TCP/IP プロトコルスタック（TINET）
サンプルアプリケーション（リリース 1.7）［2017/5/15］

1. サンプルアプリケーション

サンプルとして、以下のアプリケーションを提供している。【】内は【アプリケーション名、対応するネットワーク層】である。IPv64 はネットワーク層として IPv6 と IPv4 の両方を使用することを意味しており、IPv64m は、IPv4 は IPv6 の IPv4 射影アドレスにより実装している。

(1) IPv6 TCP ECHO サーバ【`echos6`、IPv6】
　ネットワークアプリケーションに必要な各ファイルの設定方法の参考となる、TCP エコーサーバ機能のみのシンプルなアプリケーションである。

(2) IPv4 TCP ECHO サーバ【`echos4`、IPv4】
　ネットワークアプリケーションに必要な各ファイルの設定方法の参考となる、TCP エコーサーバ機能のみのシンプルなアプリケーションである。

(3) IPv6 UDP ECHO サーバ【`usrv6`、IPv6】
　ネットワークアプリケーションに必要な各ファイルの設定方法の参考となる、UDP エコーサーバ機能のみのシンプルなアプリケーションである。

(4) IPv4 UDP ECHO サーバ【`usrv4`、IPv4】
　ネットワークアプリケーションに必要な各ファイルの設定方法の参考となる、UDP エコーサーバ機能のみのシンプルなアプリケーションである。

(5) クライアントサーバ・セット【`nserv`、IPv64/IPv6/IPv64m/IPv4】
　以下に示すサーバが提供されており、必要に応じて組み込むサーバを選択できる。

　[1] WWW サーバ
　[2] TCP エコーサーバ
　[3] UDP エコーサーバ
　[4] TCP ディスカードサーバ
　[5] 簡易コンソール

　また、クライアントとしては以下の機能が提供されており、必要に応じて組み込むクライアントを選択できる。

　[1] TCP エコークライアント
　[2] UDP エコークライアント
　[3] TCP ディスカードクライアント
　[4] UDP ディスカードクライアント
　[5] DNS リゾルバー
　[6] DHCPv6 クライアント
　[7] DHCPv4 クライアント
　[8] ping

(6) TOPPERS ASP/JSP サンプルプログラム sample1 のネットワーク対応プログラム【`sample1n`、IPv6/IPv4】
　TOPPERS/ASP と TOPPERS/JSP のサンプルプログラム sample1 のネットワーク対応プログラムである。telnet で接続すると、シリアルの入出力を引き継いで実行する。切断すると、元のシリアルに入出力を戻す。

(7) 最小構成サーバ【minsv、IPv4】

WWW サーバ・タスクと TCP エコーサーバ・タスクのみからなる最小構成のサーバである。H8/3069F が内蔵している RAM（16K バイト）と ROM（512K バイト）に収まり、外部メモリは不要である。現在は、品川通信計装サービス製 NKEV-010H8（TOPPERS/JSP リリース 1.4.2 のみ）と秋月電子通商製 H8/3069F（TOPPERS/JSP リリース 1.4.1 以降と TOPPERS/ASP）のシステムに対応している。

2. サンプルアプリケーションの構築

TINET サンプルアプリケーションの構築は、TOPPERS/ASP 環境と TOPPERS/JSP 環境におけるのサンプルアプリケーションの構築とほぼ同じである。

(1) TINET コンフィギュレーションスクリプトの実行

TINET サンプルアプリケーションの構築用ディレクトリを作成し、TINET コンフィギュレーションスクリプトを実行する。

[1] TOPPERS/ASP 用 TINET コンフィギュレーションスクリプトの実行

```
$ mkdir NETOBJ
$ cd NETOBJ
$ perl ../tinet/tinet_asp_configure -T akih8_3069f_gcc -A echos6
                                    -i ether -v if_ed -n inet6 -s tcp
```

オプション（-T、-A、-a、-U、-L、-f、-D、-l、-t、-d、-r、-p、-g）は、TOPPERS/ASP コンフィギュレーションスクリプトと同じである。TOPPERS/ASP カーネルユーザーズマニュアル（user.txt）の「**５．コンフィギュレーションスクリプトの使い方**」を参照すること。その他のオプションについては、「**9.2 TINET コンフィギュレーションスクリプトのオプション**」を参照すること。

アプリケーションとカーネルを別々に構築する方法については、TOPPERS/ASP 環境におけるサンプルアプリケーションの構築と同じである。TOPPERS/ASP カーネルユーザーズマニュアル（user.txt）の「**3.5 アプリケーションとカーネルを別々に構築する方法**」を参照すること。

[2] TOPPERS/JSP 用 TINET コンフィギュレーションスクリプトの実行

```
$ mkdir NETOBJ
$ cd NETOBJ
$ perl ../tinet/tinet_jsp_configure -C h8 -S akih8_3069f -A echos6
                                    -i ether -v if_ed -n inet6 -s tcp
```

