

アプリケーションノート μNet3 サンプルプログラム エコーサーバ

> Rev.1.0 April 2, 2019

## 改訂記録

			改訂内容
Rev.	発行日	ページ	内容
1.0	2019.04.02	_	初版

# 目次

改訂記錄	禄	2
目次	3	
1.	概要	4
2.	ソースファイル	5
3.	プロジェクトへ組み込む手順	6
3.1.	プロジェクトヘファイルを追加する。	6
3.2.	コンフィグレータ・ソースファイル編集	7
3.2	2.1. タスクの作成	7
3.2	2.2. ソケットの作成1	0
3.2	2.3. タスクの起動1	5
3.2	2.4. ソースコード生成1	5
4.	ソースコードの解析1	7
4.1.	UDP 版1	7
4.2.	TCP 版1	7
5.	動作確認1	8
5.1.	UDP版1	8
5.2.	TCP版	1

### 1. 概要

本ドキュメントはμ**Net3**のソケットを使用した簡単なサンプル「エコーサーバ」を組み込 む手順、使用方法について説明します。

エコーサーバは通信相手から受信したデータを、通信相手へ送信します。



UDP で動作する UDP 版、TCP で動作する TCP 版を用意しています。

エコーサーバのプログラム構造については簡単なプログラムであり、またソースコード上 のコメントで解説しているので、ソースコードを参照してください。 また、μ Net3 の各 API の詳細はμ Net3 のユーザーズガイドを参照してください。

μ C3 はイー・フォース株式会社の登録商標です。
 μ Net3 はイー・フォース株式会社の登録商標です。
 本書で記載されている内容は予告無く変更する場合があります。
 い。



## 2. ソースファイル

エコーサーバは次のソースファイルを使用します。

表 2.1 ソースファイル

ファイル名	内容
sample_socket_echo_server_tcp.c	uNet3 のソケットを使用したエコーサーバ
	のサンプルプログラム(TCP版)
sample_socket_echo_server_udp.c	uNet3 のソケットを使用したエコーサーバ
	のサンプルプログラム(UDP版)



## 3. プロジェクトへ組み込む手順

パッケージに収録されているネットワークサンプルプログラムへエコーサーバを追加する 手順を説明します。

## 3.1. プロジェクトへファイルを追加する。

Sample - IAR Embedded Workbench IDE

「**2** ソースファイル」のファイルをサンプルプログラムが収録されているフォルダへコピー してください。

その後、コピーしたファイルをプロジェクトへ追加してください。

IAR Embedded Work Bench の場合は以下のような形になります。

ファイル( <u>F</u> ) 編集( <u>E</u> ) 表示( <u>V</u> ) プロジェクト( <u>P</u> ) I-jet/JTAGjet( <u>I</u> ) ツール( <u>T</u> )	ウィンドウ( <u>W</u> )	ヘルプ( <u>H</u>	)		
다 🖆 🖬 🕼 👗 🖻 💼 🗠 어머니	~ 1	× 🐂 🦄	Ŧ 🏹	•	>
ワークスペース		×			
SRAM		~			
ファイル	٣.	۲ <u>.</u>			
🗆 🗇 sample – SRAM *	~				
🛛 🛏 🗖 Driver					
🛛 🛏 🗀 Kernel		- <b>F</b>			
🛛 🛏 🗀 Net					
🛛 🛏 🗀 Util		<b>-</b>			
🛛 🛏 🗀 Wifi					
–⊞ 🖸 cgi_sample.c	追加				
tw_param.c					
- 🕀 🖸 sample_socket_echo_server_tcp.c					
- 🕀 🖸 sample_socket_echo_server_udp.c					
1					

他のビルドツールをご使用の場合は、そのツール向けに読み替えてください。

eForce

### 3.2. コンフィグレータ・ソースファイル編集

エコーサーバはタスクから実行するので、タスクとソケットを作成、起動してください。

タスクの作成、ソケットの作成は uC3/Compact と uC3/Standard で異なります。 ご利用のパッケージが uC3/Compact か uC3/Standard か確認してください。

