On-Chip Embedded Network Solution

ミドルウェア編

ユーザーズガイド

MiSPO

株式会社ミスポ

目次

第 1 章 導入

1.1. はじめに

NORTi Oceans は、μITRON4.0 仕様サブセットのリアルタイム OS、ITRON TCP/IP API 仕様サブセットの TCP/IP プロトコルスタックとミドルウェアから構成される商品です。
本マニュアルは、NORTi Oceans のミドルウェアのユーザーズガイドです。ミドルウェアの概要、コンフィグレーションの方法や API 仕様について記載しています。
本製品のリアルタイム OS については、「NORTi Oceans ユーザーズガイド カーネル編」を参照してください。TCP/IP プロトコルスタックについては、「NORTi Oceans ユーザーズガイド TCP/IP 編」を参照してください。

1.2. 特徴

NORTi Oceans のミドウルェアは、TELNET サーバ/クライアント、FTP サーバ/クライアント,TFTP サーバ/クライアント、ファイルシステム、DHCP クライアント、DNS クライアント、SNTP クライアント、ping や ECHO サーバを提供します。
これらは動作確認済みのライブラリとして提供されますが、ミドルウェアのソースコードが付属しますので、デバッガでミドルウェア内部の動作を追跡することも可能です。

1.3. 制限事項

NORTi Oceans の下で動作するよう特化しているため、他の OS へ移植することは保証対象外です。

1.4. ファイル構成

NORTi Oceans ミドルウェアのファイル構成は次の通りです。

```
NORTi¥NORTiOC¥(CPU)¥INC¥
    nocdhcp.h       DHCP クライアント ヘッダ
    nocdns.h        DNS クライアント ヘッダ
    nocftp.h        FTP クライアント ヘッダ
    nocsntp.h       SNTP クライアント ヘッダ
    sntptime.h      日付/時間 ヘッダ

NORTi¥NORTiOC¥(CPU)¥SRC¥
    nocdhcp.c       DHCP クライアント ソース
    nocdns.c        DNS クライアント ソース
    nocftp.c        FTP クライアント ソース
    nocsntp.c       SNTP クライアント ソース
    sntptime.c      日付/時間 ソース

NORTi¥NORTiOC¥(CPU)¥LIB¥(CCC)¥
    nocxxxx.a(lib)  NORTi Oceans ライブラリ
```

フォルダ名の(CPU)は CPU 名略称、(CCC)はコンパイラ名略称を示します。NORTi Oceans はカーネル、TCP/IP プロトコルスタックやミドルウェアの単位で複数のライブラリを提供せず、NORTi Oceans として全てを含んだ一つのライブラリを提供します。

第 2 章 ミドルウェア概要

NORTi Oceans は、よく使われる機能をまとめ、次のミドルウェアを提供しています。

サーバ系
　TELNET サーバ、FTP サーバ、TFTP サーバ、TCP ECHO サーバ
クライアント系
　TELNET、FTP、TFTP、DHCP、SNTP、ping
その他
　ファイルシステム、RAM ディスク

これらのミドルウェアは、タスクなどのカーネルリソースや通信端点などの TCP/IP プロトコルスタックのリソースを必要とします。しかし、コンフィグレーションする時にはミドルウェアが必要とするリソース数は自動的に確保されませんので、ユーザーが必要なリソースを確保してください。

第 3 章 DHCP クライアント

NORTi DHCP クライアントは、IP アドレス、サブネットマスク、デフォルトゲートウェイや DNS サーバのアドレスなどのネットワークの設定を自動的に DHCP サーバから取得します。

3.1. ファイル構成

INC¥nocdhcp.h … DHCP クライアントのヘッダファイル
SRC¥nocdhcp.c … DHCP クライアントのソースファイル

DHCP クライアント機能の API を発行するユーザープログラムで、nocdhcp.h をインクルードしてください。

nocdhcp.c はコンパイルされて、NORTi Oceans のライブラリに結合されていますので、カスタマイズをされる場合にだけ、nocdhcp.c をプロジェクトに加えてビルドしてください。(ライブラリ内の nocdhcp モジュールに優先してリンクされます)

3.2. 使用するリソース

タスク	1
セマフォ	0
イベントフラグ	0
メールボックス	0
固定長メモリプール	0
データキュー	0
周期ハンドラ	0
TCP 受付口	0
TCP 通信端点	0
UDP 通信端点	1

3.3. 注意事項

これらの API が複数のタスクで同時に発行されることを考慮した排他制御は、省略されています。待ち状態の発生しない API(注 1)については、複数のタスクで発行することが可能ですが、各タスクの優先度を同じにして、同時となるのを抑制してください。