オプション（-C、-S、-T、-A、-U、-L、-D、-P、-p、-l）は、TOPPERS/JSP コンフィギュレーションスクリプトと同じである。TOPPERS/JSP カーネルユーザーズマニュアル（user.txt）の「**7.6 コンフィギュレーションスクリプトの使い方**」を参照すること。その他のオプションについては、「**TINET コンフィギュレーションスクリプトのオプション**」を参照すること。

アプリケーションとカーネルを別々に構築する方法については、TOPPERS/JSP 環境におけるサンプルアプリケーションの構築と同じである。TOPPERS/JSP カーネルユーザーズマニュアル（user.txt）の「**7.5 アプリケーションとカーネルを別々に構築する方法**」を参照すること。

上記の例で、TOPPERS/ASP 環境と TOPPERS/JSP 環境のいずれの場合も、ディレクトリ

NETOBJ に以下のファイルが生成される。

Makefile	Makefile
echos6.c	サンプルプログラム本体
echos6.h	サンプルプログラムのヘッダファイル
echos6.cfg	サンプルプログラム用 ASP コンフィギュレーションファイル
tinet_echos6.cfg	サンプルプログラム用 TINET コンフィギュレーションファイル
route_cfg.c	サンプルプログラム用ルーティング表
tinet_app_config.h	サンプルプログラム用コンパイル時指定コンフィギュレーション

必要であれば、Makefile を修正する。修正については、TOPPERS/ASP 環境では「**7.3 アプリケーションの Makefile**」、TOPPERS/JSP 環境では「**8.3 アプリケーションの Makefile**」を参照すること。

以下のサンプルアプリケーションでは、構築上の注意がある。各章を参照すること。

[1] 「**9.3** クライアントサーバ・セット」
[2] 「**9.4** TOPPERS ASP/JSP サンプルプログラム sample1 のネットワーク対応プログラム」
[3] 「**9.5** 最小構成サーバ」

(2) tinet_app_config.h の設定

IPv4 の場合、IP アドレス、サブネットマスク、ディフォルトゲートウェイを指定する。

[1] IPV4_ADDR_LOCAL
自分の IP アドレスを指定する。ただし、PPP を使用するとき、相手に割当ててもらう場合は 0 を指定すること。なお、PPP は参考実装である。

[2] IPV4_ADDR_REMOTE
相手の IP アドレスを指定する。ただし、PPP を使用するとき、相手に割当ててもらう場合は 0 を指定すること。なお、PPP は参考実装である。

[3] IPV4_ADDR_LOCAL_MASK
サブネットマスクを指定する。ただし、ネットワークインタフェースがイーサネットのとき有効である。

[4] IPV4_ADDR_DEFAULT_GW
ディフォルトゲートウェイを指定する。ただし、ネットワークインタフェースがイーサネットのとき有効である。

(3) route_cfg.c の設定

ネットワークインタフェースがイーサネットの場合は、ルーティング表 routing_tbl を設定する。ただし、ディフォルトゲートウェイのみのシンプルなネットワークでは、変更する必要はない。

(4) サンプルプログラムのコンパイル・リンク

TINET サンプルアプリケーションの構築用ディレクトリで、サンプルプログラムをコンパイル・リンクする。コンパイル・リンクの方法を以下に示す。

```
$ make depend
$ make
```

3. TINET コンフィギュレーションスクリプトのオプション

3. 1 TOPPERS/APS 用 TINET コンフィギュレーションスクリプトのオプション

TOPPERS/ASP 環境では、オプション（-T、-A、-a、-U、-L、-f、-D、-l、-d、-r、-p、-g）は、TOPPERS/ASP コンフィギュレーションスクリプトと同じである。TOPPERS/ASP 用 TINET コンフィギュレーションスクリプト特有のオプションを以下に示す。

-e <tinetdir>	TINET のソースの置かれているディレクトリ
-i <net_if>	ネットワークインタフェース（必須） <net_if> には ether、ppp、loop の何れかを指定する。
-v <net_dev>	イーサネット・ディバイスドライバ （ネットワークインタフェースに ether を指定した場合は必須）
-n <net_proto>	ネットワーク層プロトコル（必須） <net_proto> に指定できるプロトコルを以下に示す。

inet6	IPv6 のみ
inet4	IPv4 のみ
inet6/inet4	IPv6/IPv4 の両方（デュアルスタック）
inet6m	IPv6 のみで、IPv4 射影アドレス指定可能
inet6m/inet4	IPv6/IPv4 の両方と IPv4 射影アドレス指定可能

-s <trans_proto>	トランスポート層プロトコル（必須） <trans_proto> には tcp、udp、tcp/udp の何れかを指定する。
-t <templatedir>	TINET テンプレートディレクトリ デフォルトは tinet/asp_sample である。なお、TOPPERS/ASP の同じオプションのデフォルトは sample である。
-m <asptemplatedir>	ASP テンプレートディレクトリ デフォルトは sample である。