## 3.2.1. タスクの作成

#### ■uC3/Compact の場合

コンフィグレータにタスクを追加します。

💿 uC3Configurator							
: ファイル( <u>F</u> ) 表示( <u>V</u> ) プロジェクト( <u>P</u> )	ツール( <u>T</u> ) ヘルプ( <u>H</u> )						
i 🐎 💕 💽 i 🕸 🚰 i 🎕 💿 🕕							
Kernel 👻 🗙							
💿 カーネル全般	タスク一覧						
920	ID	状態	関数名	優先度	コタックサイブ	뭐개로뚫	拡張情幸
1 2777	ID_IO_TSK	実行可能状態	IoTask	4		ŧ	
1/2/250	ID_INIT_TSK	実行可能状態	InitTask	4	\ <u></u>	튭	
2 テータキュー	id_WLAN_CTL_TSK	休止状態	wlan_ctl_tsk	4	追加	Ð	
<b>ク</b> メールボックス	id_WLAN_IO_TSK	休止状態	wlan_io_tsk			Đ	
「四」 固定長メモリプール	id_WLAN_SND_TSK	休止状態	wlan_snd_tsl	4		Ā	
🔞 周期ハンドラ	ID_TCP_TIM_TSK	休止状態	net_tim	4	512	尚被言語	
🎲 割込みサービスルーチン	ID HTTPD TSK1	休止状態	bt au tskil	4	768	高級言語	
	ID_UDP_ECHO_TSK	休止状態	EchoServerTas	4	1024	高級言語	
1 共有スタック	ID_TCP_ECHO_TSK	休止状態	EchoServerTas	4	1024	高級言語	
							J
II II							

	ID	UDP	ECHO	TSK (	(UDP	版のタ	マスク)
--	----	-----	------	-------	------	-----	------

タスク		×
IDの定義名 タスクの関数名 優先度の初期値 払張情報	スタック ・ファン・ファン・ファン・ファン・ファン・ファン・ファン・ファン・ファン・ファン	
- タスク属性 □実行可能状態(TA_ACT) □制約タスク(TA_RSTR)	<ul> <li>○ 共有スタックを使用</li> <li>共有スタックID</li> </ul>	
記述言語 ・ 高級言語(TA_HLNG)   ・ アセンブリ言語(TA_ASM)		
	OK キャンセノ	١

# ID\_TCP\_ECHO\_TSK (TCP 版のタスク)

タスク						$\times$
IDの定義 タスクの「 優先度 拡張情報 タスクが □ 実 □ 制		ID_TCP_ECHO_TSK EchoServerTask_TCP 4 ( A_ACT) STR)	スタック       ・	スタックを使用 イズ 1024 やりを使用 ックID	×	
記述 ア 4	級言語(TA_H 2ンブリ言語(TA	L <b>NG)</b> A_ASM)			ОК <b>キャン</b>	セル

#### ■uC3/Standard の場合

uC3/Standard の API acre\_tsk()を使用してタスクを作成してください。

acre\_tsk の引数の値 ・UDP 版 tskatr タスク属性 TA HLNG **exinf** タスクの拡張情報 0 task タスクの起動番地 EchoServerTask\_UDP タスクの起動時優先度 itskpri 4 stksz タスクのスタック領域のサイズ 1024 stk タスクのスタック領域の先頭番地 0 ・TCP 版 tskatr タスク属性 TA\_HLNG exinf タスクの拡張情報 0 タスクの起動番地 task EchoServerTask UDP itskpri タスクの起動時優先度 4 タスクのスタック領域のサイズ stksz 1024 stk タスクのスタック領域の先頭番地 0

## 3.2.2. ソケットの作成

■uC3/Compact の場合

コンフィグレータにソケットを追加します。

ソケットは UDP 版向けに UDP ソケットを 1 つ、TCP 版向けに TCP ソケットを 1 つ追加 します。

uC3Configurator						
: ファイル( <u>F</u> ) 表示( <u>V</u> ) プロジェクト( <u>P</u> )	ツール( <u>T</u> ) ヘルプ( <u>H</u> )					
i 🕃 💕 🚺 🛸 🚰 i 🎕 🞯 🕕						
TCP/IP	ソケット一覧					_
<ul> <li>         ・         ・         ・</li></ul>	ソケットID ID_SOC_DHCP ID_ICMP ID_SOC_DNS	インタフェースID	IPパージョン IPv4 IPv4 IPv4	70 UD	追加	
③ IPアドレス取得	ID_SOC_HTTP1		IPw*	TOP	80	:
	ID_SOC_SNTPC		1Pv4	UDP	0	
	ID_UDP_ECHO		IPv4	UDP	12345	
	ID TOP ECHO		IP∨4	TCP	12345	



