21]
[Default IP Address] : [192.168.21.12]
[Default Gateway　] : [192.168.21.126]
[Subnet Mask　　] : [255.255.255.0]
[DHCP　　] : [192.168.21.63]
[DNS　　] : [192.168.21.126]
[PORT Number　　] : [50000]

SNTP　コマンド
SNTP サーバーにアクセスして標準時間を取得し、RTC に年月日曜日と時分秒を設定する。

SNTP⏎

ex)
Tue Mar 31 17:49:29 2015
DATE 2015/03/31[Tue]
TIME 17:49:29

DNS　コマンド
DNS サーバーにアクセスして IP アドレスを取得する。

DNS␣{ドメイン名}⏎

ex)
DNS␣www.yahoo.co.jp⏎
www.yahoo.co.jp has address 182.22.59.229

GMDATE　コマンド
SNTP コマンドにより取得したグリニッジ標準時間を表示する。

GMDATE⏎

ex)
Tue Mar 31 08:53:18 2015

SWB　コマンド
EV-RX-xx 側の SW2_2～5 の状態を表示します。

SWB⏎

ex)
BaseBord-SW2_2[ON/OFF] SW2_3[ON/OFF] SW2_4[ON/OFF] SW2_5[ON/OFF]

LEDB　コマンド
EV-RX-xx 側の LED2～5 を点灯・消灯します。

LEDB␣{0/1}␣{0/1}␣{0/1}␣{0/1}⏎　// LEDB {LED2} {LED3} {LED4} {LED5} 0:消灯　1:点灯

CTEXT　コマンド
キャラクタ LCD に英数文字を表示します。

CTEXT␣{0~19}␣{0~3}␣{Text}⏎　// CTEXT {X:列}{Y:行}{英数文字}

MTEXT　コマンド
モノクロ LCD に英数文字／漢字を表示します。

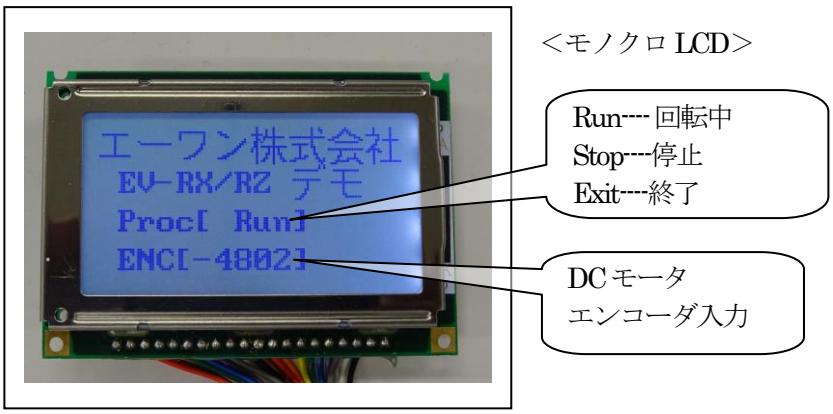
MTEXT␣{0}␣{0~3}␣{文字}⏎　//MTEXT {X:0 固定}{Y:行}{文字}