3. 2 TOPPERS/JPS 用 TINET コンフィギュレーションスクリプトのオプション

TOPPERS/JSP 環境では、オプション（-C、-S、-T、-A、-U、-L、-D、-P、-p、-l）は、TOPPERS/JSP コンフィギュレーションスクリプトと同じである。TOPPERS/ASP 用 TINET コンフィギュレーションスクリプト特有のオプションを以下に示す。

-e <tinetdir>	TINET のソースの置かれているディレクトリ
-i <net_if>	ネットワークインタフェース（必須） <net_if> には ether、ppp、loop の何れかを指定する。
-v <net_dev>	イーサネット・ディバイスドライバ （ネットワークインタフェースに ether を指定した場合は必須）
-n <net_proto>	ネットワーク層プロトコル（必須） <net_proto> に指定できるプロトコルを以下に示す。

inet6	IPv6 のみ
inet4	IPv4 のみ
inet6/inet4	IPv6/IPv4 の両方（デュアルスタック）
inet6m	IPv6 のみで、IPv4 射影アドレス指定可能

inet6m/inet4	IPv6/IPv4 の両方と IPv4 射影アドレス指定可能
-s <trans_proto>	トランスポート層プロトコル（必須） <trans_proto> には tcp、udp、tcp/udp の何れかを指定する。
-d <dir>	TINET テンプレートディレクトリ デフォルトは tinet/jsp_sample である。なお、TOPPERS/JSP の同じオプションのデフォルトは sample である。
-m <jsptemplatedir>	JSP テンプレートディレクトリ デフォルトは sample である。

4. クライアントサーバ・セットの構築

以下に示すサーバが提供されており、必要に応じて組み込むサーバを選択できる。（）内は、tinet/netapp 内にあるソースファイル名である。なお、ディフォルトで、ITRON TCP/IP API 仕様の TCP と UDP の拡張機能が組込まれているので、「**9.3.2 簡易コンソールのコマンド**」の wtw（wtw4）、wte（wte4）、wue、wtd コマンドで、それぞれのサーバ・タスクを起動する必要がある。

[1] WWW サーバ・タスク（wwws.c）
ルートページの他に、ネットワーク統計情報を表示する 2 ページから構成され、タスク数は最大 2 である。
[2] TCP エコーサーバ・タスク
tcp_echo_srv1.c と tcp_echo_srv2.c のどちらかを選択する。tcp_echo_srv1.c を選択した場合、省コピー API を使用して、ノンブロッキングコールを使用しない時は、タスク数を 8 まで指定可能である。
[3] UDP エコーサーバ・タスク（udp_echo_srv.c）
[4] TCP ディスカードサーバ・タスク（tcp_discard_srv.c）
[5] 簡易コンソール・タスク（dbg_cons.c）
シリアルインタフェースだけでなく、telnet プロトコルを使用して、ネットワーク経由で利用することも可能である。telnet で接続すると、シリアルの入出力を引き継いで実行する。切断すると、元のシリアルに入出力を戻す。ただし、TCP のノンブロッキングコールを組込む必要がある。

また、クライアントとしては以下の機能が提供されており、必要に応じて組み込むクライアントを選択できる。

[1] TCP エコークライアント・タスク（tcp_echo_cli.c）
[2] UDP エコークライアント・タスク（udp_echo_cli.c）
[3] TCP ディスカードクライアント・タスク（tcp_discard_cli.c）
[4] UDP ディスカードクライアント・タスク（udp_discard_cli.c）
[5] ping（ping.c）

4. 1 Makefile の設定

(1) 組込む機能の選択

[1] TCP の受信ウィンドバッファの省コピー機能
サンプルアプリケーションで、TCP の受信ウィンドバッファの省コピー機能を組込む場合は、

```
TCP_CFG_RWBUF_CSAVE_ONLY = true
```

または、

```
TCP_CFG_RWBUF_CSAVE = true
```

を選択する。TCP_CFG_RWBUF_CSAVE_ONLY を選択すると、TCP 通信端点を生成する静的 API である TCP_CRE_CEP で、受信ウィンドバッファ rbuf に、メモリアドレスを指定しても、プロトコルスタックは、この指定を無視して、TCP の受信ウィンドバッファの省コピー機能により処理する。また、TCP_CFG_RWBUF_CSAVE を選択すると、TCP_CRE_CEP で、受信ウィンドバッファ rbuf に、NADR（NULL）を指定したときのみ、プロトコルスタックは TCP の受信ウィンドバッファの省コピー機能により処理する。

[2] TCP の送信ウィンドバッファの省コピー機能

サンプルアプリケーションで、TCP の送信ウィンドバッファの省コピー機能を組込む場合は、

```
TCP_CFG_SWBUF_CSAVE_ONLY = true
```

または、

```
TCP_CFG_SWBUF_CSAVE = true
```

を選択する。TCP_CFG_SWBUF_CSAVE_ONLY を選択すると、TCP 通信端点を生成する静的 API である TCP_CRE_CEP で、送信ウィンドバッファ rbuf に、メモリアドレスを指定しても、プロトコルスタックは、この指定を無視して、TCP の送信ウィンドバッファの省コピー機能により処理する。また、TCP_CFG_SWBUF_CSAVE を選択すると、TCP_CRE_CEP で、送信ウィンドバッファ rbuf に、NADR（NULL）を指定したときのみ、プロトコルスタックは TCP の送信ウィンドバッファの省コピー機能により処理する。