ID\_UDP\_ECHO のソケット(UDP 版で使用するソケット)

ソケット		×
IDの定義名	ID_UDP_ECHO	
インターフェースのバインディング	~	•
IPバージョン	IPv4 ~	
プロトコル	UDP ~	
ローカルポート	12345	
		待ち受けの
タイムアウト設定		ポート番号を入力
snd_socのタイムアウト(ミリ秒)	-1	
rcv_socのタイムアウト(ミリ秒)	-1	
con_socのタイムアウト(ミリ秒)	-1	
cls_socのタイムアウト(ミリ秒)	-1	
TCPバッファ設定		
送信バッファ(byte)	1024	
受信バッファ(byte)	1024	
	OK キャンセ	١



ID\_TCP\_ECHOのソケット(TCP版で使用するソケット)





■uC3/Standard の場合

ソケットの作成はエコーサーバの初期化処理で実行していますので、特に何も行う必要は ありません。既にソースコードに実装しています。

以下はソケットを作成している箇所です。

・UDP 版

sample\_socket\_echo\_server\_udp.c 𝒪 echo\_server\_UDP()





・TCP 版

sample\_socket\_echo\_server\_tcp.c Ø echo\_server\_TCP( )

/\* エコーサーバ処理 \*/↩ ER echo\_server\_TCP(void)↔ {↩ ER ercd:← T\_NODE host; ← VB ebuf[256]; ← ここでソケットを 作成しています。 #ifndef NET\_C↩ ID ID\_TCP\_ECHO; ←  $\leftarrow$ /\* ソケットの生成 \*/↩ memset(&host, 0, sizeof(host)); ← host num =  $1: \leftarrow$ host.ver = IP VER4;↔ host.ipa = INADDR\_ANY;↔ host.port = 12345; /\* 待ち受けのポート ercd = cre\_soc(IP\_PROTO\_TCP, &host); <if (**0** >= ercd) {↩ return ercd; ← } ← ID TCP ECHO = ercd; ← #endite

#### 3.2.3. タスクの起動

「3.2.1 タスクの作成」で作成したタスクを起動させるには $\mu$  C3 の API「act\_tsk()」を実行する必要があります。

これは、ソースコードをテキストエディタで編集してください。

タスクの起動はμNet3の初期化が完了した後に実行してください。

具体的には、パッケージに収録されているサンプルプログラムでは net\_setup()の後になり ます。



#### 3.2.4. ソースコード生成

コンフィグレータを使用している場合は、タスク、ソケットを追加したので、ソースコー ドを生成してください。出力先のフォルダはは元にしたサンプルプログラムと同じです。

メニューから「プロジェクト」→「ソース生成」

💩 uC3Configurator						
: ファイル(F) 表示(V) プロジェクト(P)	ツール(T) ヘルプ(H)					
🗄 📸 🚺 🛸 😭 🖄 ソース生成	(G) F7					
TCP/IP 😭 プロパティ(F	P) Alt+F7					
🐌 uNet3全般	ソケット一覧					
も インタフェース	ソケットID	インタフェースID	IPバージョン	プロトコル	ローカルポート	snd_s
<ul> <li>シゾケット</li> </ul>	ID_SOC_DHCP		IPv4	UDP	68	3000
● ネットアフリケーション	ID_ICMP		IPv4	ICMP	0	-1
SSL	ID_SOC_DNS		IPv4	UDP	0	5000
💿 IPアドレス取得	ID_SOC_HTTP1		IPv4	TOP	80	25000
	ID_SOC_SNTPC		IPv4	UDP	0	3000
	ID_UDP_ECHO		IPv4	UDP	12345	-1
	ID_TCP_ECHO		IPv4	TCP	12345	-1