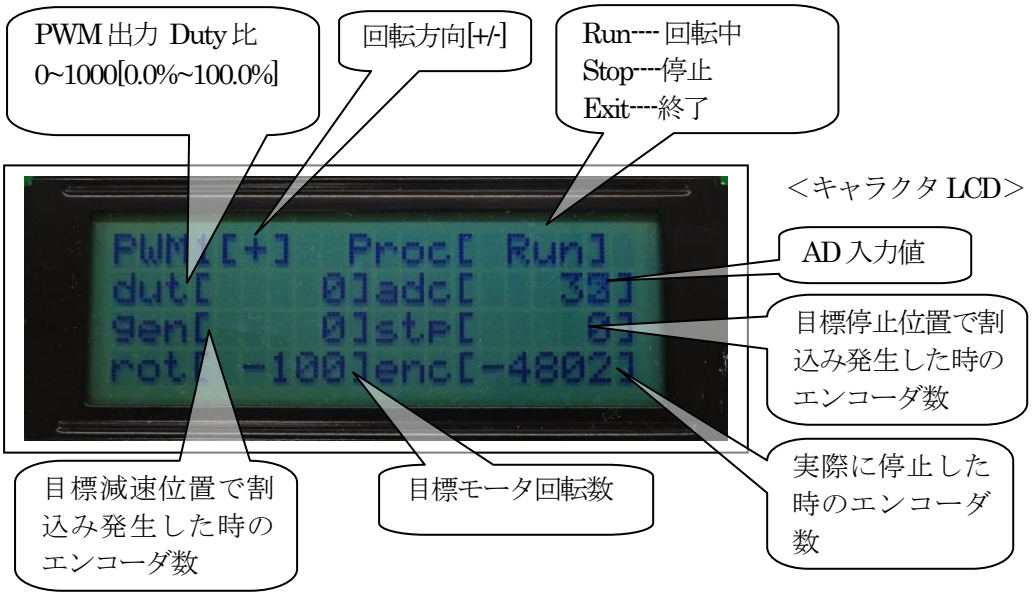
PWM　コマンド
DC モータの回転とデモ運転します。

PWM⏎　// SWB [5:++duty 4:--duty 3:demo 2:exit]

EV-RX-xx 基板上の SW 指示により DC モータが動作します。
SW5[ON] ---------- [+]方向に回転
SW4[ON] ---------- [−]方向に回転
SW3[ON] ---------- デモ運転
SW2[ON] ---------- PWM 処理終了

キャラクタ LCD とモノクロ LCD にデモ表示します。





KEY操作

簡単な１ラインエディタ機能を入れてあります。

・BS　　バックスペース
・←　　左にカーソル移動
・→　　右にカーソル移動
・↑　　１回前に入力した内容のリコール
・ESC　　コマンド処理中の中断

２－２．「EVRX_NORTi_USB」プロジェクトの説明

１）動作説明

・EVRX_NORTi に USB-Function 機能を追加したプロジェクトになります。
・各コマンド体系は、２－２項を参照して下さい。
・USB-Function ライブラリーは、別途有償にて提供しております。ご購入前の評価用として実行用ファイルは添付しております。

２）フォルダ構成とファイル名(評価用)

Sample_RX¥_			
	EV-RX¥_EVRX_USB	EVRX_NORTi_USB.mot	実行用 Hex ファイル
	_PC_Test	ITF_USB_TEST..EXE	PC 用 USB テストプログラム
		DRIVER¥ITFUSBLib	PC 側 USB ドライバー
		TCP_IP_TEST.EXE	PC 用 TCP/IP テストプログラム

３）フォルダ構成とファイル名(有償用)　ご購入 ITFUSBLib_RX63N_xx を添付

Sample_RX¥EV-RX¥Hew4¥EVRX_NORTi_USB			【Hew4】
	Debug	ビルドにより生成された実行ファイル等が格納される場所	
	ITF_LIB	空	
		ReadMe.txt	ITF_LIB オリジナル CD からのインポート手順書
	lnk_NORTi	空	EVRX_NORTi¥NORTi にリンク
	lnk_NORTi_smp	空	EVRX_NORTi¥NORTi_smp にリンク
	lnk_app	空	EVRX_NORTi¥src_app にリンク
	lnk_eva	空	EVRX_NORTi¥src_eva にリンク
	lnk_evb	空	EVRX_NORTi¥src_evb にリンク
	lnk_sys	空	EVRX_NORTi¥src_sys にリンク