[3] TCP のノンブロッキングコール

サンプルアプリケーションで、TCP のノンブロッキングコールを組込む場合は、

```
TCP_CFG_NON_BLOCKING = true
```

を選択する。

[4] UDP のノンブロッキングコール

サンプルアプリケーションで、UDP のノンブロッキングコールを組込む場合は、

```
UDP_CFG_NON_BLOCKING = true
```

を選択する。

[5] ITRON TCP/IP API 仕様の TCP の拡張機能

サンプルアプリケーションで、ITRON TCP/IP API 仕様の TCP の拡張機能を組込む場合は、

```
TCP_CFG_EXTENTIONS = true
```

を選択する。ディフォルトで true に設定されている。

[6] ITRON TCP/IP API 仕様の UDP の拡張機能

サンプルアプリケーションで、ITRON TCP/IP API 仕様の UDP の拡張機能を組込む場合は、

```
UDP_CFG_EXTENTIONS = true
```

を選択する。ディフォルトで true に設定されている。

(2) ノンブロッキングコール、ITRON TCP/IP API 仕様の拡張機能と省コピー API の選択

[1] TCP のノンブロッキングコール

サンプルアプリケーションで、TCP のノンブロッキングコールを使用する場合は、

```
USE_TCP_NON_BLOCKING = true
```

を選択する。

[2] ITRON TCP/IP API 仕様の TCP の拡張機能

サンプルアプリケーションで、ITRON TCP/IP API 仕様の TCP の拡張機能を使用する場合は、

```
USE_TCP_EXTENTIONS = true
```

を選択する。ディフォルトで true に設定されている。

[3] 省コピー API

サンプルアプリケーションで、省コピー API を使用する場合は、

```
USE_COPYSAVE_API = true
```

を選択する。

[4] UDP のノンブロッキングコール

サンプルアプリケーションで、UDP のノンブロッキングコールを使用する場合は、

```
USE_UDP_NON_BLOCKING = true
```

を選択する。ただし、コールバックとは同時に使用できない。

[5] ITRON TCP/IP API 仕様の UDP の拡張機能

サンプルアプリケーションで、ITRON TCP/IP API 仕様の UDP の拡張機能を使用する場合は、

```
USE_UDP_EXTENTIONS = true
```

を選択する。ディフォルトで true に設定されている。

[6] UDP のコールバック

サンプルアプリケーションで、UDP のコールバックを使用する場合は、

```
USE_UDP_CALL_BACK = true
```

を選択する。ただし、ノンブロッキングコールとは同時に使用できない。

(3) 共通サーバプログラムの選択

[1] WWW サーバプログラムを使用する場合は、以下の行を有効にする。

```
USE_WWW_SRV = ture
```

[2] WWW サーバプログラム（IPv4 優先接続待ち）を使用する場合は、以下の行を有効にする。

```
USE_WWW4_SRV = ture
```

[3] UDP エコーサーバプログラムを使用する場合は、以下の行を有効にする。

```
USE_UDP_ECHO_SRV = ture
```

[4] UDP エコーサーバプログラム（IPv4 優先接続待ち）を使用する場合は、以下の行を有効にする。

```
USE_UDP4_ECHO_SRV = ture
```

[5] TCP ディスカードサーバプログラムを使用する場合は、以下の行を有効にする。

```
USE_TCP_DISCARD_SRV = ture
```

[6] 送受信タスク同一型の TCP エコーサーバプログラムを使用する場合は、以下の行を有効にする。

```
TCP_ECHO_SRV = tcp_echo_srv1
```

[7] 送受信タスク分離型の TCP エコーサーバプログラムを使用する場合は、以下の行を有効にする。

```
TCP_ECHO_SRV = tcp_echo_srv2
```

[8] TCP エコーサーバプログラム（IPv4 優先接続待ち）を使用する場合は、以下の行を有効にする。ただし、送受信タスク同一型の TCP エコーサーバプログラムを使用する場合のみ有効である。

```
USE_TCP4_ECHO_SRV = ture
```

[9] シリアル経由のみコンソール入出力を使用する場合は、以下の行を有効にする。

```
USE_DBG_CONS = true
```

[10] シリアルとネットワーク経由のコンソール入出力を使用する場合は、以下の行を有効にする。

```
USE_NET_CONS = true
```

ただし、ノンブロッキングコールを組込んだ時のみ有効である。

(4) 共通クライアントプログラムの選択

[1] TCP エコークライアントプログラムを使用する場合は、以下の行を有効にする。

```
USE_TCP_ECHO_CLI = ture
```

[2] TCP エコークライアントプログラムで、IPv4 アドレスを指定した時、tcp_con_cep を呼出し、IPv4 により接続する場合は以下の行を有効にする。なお、ネットワーク層の選択で、IPv6 と IPv4 の両方を指定した場合、この指定を行わないと tcp_con_cep ではなく、tcp6_con_cep を呼出す。従って、API_CFG_IP4MAPPED_ADDR を指定していないと、