ダイアログが起動するので、生成先を入力してください。

フォルダーの参照	×
Generate source to the folder:	
> WBSACVLXY.SSL	MQTT
> WBSACVLXY.uAP	
> WBSACVLXY.WPS	
> WBSACVLXY-1	
> WBSACVLXY-1.FV	V_UP
> WBSACVLXY-1.H	ПРС
> 🛃 WBSACVLXY-1.N	ET
> WBSACVLXY-1.R	rc l
> WBSACVLXY-1.SS	L 🗸
フォルダー(E): WBSACVLXY-1.NET	
新しいフォルダーの作成( <u>N</u> )	OK キャンセル



## 4. ソースコードの解析

#### 4.1. UDP 版

次の順に実行されます。

- (1) 「3.2 コンフィグレータ・ソースファイル編集」で追加した act\_tsk()でエコーサーバを 実行するタスクが起動します。
- (2) タスクは「sample\_socket\_echo\_server\_udp.c」の関数 echo\_server\_UDP()を実行しま す。

上記の順にソースコードを参照してください。

ソースコードには解説のたえに、コメントを書いております。

- また、このプログラムは、μ Net3 の API を実行しています。
- $\mu$  Net3 の API の詳細は $\mu$  Net3 のユーザーズガイドを参照してください。

#### 4.2. TCP版

次の順に実行されます。

- (1) 「3.2 コンフィグレータ・ソースファイル編集」で追加した act\_tsk()でエコーサーバを 実行するタスクが起動します。
- (2) タスクは「sample\_socket\_echo\_server\_tcp.c」の関数 echo\_server\_TCP()を実行しま す。

上記の順にソースコードを参照してください。 ソースコードには解説のたえに、コメントを書いております。

- また、このプログラムは、μ Net3 の API を実行しています。
- μ Net3 の API の詳細は μ Net3 のユーザーズガイドを参照してください。

#### 5. 動作確認

ターゲットへのダウンロード、ネットワーク接続の手順、使用する機材についてはパッケージの サンプルプログラムのドキュメントを参照してください。

ここでは、追加機能「エコーサーバ」の動作確認を説明します。

5.1. UDP版

PC とターゲットを同じネットワークへ接続して、PC から UDP データを送信してください。 ターゲットは UDP データを受信して同じデータを PC へ送信します。

適当なツールを使用して PC からターゲットへ UDP データを送信して、その後、受信データを 確認してください。

ここでは、netcat を使用した場合の手順を説明します。

netcat は UNIX 系 OS のコマンドラインツールです。Linux や Cygwin 等で利用できます。 netcat のコマンド名は「nc」としていますが、ご利用の環境に合わせて読み替えてください。

次のコマンドを入力してください。
 nc -u IP アドレス ポート番号

IP アドレスはターゲットの IP アドレス ポート番号はターゲットが待ち受けているポート番号





(2)送信する文字列を入力後、Enterを押してください。

下の画像では「01234567890abcdef」を送信します。



(3) データを受信すると、受信データが表示されます。



(4) Ctrl-C を入力することで、netcat は終了します。



通信のパケットを Wireshark で取得すると、PC からの送信とターゲットからの応答があること がわかります。

🚄 *Realtek PCle GBE Family Controller: イーサネット										
ファイル( <u>F</u> ) 編集( <u>E</u> ) 表示	示( <u>V</u> ) 移動( <u>G</u> ) キャプチャ( <u>C</u> )	分析( <u>A</u> ) 統計( <u>S</u> ) 電話(⊻)	無線( <u>W</u> ) ツール	( <u>T</u> ) ヘルプ	( <u>H</u> )					
🧉 🔳 🖉 💿 📘 🚠	🔀 😂 । ९ 👄 🔿 🗟 🤇	🕅 🕹 📃 📃 🔍 Q, Q,								
■ 表示フィルタ … 〈Otrl-/〉を適用します										
Time	Source	Destination	Protocol	Length	Info					
0.00000	192.168.11.88	192.168.11.100	UDP	60	54554	→ 12				
0.003194	192.168.11.100	192.168.11.88	UDP	60	12345	→ 54				
<										
> Frame 2: 60 byt	es on wire (480 bit	s), 60 bytes captured	(480 bits)	on inter	face 0					
> Ethernet II, Src: TaiyoYud_99:a1:e4 (ac:3f:a4:99:a1:e4), Dst: Clevo_2d:a2:71 (80:fa:5										