「EVRX_NORTi_USB」に黄色部分(有償)がインポートします。

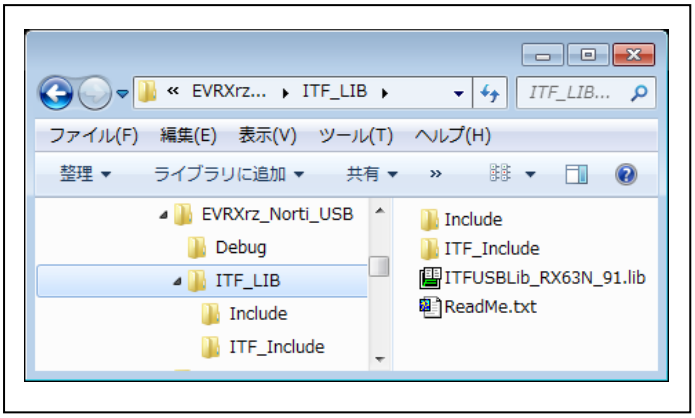
Sample_RX¥EV-RX¥Cube¥ EVRX_NORTi_USB			【CS+】
	DefaultBuild	ビルドにより生成された実行ファイル等が格納される場所	
	ITF_LIB	空	Hew4¥EVRX_NORTi_USB¥ITF_LIB にリンク
	lnk_NORTi	空	Hew4¥EVRX_NORTi¥NORTi にリンク
	lnk_NORTi_smp	空	Hew4¥EVRX_NORTi¥NORTi_smp にリンク
	lnk_app	空	Hew4¥EVRX_NORTi¥src_app にリンク
	lnk_eva	空	Hew4¥EVRX_NORTi¥src_eva にリンク
	lnk_evb	空	Hew4¥EVRX_NORTi¥src_evb にリンク
	lnk_sys	空	Hew4¥EVRX_NORTi¥src_sys にリンク

4）ITF_LIB オリジナル CD（有償）からサンプル CD にインポートする手順

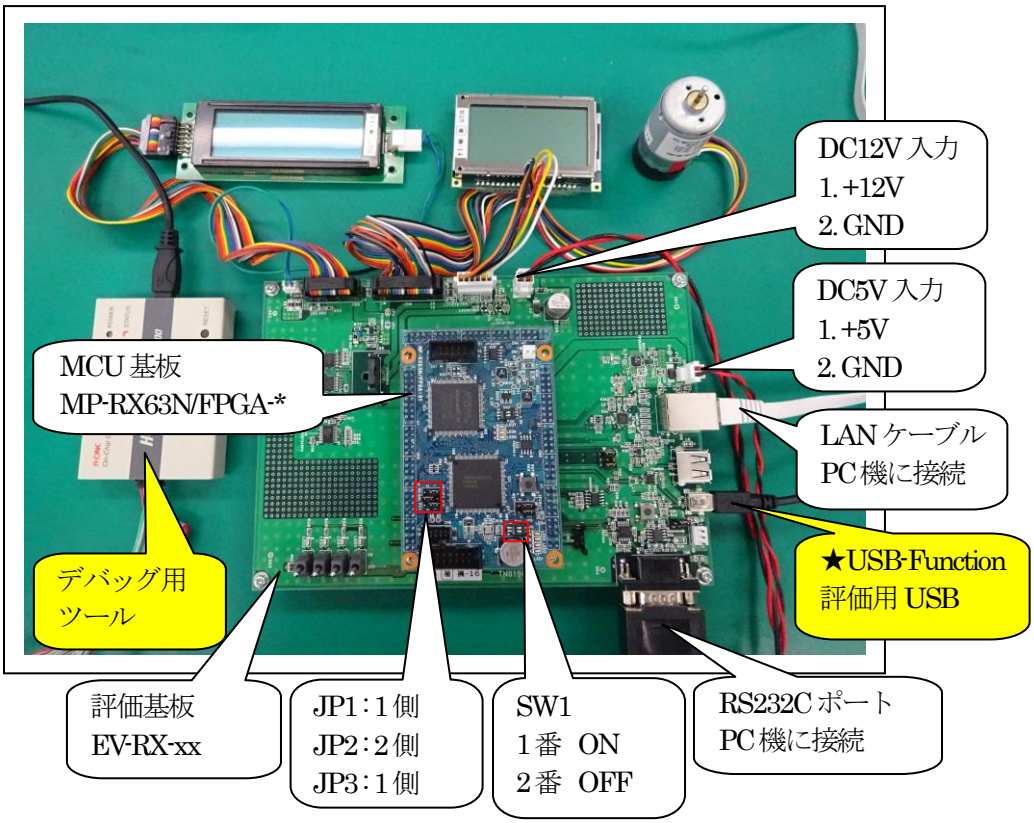
a． 3）表の黄色部は、空ディレクトリになっていますので、ITF_LIB オリジナル CD から必要なファイルを Copy します。