E_PAR が返されるが、tcp6_con_cep のテストのためであり、問題はない。

```
USE_TCP4_ECHO_CLI = ture
```

[3] UDP エコークライアントプログラムを使用する場合は、以下の行を有効にする。

```
USE_UDP_ECHO_CLI = ture
```

[4] UDP エコークライアントプログラム（IPv4 優先接続待ち）を使用する場合は、以下の行を有効にする。

```
USE_UDP4_ECHO_CLI = ture
```

[5] TCP ディスカードクライアントプログラムを使用する場合は、以下の行を有効にする。

```
USE_TCP_DISCARD_CLI = ture
```

[6] UDP ディスカードクライアントプログラムを使用する場合は、以下の行を有効にする。

```
USE_UDP_DISCARD_CLI = ture
```

[7] DHCPv6 クライアントプログラムを使用する場合は、以下の行を有効にする。

```
USE_DHCP6_CLI = true
```

[8] DHCPv4 クライアントプログラムを使用する場合は、以下の行を有効にする。

```
USE_DHCP4_CLI = true
```

[9] DNS リゾルバーを使用する場合は、以下の行を有効にする。

```
USE_RESOLVER = true
```

[10] PING クライアントプログラムを使用する場合は、以下の行を有効にする。

```
USE_PING = true
```

(5) 共通サーバタスク数の選択

[1] WWW サーバタスク数は以下の行で選択する。ただし最大 2 タスクである。

```
CDEFS := $(CDEFS) -DNUM_WWW_SRV_TASKS=2
```

[2] TCP ECHO サーバタスク数を選択する。ただし以下の条件のとき有効である。

・tcp_echo_srv1.c を選択した。
・省コピー API を使用する。
・ノンブロッキングコールを使用しない。

TCP ECHO サーバタスク数は以下の行で選択する。ただし最大 8 タスクである。

```
CDEFS := $(CDEFS) -DNUM_TCP_ECHO_SRV_TASKS=8
```

(6) 予約 ID 数の選択

[1] TCP/IPv4 受付口予約 ID 数は以下の行で選択する。ただし最大 2 である。

```
CDEFS := $(CDEFS) -DNUM_VRID_TCP_REPS=2
```

[2] TCP/IPv4 通信端点予約 ID 数は以下の行で選択する。ただし最大 4 である。

```
CDEFS := $(CDEFS) -DNUM_VRID_TCP_CEPS=4
```

[3] UDP/IPv4 通信端点予約 ID 数は以下の行で選択する。ただし最大 2 である。

```
CDEFS := $(CDEFS) -DNUM_VRID_UDP_CEPS=2
```

[4] TCP/IPv6 受付口予約 ID 数は以下の行で選択する。ただし最大 2 である。

```
CDEFS := $(CDEFS) -DNUM_VRID_TCP6_REPS=2
```

[5] TCP/IPv6 通信端点予約 ID 数は以下の行で選択する。ただし最大 4 である。

```
CDEFS := $(CDEFS) -DNUM_VRID_TCP6_CEPS=4
```

[6] UDP/IPv6 通信端点予約 ID 数は以下の行で選択する。ただし最大 2 である。

```
CDEFS := $(CDEFS) -DNUM_VRID_UDP6_CEPS=2
```

(7) その他

[1] TCP のセグメントサイズを MSS にする場合は、以下の行を有効にする。

```
CDEFS := $(CDEFS) -DUSE_TCP_MSS_SEG
```

[2] IPv6 MMTU サイズのネットワークバッファを組込む場合は、以下の行を有効にする。

```
CDEFS := $(CDEFS) -DUSE_IPV6_MMTU
```

4.2 サンプルアプリケーションのコンパイル時コンフィギュレーション

(1) DHCPv6 クライアント

[1] DHCP6_CLI_CFG_MODE
DHCPv6 クライアントの動作モードを指定する。
ステートレスモードを選択する時は fDHCP6_CLI_CFG_STATELESS、ステートフルモードを選択する時は fDHCP6_CLI_CFG_STATEFULL を指定する。ステートレスモードでは、IPv6 アドレスを取得せず、DNS サーバの情報などのみ取得する。

[2] DHCP6_CLI_CFG_REQUIRED_OLIST
DHCPv6 サーバから取得する必須オプションリストを配列の初期化形式で指定する。指定例を以下に示す。

```
#define DHCP6_CLI_CFG_REQUIRED_OLIST {   \
        DHCP6_OPT_NAME_SERVERS,          \
        }
```

オプションは tinet/netapp/dhcp6.h を参照すること。

[3] DHCP6_CLI_CFG_REQUEST_OLIST
DHCPv6 サーバから取得する必須オプションリストを配列の初期化形式で指定する。指定例を以下に示す。

```
#define DHCP6_CLI_CFG_REQUEST_OLIST {    \
        DHCP6_OPT_DOMAIN_SEARCH,         \
        }
```