- > Internet Protocol Version 4, Src: 192.168.11.100, Dst: 192.168.11.88
- > User Datagram Protocol, Src Port: 12345, Dst Port: 54554
- v Data (18 bytes)
  Data: 30313233343536373839306162636465660a
  [Length: 18]
- PCのIPアドレス 192.168.11.88

ターゲットの IP アドレス 192.168.11.100



#### 5.2. TCP 版

PC とターゲットを同じネットワークへ接続して、PC から TCP でデータを送信してください。 ターゲットはデータを受信して同じデータを PC へ送信します。

適当なツールを使用して PC からターゲットへ TCP 接続してデータを送信して、その後、受信 データを確認してください。

ここでは、netcatを使用した場合の手順を説明します。

netcat は UNIX 系 OS のコマンドラインツールです。Linux や Cygwin 等で利用できます。 netcat のコマンド名は「nc」としていますが、ご利用の環境に合わせて読み替えてください。

(1) 次のコマンドを入力してください。

nc IP アドレス ポート番号

IP アドレスはターゲットの IP アドレス ポート番号はターゲットが待ち受けているポート番号





- (2)送信する文字列を入力後、Enterを押してください。
  - 下の画像では「01234567890abcdef」を送信します。



(3) データを受信すると、受信データが表示されます。



(4) Ctrl-C を入力することで、netcat は終了します。



通信のパケットを Wireshark で取得すると、PC からの送信とターゲットからの応答があること がわかります。

🚄 \*Realtek PCIe GBE Family Controller: イーサネット

ファイル(E) 編集(E) 表示(V) 移動(G) キャプチャ(C) 分析(A) 統計(S) 電話(y) 無線(W) ツール(D) ヘルプ(H)

🛋 🔳 🖉 💿 📙 🔚 🕱 🔂 I 9. 🗢 🗢 🕾 T 👲 🚍 🖳 9. 9. 9. 9.

ip.addr==192.168.11.100									
Time	Source	Destination	Protocol	Length	Info				
16.107057	192.168.11.88	192.168.11.100	TCP	66	64640 → 12345 [SYN				
16.110411	192.168.11.100	192.168.11.88	TCP	62	12345 → 64640 [SYN				
16.110482	192.168.11.88	192.168.11.100	TCP	54	64640 → 12345 [ACK				
53.279801	192.168.11.88	192.168.11.100	TCP	72	64640 → 12345 [PSH				
53.283031	192.168.11.100	192.168.11.88	TCP	60	12345 → 64640 [ACK				
53.291881	192.168.11.100	192.168.11.88	TCP	72	12345 → 64640 [PSH				
53.291978	192.168.11.88	192.168.11.100	TCP	54	64640 → 12345 [ACK				
99.379555	192.168.11.88	192.168.11.100	TCP	54	64640 → 12345 [FIN				
99.386411	192.168.11.100	192.168.11.88	TCP	60	12345 → 64640 [ACK				
99.386412	192.168.11.100	192.168.11.88	TCP	60	12345 → 64640 [FIN				
99.386504	192.168.11.88	192.168.11.100	ТСР	54	64640 → 12345 [ACK				

<

> Frame 23: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface 0

> Ethernet II, Src: TaiyoYud\_99:a1:e4 (ac:3f:a4:99:a1:e4), Dst: Clevo\_2d:a2:71 (80:fa:5b:2d:a2:

> Internet Protocol Version 4, Src: 192.168.11.100, Dst: 192.168.11.88

> Transmission Control Protocol, Src Port: 12345, Dst Port: 64640, Seq: 1, Ack: 19, Len: 18

✓ Data (18 bytes)

Data: 30313233343536373839306162636465660a [Length: 18]

PCのIPアドレス 192.168.11.88

ターゲットの IP アドレス 192.168.11.100



μNet3 サンプルプログラム エコーサーバ

# アプリケーションノート uNet3 サンプルプログラム エコーサーバ

2019年04月02日 Rev.1.0

イー・フォース株式会社 http://www.eforce.co.jp/ お問い合わせ support@eforce.co.jp Copyright (C) 2019 eForce Co.,Ltd. All Rights Reserved.