ITF_LIB オリジナル CD		サンプル(EVRX_NORTi_USB)
ITF_LIBi¥Include	→	ITF_LIB¥Include
ITF_LIB¥ITF_Include	→	ITF_LIB¥ITF_Include
ITF_LIB¥ITFUSBLIB_xx.lib	→	ITF_LIB¥ITFUSBLIB_xx.lib

上記のように ITF_LIB オリジナル CD から、サンプル「EVRX_NORTi_USB¥ITF_LIB」の空ディレクトリに Copy して下さい。

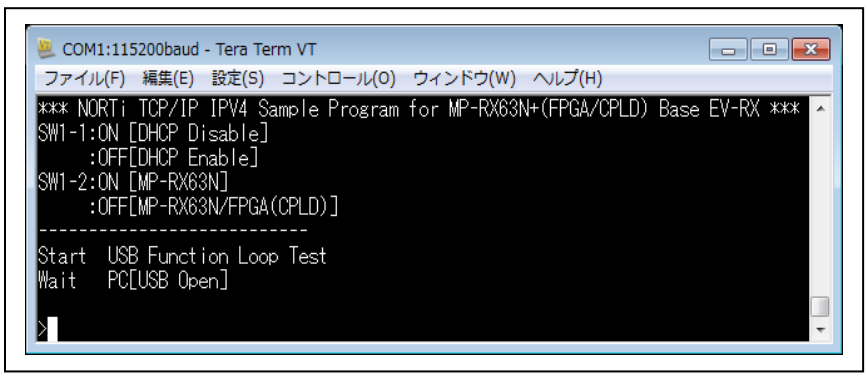


5）動作構成（電源 OFF）



6）動作手順

a．ターゲット基板側の電源 OFF の状態で上図の★USB-Function 評価用 USB ケーブル以外を接続する。
b．ターゲット基板側の電源を ON にする。
c．デバッガ「DEF」を立ち上げる。
d．デバッガ「DEF」画面の左下隅の「Start」をクリックする。
e．デバッガ「DEF」の【オプション】－【フラッシュ ROM ライター】を起動する。
f．無償評価用 Hex ファイル「EVRX_NORTi_USB.mot」を内蔵 ROM へ書き込みをする。
g．ターゲット側の電源を OFF にする。
h．ジャンパー(JP2)を 1 側にする。
i．デバッガツールを外す。
j．★USB-Function 評価用 USB ケーブルを PC 機に接続する。
k．RS232C ケーブルが PC 機に接続されているのを確認後、「TeraTerm pro」を起動する。
l．ターゲット基板側の電源を ON にする。