複数のクライアントを設ける場合は、待ち状態の発生する API であっても、対象クライアントが異なるならば、別々のタスクから同時に発行することが可能ですが、やはり、タスクの優先度は同じとしてください。

(注 1) dhcp_select()、dhcp_option()

3.4. API

dhcp_ini

機能　DHCP クライアント初期化

形式　ER dhcp_ini(T_DHCP *dhcp);

引数　dhcp　DHCP クライアント管理ブロックへのポインタ

解説　DHCP クライアント用の変数領域(管理ブロック)を、T_DHCP 構造体としてユーザープログラム側に定義し、その先頭アドレスを dhcp に指定して最初に一度だけ呼び出してください。

戻値　E_OK　正常終了

dhcps_ini

機能　　複数の DHCP クライアント初期化

形式　　ER dhcps_ini(T_DHCP *dhcp, INT n);

引数　　dhcp　　DHCP クライアント管理ブロックへのポインタ
　　n　　DHCP クライアント数（1～）

解説　　DHCP クライアントを複数設けて、別々の DHCP サーバに同時に接続することができます。
DHCP クライアントの数分の変数領域(管理ブロック)を、T_DHCP 構造体の配列としてユーザープログラム側に定義し、その先頭アドレスを dhcp に、配列の要素数を n に指定して、最初に一度だけ呼び出してください。
本 API で初期化する場合には、dhcp_ini()を使用しないでください。n に 1 を指定した場合には、dhcp_ini()と同じになります。

戻値　　E_OK　　正常終了
　　E_PAR　　DHCP クライアント数が 0 以下

dhcp_select

機能　DHCP クライアントの選択

形式　ER dhcp_select(INT i);

引数　i　DHCP クライアント番号 (0～)

解説　dhcps_ini()で複数クライアントを初期化した場合は、続く API の対象クライアントを、本 API で選択してください。dhcp_ini()で初期化した場合は、本 API を発行する必要はありません。
DHCP クライアント毎にユーザータスクを設けて、その各先頭で、本 API が発行されることを想定していますが、dhcp_select()を適宜発行することで、1 つのユーザータスクで、複数のクライアントを切り替えながら使用することもできます。

戻値　E_OK　正常終了
E_PAR　DHCP クライアント番号が範囲外

dhcp_option

機能　DHCP オプションの設定

形式　ER dhcp_option(INT optname, VP_INT optval);

引数　optname　オプションの種別
　　optval　オプションの値

解説　次の optname と optval 組み合わせで、オプションを 1 つずつ設定できます。

optname	optval (型)
OPT_DHCP_NIF	ネットワークインタフェース (T_NIF *型)
OPT_DHCP_CALLBACK	エラーを通知するコールバック関数 (DHCP_CALLBACK 型)

dhcp_request()の前に実行してください。dhcp_request()実行後は、動作を保証できません。

戻値　E_OK　正常終了
　　E_NOSPT　未サポートのオプション
　　E_OBJ　リクエスト済み、または、DHCP クライアントが未選択

dhcp_request

機能　　DHCP クライアントの開始

形式　　ER dhcp_request(void);

引数　　なし

解説　　DHCP サーバを探索し、リクエストを行います。
また、リース管理用の DHCP リクエストタスクを開始します。

戻値　　E_OK　　正常終了
E_OBJ　　DHCP クライアントが未選択
その他　　内部で発行している TCP/IP API のエラー

コールバック

機能　　エラーを通知するコールバック関数

形式　　void callback(ER ercd);

引数　　ercd　　エラーコード

解説　　DHCP リクエストタスクで発生したエラーを通知します。

戻値　　なし

3.5. コンフィグレーション

DHCP クライアントは、下記のコンフィグレーションを変更することができます。コンフィグレーションを行う場合は、noccfg.h をインクルードする前に各マクロを定義してください。

DHCP クライアントの受信のタイムアウト

DHCP クライアントの受信の基準となるタイムアウト時間は、デフォルトで 4 秒ですが、DHCP_TMOUT マクロにより変更できます。タイムアウト時間の単位は、カーネルのタイマの割り込み周期(Tick)ですので、4000/MSEC のように指定してください。