オプションは tinet/netapp/dhcp6.h を参照すること。

(2) DHCPv4 クライアント

[1] DHCP4_CLI_CFG_REQUIRED_OLIST

DHCPv4 サーバから取得する必須オプションリストを配列の初期化形式で指定する。指定例を以下に示す。

```
#define DHCP4_CLI_CFG_REQUIRED_OLIST {  \
        DHCP4_OPT_NAME_SERVERS,         \
        }
```

オプションは tinet/netapp/dhcp4.h を参照すること。

[2] DHCP4_CLI_CFG_REQUEST_OLIST

DHCPv4 サーバから取得する必須オプションリストを配列の初期化形式で指定する。指定例を以下に示す。

```
#define DHCP4_CLI_CFG_REQUEST_OLIST {   \
        DHCP4_OPT_DOMAIN_SEARCH,        \
        }
```

オプションは tinet/netapp/dhcp6.h を参照すること。

(3) DNS リゾルバー

[1] RSLV_CFG_DNS_DOMAIN_NAME_STR

ドメイン名の文字列を指定する。DHCPv6 クライアント、または DHCPv4 クライアントにより取得することも可能である。

[2] IPV6_ADDR_DNS_INIT

DNS サーバの IPv6 アドレスを配列の初期化形式で指定する。DHCPv6 クライアントにより取得することも可能である。DNS サーバの IPv6 アドレスが、
fd90:cce5:25f6:ff81:201:2ff:fe81:e7c9 の場合の指定例を以下に示す。

```
#define IPV6_ADDR_DNS_INIT \
    {{{ UINT_C(0xfd), UINT_C(0x90), UINT_C(0xcc), UINT_C(0xe5), \
        UINT_C(0x25), UINT_C(0xf6), UINT_C(0xff), UINT_C(0x81), \
        UINT_C(0x02), UINT_C(0x01), UINT_C(0x02), UINT_C(0xff), \
        UINT_C(0xfe), UINT_C(0x81), UINT_C(0xe7), UINT_C(0xc9) }}}
```

[3] IPV4_ADDR_DNS

DNS サーバの IPv4 アドレスを指定する。DHCPv4 クライアントにより取得することも可能である。DNS サーバの IPv4 アドレスが、172.25.129.140 の場合の指定例を以下に示す。

```
#define IPV4_ADDR_DNS   MAKE_IPV4_ADDR(172,25,129,140)
```

4. 3 簡易コンソールコマンド

インターネットサーバ・セットに組み込まれている簡易コンソールのコマンドを以下に示す。

cf　　tinet_app_config.h 等で指定された、コンパイル時コンフィギュレーションを表示する。

ct <cepid> [<fncd>]

ペンディングしている TCP 通信端点 <cepid> の処理をキャンセルする。キャ

ンセルする処理は <fncd> で指定する。<fncd> を省略した場合は、全ての処理をキャンセルする。

`cu <cepid> [<fncd>]`
: ペンディングしている UDP 通信端点 <cepid> の処理をキャンセルする。キャンセルする処理は <fncd> で指定する。<fncd> を省略した場合は、全ての処理をキャンセルする。

`dc`
: ネットワークインタフェースが PPP の時、または、シリアルとネットワーク経由のコンソール入出力を使用する時に有効であり、接続を切断する。なお、PPP は参考実装である。

`dh`
: DHCPv6 と DHCPv4 クライアントの情報を出力する。

`dh6`
: DHCPv6 クライアントの情報を出力する。

`dh6n`
: IPv6 アドレス情報を解放して、DHCPv6 サーバから IPv6 アドレス情報を再取得する。

`dh6r`
: IPv6 アドレス情報を解放する。

`dh4`
: DHCPv4 クライアントの情報を出力する。

`dh4n`
: IPv4 アドレス情報を解放して、DHCPv4 サーバから IPv4 アドレス情報を再取得する。

`dh4r`
: IPv4 アドレス情報を解放する。

`dt <host> [<portno> [<repeat>]]`
: TCP ディスカードクライアント・タスクを起動し、ディスカードサーバ <host> にディスカードパターンを送信する。<portno> は、ディスカードサーバのポート番号で、省略時（- を指定する）は 9 である。<repeat> は、繰り返し回数で、省略時は 1 である。

`dts`
: TCP ディスカードクライアント・タスクの繰り返し動作を停止する。

`du <host> [<portno>]]`
: UDP ディスカードクライアント・タスクを起動し、ディスカードサーバ <host> にディスカードパターンを繰り返し送信する。<portno> は、ディスカードサーバのポート番号で、省略時は 9 である。

`dus`
: UDP ディスカードクライアント・タスクの繰り返し動作を停止する。

`et <host> [<portno> [<repeat>]]`
`et4 <host> [<portno> [<repeat>]]`
: et では TCP エコークライアント・タスクを起動する。et4 では IPv4 専用 TCP エコークライアント・タスクを起動する。何れも、エコーサーバ <host> にエコーパターンを送信する。<portno> は、エコーサーバのポート番号で、省略時（- を指定する）は 7 である。<repeat> は、繰り返し回数で、省略時は 1 である。