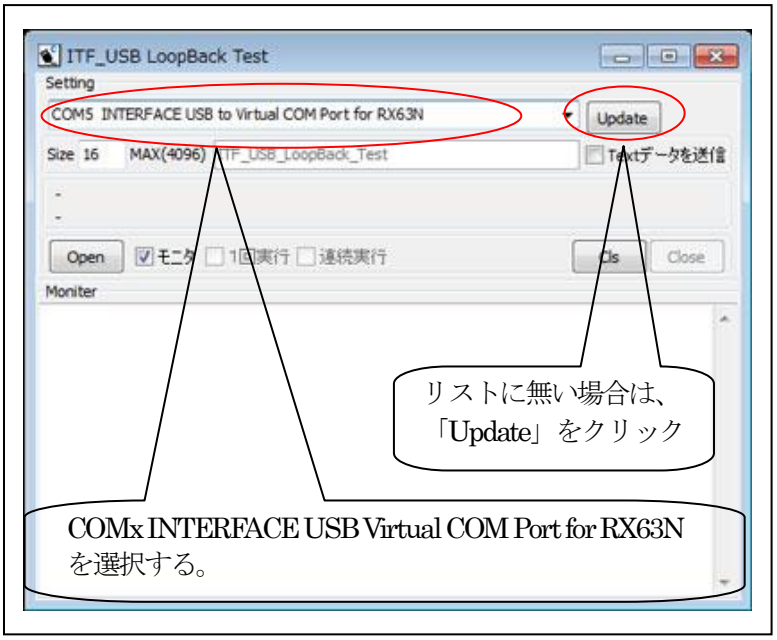


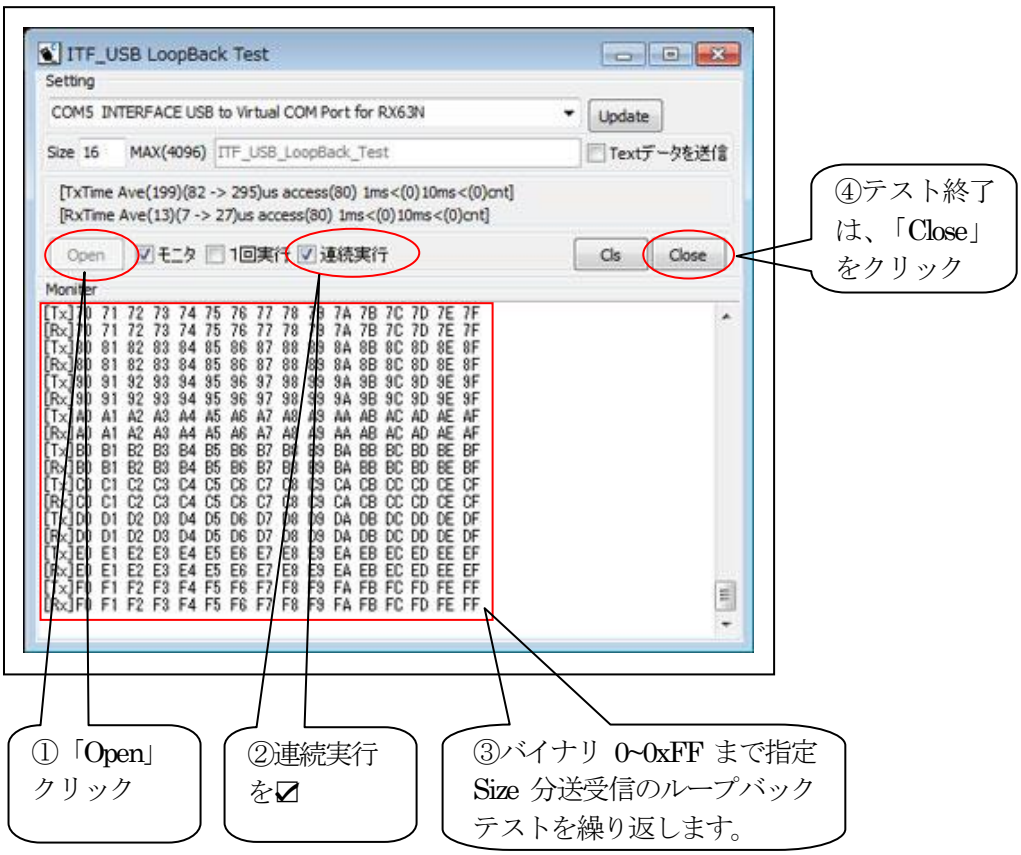
TeraTerm pro の起動画面

7）USB ファンクションの動作確認

a．Windows が、USB ドラーバーのインストールを要求しますので USB-Driver をインストールする。
「Sample_RX¥_PC_Test¥ DRIVER¥ITFUSBLib」にドライバーがあります。