リトライ回数

受信がタイムアウトした場合のリトライ回数はデフォルトで 3 回ですが、DHCP_RETRY マクロにより変更できます。

第 4 章 DNS クライアント

NORTi DNS クライアントは、インターネット上のホスト名であるドメイン名に対応する IP アドレスを DNS サーバから取得します。

4.1. ファイル構成

INC¥nocdns.h … DNS クライアントのヘッダファイル
SRC¥nocdns.c … DNS クライアントのソースファイル

DNS クライアント機能の API を発行するユーザープログラムで、nocdns.h をインクルードしてください。
nocdns.c はコンパイルされて、NORTi Oceans のライブラリに結合されていますので、カスタマイズをされる場合にだけ、nocdns.c をプロジェクトに加えてビルドしてください。(ライブラリ内の nocdns モジュールに優先してリンクされます)

4.2. 使用するリソース

タスク	0
セマフォ	0
イベントフラグ	0
メールボックス	0
固定長メモリプール	0
データキュー	0
周期ハンドラ	0
TCP 受付口	0
TCP 通信端点	0
UDP 通信端点	1

4.3. 注意事項

これらの API が複数のタスクで同時に発行されることを考慮した排他制御は、省略されています。待ち状態の発生しない API(注 1)については、複数のタスクで発行することが可能ですが、各タスクの優先度を同じにして、同時となるのを抑制してください。
複数のクライアントを設ける場合は、待ち状態の発生する API であっても、対象クライアントが異なるならば、別々のタスクから同時に発行することが可能ですが、やはり、タスクの優先度は同じとしてください。

(注 1) dns_select()、dns_option()

4.4. API

dns_ini

機能　DNS クライアント初期化

形式　ER dns_ini(T_DNS *dns);

引数　dns　DNS クライアント管理ブロックへのポインタ

解説　DNS クライアント用の変数領域(管理ブロック)を、T_DNS 構造体としてユーザープログラム側に定義し、その先頭アドレスを dns に指定して最初に一度だけ呼び出してください。

戻値　E_OK　正常終了

dnss_ini

機能　　複数の DNS クライアント初期化

形式　　ER dnss_ini(T_DNS *dns, INT n);

引数　　dns　　DNS クライアント管理ブロックへのポインタ
　　n　　DNS クライアント数 (1～)

解説　　DNS クライアントを複数設けて、別々の DNS サーバに同時に接続することができます。
DNS クライアントの数分の変数領域(管理ブロック)を、T_DNS 構造体の配列としてユーザープログラム側に定義し、その先頭アドレスを dns に、配列の要素数を n に指定して、最初に一度だけ呼び出してください。
本 API で初期化する場合には、dns_ini()を使用しないでください。n に 1 を指定した場合には、dns_ini()と同じになります。

戻値　　E_OK　　正常終了
　　E_PAR　　DNS クライアント数が 0 以下

dns_select

機能　DNS クライアントの選択

形式　ER dns_select(INT i);

引数　i　DNS クライアント番号 (0～)

解説　dnss_ini()で複数クライアントを初期化した場合は、続く API の対象クライアントを、本 API で選択してください。dns_ini()で初期化した場合は、本 API を発行する必要はありません。
DNS クライアント毎にユーザータスクを設けて、その各先頭で、本 API が発行されることを想定していますが、dns_select()を適宜発行することで、1 つのユーザータスクで、複数のクライアントを切り替えながら使用することもできます。

戻値　E_OK　正常終了
E_PAR　DNS クライアント番号が範囲外

dns_option

機能　DNS オプションの設定

形式　ER dns_option(INT optname, VP_INT optval);

引数　optname オプションの種別
optval オプションの値

解説　次の optname と optval 組み合わせで、オプションを 1 つずつ設定できます。

optname	optval (型)
OPT_DNS_NIF	ネットワークインタフェース (T_NIF *型)

dns_resolve()後でもネットワークインタフェースを設定し、dns_relolve()を再発行できます。

戻値　E_OK 正常終了
E_NOSPT 未サポートのオプション
E_OBJ DNS クライアントが未選択

dns_resolve

機能　名前解決

形式　ER dns_resolve(UW dns_ip, const char *name, T_IPEP *host);

引数　dns_ip　DNS サーバの IP アドレス
　　name　ドメイン名
　　host　IP アドレスを格納する構造体ポインタ

解説　指定したドメイン名に対する IP アドレスを DNS サーバから取得します。

戻値　E_OK　正常終了
　　E_OBJ　DNS クライアントが未選択
　　その他　内部で発行している TCP/IP API のエラー

4.5. コンフィグレーション

DNS クライアントは、下記のコンフィグレーションを変更することができます。コンフィグレーションを行う場合は、noccfg.h をインクルードする前に各マクロを定義してください。

DNS 要求の再送間隔

DNS 要求の再送間隔のデフォルトは 2 秒ですが、DNS_TMOUT マクロにより変更できます。再送間隔の単位は、カーネルのタイマの割り込み周期(Tick)ですので、2000/MSEC のように指定してください。