`ets`
: TCP エコークライアント・タスクの繰り返し動作を停止する。

`ets4`
: IPv4 専用 TCP エコークライアント・タスクの繰り返し動作を停止する。

eu <host> [<portno>] [<msg> | <repeat>]
eu4 <host> [<portno>] [<msg> | <repeat>]

eu では UDP エコークライアント・タスクを起動する。eu4 では IPv4 専用 UDP エコークライアント・タスクを起動する。何れも、エコーサーバ <host> にエコーパターンを送信する。<portno> は、エコーサーバのポート番号で、省略時（- を指定する）は 7 である。<repeat>（数字）を指定した場合は、定型のメッセージを <repeat> 回繰り返し送信する。<msg>（数字以外）を指定した場合は、メッセージ <msg> を送信する。

eus — UDP エコークライアント・タスクの繰り返し動作を停止する。

eus4 — IPv4 専用 UDP エコークライアント・タスクの繰り返し動作を停止する。

i — ネットワークインタフェースが PPP の時に有効である。直接接続の場合は、直ちに LCP を起動して、サーバに接続する。モデム接続の場合は、コンパイル時コンフィギュレーションの MODEM_CFG_PHONE_NUMBER パラメータで指定されているサーバに発呼する。なお、PPP は参考実装である。

if [<addr> <mask>]

ネットワークインタフェースが PPP の時は、IP アドレス、サブネットマスク、ブロードキャストアドレスを出力する。ネットワークインタフェースがイーサネットで、ネットワーク層が IPv4 の時は、[<addr> <mask>] を指定できる。[<addr> <mask>] を指定しなければ、MAC アドレス、IPv4 アドレス、サブネットマスク、ブロードキャストアドレスの出力のみ行う。[<addr> <mask>] を指定した時は、IPv4 アドレスとサブネットマスクを変更した後、MAC アドレス、IPv4 アドレス、サブネットマスク、ブロードキャストアドレスを出力する。<addr> は、IPv4 アドレス、<mask> は、サブネットマスクである。なお、PPP は参考実装である。

na — ネットワークインタフェースがイーサネットの時に有効である。IPv6 では近隣アドレスキャッシュの状態、IPv4 では ARP キャッシュを出力する。

nb — ネットワークバッファの統計情報を出力する。

nc — ネットワーク統計情報を表示する。

nr [<index> <target> <mask> <gateway>]

ネットワークインタフェースがイーサネットのとき有効である。ネットワーク層が IPv4 の時は、[<index> <target> <mask> <gateway>] を指定できる。[<index> <target> <mask> <gateway>] を指定しなければ、ルーティング表の出力のみ行う。[<index> <target> <mask> <gateway>] を指定した時は、ルーティング表を変更した後、ルーティング表を出力する。<index> は、経路エントリのインデックス、<target> は、目標ネットワークの IP アドレス、<mask> は、目標ネットワークのサブネットマスク、<gateway> は、ゲートウェイの IP アドレスである。

nrl — ネットワークインタフェースがイーサネットで、ネットワーク層が IPv6 の時に有効である。ディフォルトルータ・リストを出力する。

nrp　ネットワークインタフェースがイーサネットで、ネットワーク層が IPv6 の時に有効である。プレフィックスリストを出力する。

ns [<name>|<addr>]
DNS サーバに正引きでホスト名 <name>、または、逆引きで IP アドレス <addr> のアドレス情報を照会し出力する。いずれも指定しない時は、ドメイン名と DNS サーバの IP アドレスを出力する。

nt　TCP 通信端点と TCP 受付口の状態を表示する。

nu　UDP 通信端点の状態を表示する。

p <host> [<tmo> [<size>]]
ホスト <host> に ICMP パケットを送信する。<tmo> はタイムアウト値（単位は秒）で、省略時（- を指定する）は 3 秒である。<size> はデータサイズで、指定しない場合は 64 オクテットである。

ps　タスクの状態を表示する。

r <tskid>　タスク <tskid> の待ち状態を強制的に解除する。

tt <repid>　ITRON TCP/IP API の TCP の拡張機能を組込む必要がある。TCP 受付口 <repid> を削除し、対応するサーバを停止する。

tu <cepid>　ITRON TCP/IP API の UDP の拡張機能を組込む必要がある。UDP 通信端点 <cepid> を削除し、対応するサーバを停止する。

w <tskid>　タスク <tskid> を起床する。

wtd　ITRON TCP/IP API の TCP の拡張機能を組込む必要がある。TCP ディスカードサーバ・タスクに TCP 受付口と TCP 通信端点を割当て、TCP ディスカードサーバ・タスクを起動する。

wte　ITRON TCP/IP API の TCP の拡張機能を組込む必要がある。TCP エコーサーバ・タスクに TCP 受付口と TCP 通信端点を割当て、TCP エコーサーバ・タスクを起動する。