b．Windows 側のテストプログラム「ITF_USB_TEST.EXE」を起動する。





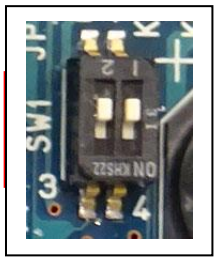
8）TCP/IP プロトコル通信の動作確認

a．TCP/IP 通信に必要な情報を EEPROM 登録する。（ルータの設定データ）
２－１「EVRX_NORTi」プロジェクトの説明
６）各コマンドの説明・q．E2P　コマンド を参照

ex)
Successfully assigned by DHCP server (RXsamp1)
[Ethernet Address] : [00-1E-C0-F0-E8-7F]
[My IP Address] : [192.168.21.21]
[Default IP Address] : [192.168.21.12]
[Default Gateway] : [192.168.21.126]
[Subnet Mask] : [255.255.255.0]
[DHCP] : [192.168.21.63]
[DNS] : [192.168.21.126]
[PORT Number] : [50000]

上記の全情報を登録しないと TCP/IP 通信は出来ません。

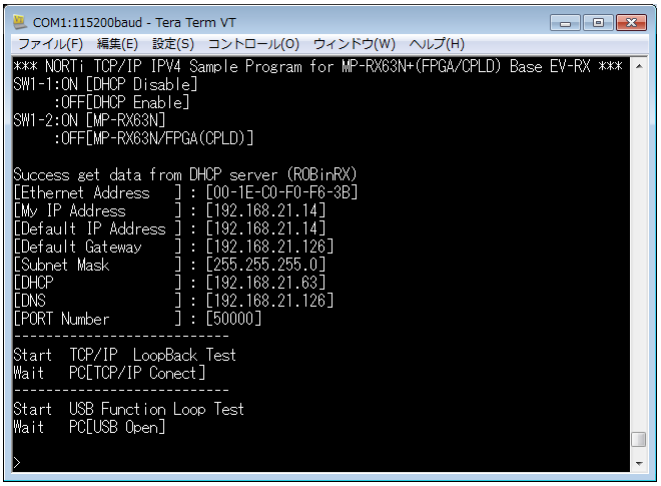
b．MCU 基板上の SW1 設定



SW1-1　OFF
[ON]- 起動時に DHCP サーバーへのアクセスを禁止する。
[OFF] 起動時に DHCP サーバーへのアクセスを許可する。
SW1-2　OFF

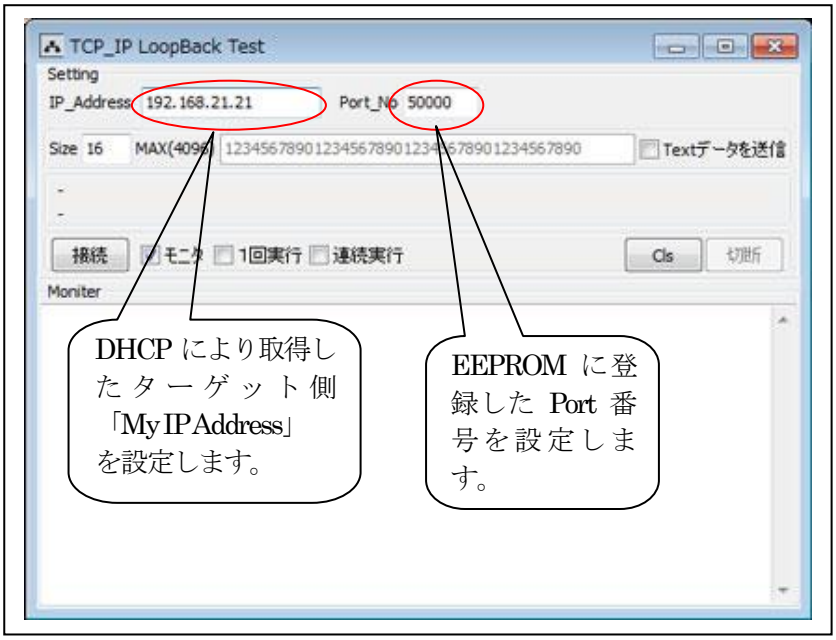
c．ターゲット基板側の電源を入れ直します。
・EEPROM に情報が全て登録されていない場合は、
TeraTerm 画面に「There is no LAN data in EEPROM」と表示します。

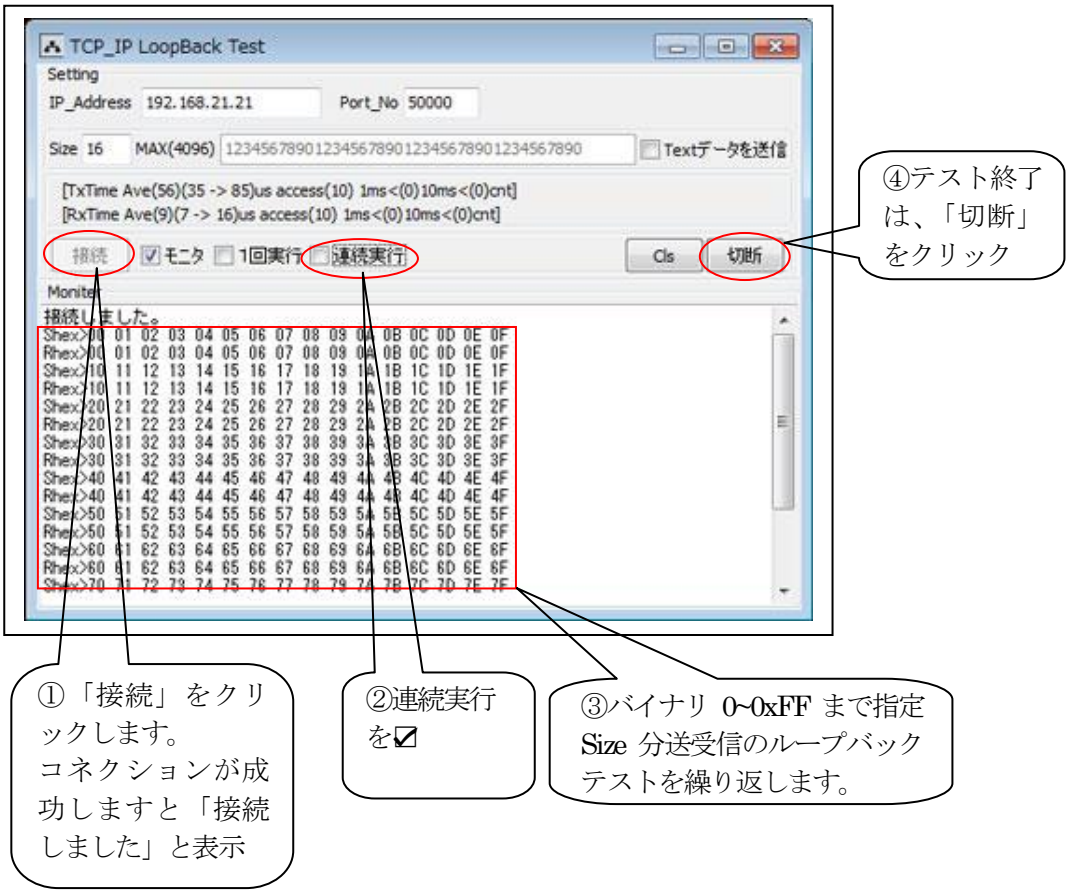
・サンプルソフトは電源 ON 時にルータに対して DHCP 処理(IP アドレスの取得処理)を実行します。



DHCP 処理が成功すると「My IP Address」に取得した IP アドレスを表示します。

d．Windows 側のテストプログラム「TCP_IP_TEST..EXE」を起動する。





以上です。

３．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。
・本サンプルプログラムに関して、インターフェイス社、ミスポ社、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、サンプルソフトを使用した結果、不具合が発生しても、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。

４．商標

・RX および RX63N 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。

５．参考文献

・「RX63N グループ　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
・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
http://www.robin-w.com