wte4　ITRON TCP/IP API の TCP の拡張機能を組込む必要がある。TCP エコーサーバ・タスクに IPv4 用 TCP 受付口と IPv4 用 TCP 通信端点を割当て、TCP エコーサーバ・タスクを起動する。

wtw4　ITRON TCP/IP API の TCP の拡張機能を組込む必要がある。WWW サーバ・タスクに IPv4 用 TCP 受付口と IPv4 用 TCP 通信端点を割当て、WWW サーバ・タスクを起動する。

wtw　ITRON TCP/IP API の TCP の拡張機能を組込む必要がある。WWW サーバ・タスクに TCP 受付口と TCP 通信端点を割当て、WWW サーバ・タスクを起動する。

wue	ITRON UDP/IP API の UDP の拡張機能を組込む必要がある。UDP エコーサーバ・タスクに UDP 通信端点を割当て、UDP エコーサーバ・タスクを起動する。

4.4 DNSリゾルバーによるホスト名の指定

Makefile の組み込む機能の選択で、

```
USE_RESOLVER = true
```

を指定して DNS リゾルバーを使用すると、簡易コンソールコマンドの <host> に指定したホスト名を、DNS サーバに照会して IP アドレスを得ることができる（正引き）。

<host> の書式を以下に示す。

```
[-[6|4][A|a|Q|q]] <name>
```

各オプションを以下に示す。

6	ネットワーク層のプロトコルとして IPv6 により DNS サーバに照会する。
4	ネットワーク層のプロトコルとして IPv4 により DNS サーバに照会する。
q	AAAA レコードのみ照会する。
Q	AAAA レコードのみ照会する。
a	A レコードのみ照会する。
A	A レコードのみ照会する。

6 と 4 のどちらも指定しない時は、まず IPv6 で DNS サーバに照会するが、応答がない場合は IPv4 にフォールバックして DNS サーバに照会する。ただし、ネットワーク層としてどちらか一方のみ組込んだ場合は、このオプションは無視される。

q、Q、a、A のいずれも指定しない時は、AAAA レコードの次に A レコードの順で DNS サーバに照会する。

5. sample1 のネットワーク対応プログラム（sample1n）の構築

TOPPERS/ASP と TOPPERS/JSP のサンプルプログラム sample1 のネットワーク対応プログラムである。telnet で接続すると、シリアルの入出力を引き継いで実行する。切断すると、元のシリアルに入出力を戻す。

以下に構築方法を述べる。

(1) ASP/JSP コンフィギュレーションスクリプトの実行

それぞれの環境におけるコンフィギュレーションスクリプトを実行する。以下は、TOPPERS/ASP 環境におけるコンフィギュレーションスクリプトの実行例である。

```
$ mkdir NETOBJ
$ cd NETOBJ
$ perl ../configure -T akih8_3069f_gcc
```

(2) TINET コンフィギュレーションスクリプトの実行

それぞれの環境における TINET コンフィギュレーションスクリプトを実行する。この時、アプリケーションプログラム名として sample1n を指定する。以下は、TOPPERS/ASP 環境における TINET コンフィギュレーションスクリプトの実行例である。

```
$ perl ../tinet/tinet_asp_configure -T akih8_3069f_gcc -A sample1n
                                    -i ether -v if_ed -n inet6 -s tcp
```

なお、オプション -n に inet6/inet4 を指定しても、IPv6 でのみ接続できる。

(3) Makefile の修正

アプリケーション本体（sample1n.c）と TOPPERS/ASP と TOPPERS/JSP の sample1.c を一緒にコンパイル・リンクするため、Makefile を修正する。

[1] TOPPERS/ASP 環境

Makefile の APPL_COBJS に sample1.o を追加する。

```
APPL_COBJS = $(APPLNAME).o sample1.o
```

[2] TOPPERS/JSP 環境

Makefile の UTASK_COBJS に sample1.o を追加する。

```
UTASK_COBJS = $(UNAME).o sample1.o
```

(4) sample1.c の修正

TOPPERS/ASP と TOPPERS/JSP の sample1.c のインクルードファイルの指定

```
#include "sample1.h"
```

の前に、以下のインクルードファイルを追加する。

```
#include "sample1n.h"
```

(5) tinet_app_config.h の設定

IPv4 の場合、IP アドレス、サブネットマスク、ディフォルトゲートウェイを指定する。

6. 最小構成サーバの構築

WWW サーバ・タスクと TCP エコーサーバ・タスクのみからなる最小構成のサーバである。H8/3069F が内蔵している RAM（16K バイト）と ROM（512K バイト）に収まり、外部メモリは不要である。現在は、品川通信計装サービス製 NKEV-010H8（TOPPERS/JSP リリース 1.4.2 のみ）と秋月電子通商製 H8/3069F（TOPPERS/JSP リリース 1.4.1 以降と TOPPERS/ASP）のシステムに対応している。

各システム依存部の Makefile.config の「実行環境の定義」で、

```
# ROM化 外部RAM未使用
#DBGENV := INMEM_ONLY
```

を有効にして、コンパイル・リンクする。