DNS 要求送信の最大リトライ回数

DNS 要求送信の最大リトライ回数のデフォルトは 3 回ですが、DNS_QRY_COUNT マクロにより変更できます。

第 5 章 FTP クライアント

FTP はファイルを転送するためのプロトコルです。

5.1. ファイル構成

INC¥nocftp.h … FTP クライアントのヘッダファイル

SRC¥nocftp.c … FTP クライアントのソースファイル

FTP クライアント機能の API を発行するユーザープログラムで、nocftp.h をインクルードしてください。

nocftp.c はコンパイルされて、NORTi Oceans のライブラリに結合されていますので、カスタマイズをされる場合にだけ、nocftp.c をプロジェクトに加えてビルドしてください。(ライブラリ内の nocftp モジュールに優先してリンクされます)

5.2. 使用するリソース

タスク	0
セマフォ	0
イベントフラグ	0
メールボックス	0
固定長メモリプール	0
データキュー	0
周期ハンドラ	0
TCP 受付口	1
TCP 通信端点	2
UDP 通信端点	0

5.3. 注意事項

これらの API が複数のタスクで同時に発行されることを考慮した排他制御は、省略されています。待ち状態の発生しない API(注 1)については、複数のタスクで発行することが可能ですが、各タスクの優先度を同じにして、同時となるのを抑制してください。

複数のクライアントを設ける場合は、待ち状態の発生する API であっても、対象クライアントが異なるならば、別々のタスクから同時に発行することが可能ですが、やはり、タスクの優先度は同じとしてください。

(注 1) ftp_setbuf()、ftp_select()、ftp_option()

5.4. API

ftp_ini

機能　FTP クライアント初期化

形式　ER ftp_ini(T_FTP *ftp);

引数　ftp　FTP クライアント管理ブロックへのポインタ

解説　FTP クライアント用の変数領域(管理ブロック)を、T_FTP 構造体としてユーザープログラム側に定義し、その先頭アドレスを ftp に指定して最初に一度だけ呼び出してください。

戻値　E_OK　正常終了

ftps_ini

機能　　複数の FTP クライアント初期化

形式　　ER ftps_ini(T_FTP *ftp, INT n);

引数　　ftp　　FTP クライアント管理ブロックへのポインタ
　　n　　FTP クライアント数 (1～)

解説　　FTP クライアントを複数設けて、別々の FTP サーバに同時に接続することができます。
FTP クライアントの数分の変数領域(管理ブロック)を、T_FTP 構造体の配列としてユーザープログラム側に定義し、その先頭アドレスを ftp に、配列の要素数を n に指定して、最初に一度だけ呼び出してください。
本 API で初期化する場合には、ftp_ini()を使用しないでください。n に 1 を指定した場合には、ftp_ini()と同じになります。

戻値　　E_OK　　正常終了
　　E_PAR　　FTP クライアント数が 0 以下

ftp_select

機能　　FTP クライアントの選択

形式　　ER ftp_select(INT i);

引数　　i　　FTP クライアント番号 (0～)

解説　　ftps_ini()で複数クライアントを初期化した場合は、続く API の対象クライアントを、本 API で選択してください。ftp_ini()で初期化した場合は、本 API を発行する必要はありません。
FTP クライアント毎にユーザータスクを設けて、その各先頭で、本 API が発行されることを想定していますが、ftp_select()を適宜発行することで、1 つのユーザータスクで、複数のクライアントを切り替えながら使用することもできます。

戻値　　E_OK　　正常終了
E_PAR　　FTP クライアント番号が範囲外

ftp_setbuf

機能　FTP クライアント用バッファのコンフィグレーション

形式　ER ftp_setbuf(VP buf, INT bufsz);

引数　buf　バッファのアドレス
　　bufsz　バッファのサイズ

解説　FTP クライアント用のバッファは、デフォルトで内部に確保されますが、バッファを外部(ユーザープログラム側)に設ける場合には、この API で、その先頭アドレスとサイズを指定できます。
buf に NULL を指定すると、内部に確保されるバッファのサイズだけを変更することができます。
デフォルトで内部に確保されるバッファサイズは、1792 バイトです。本 API で指定できるバッファサイズの最小値は、320 バイトです。
バッファをデフォルトから変更する場合にだけ、ftp_ini()、または、ftps_ini()に続けて発行してください。ftps_ini()で初期化した場合には、ftp_select()で指定しながらクライアント毎に発行する必要があります。

戻値　E_OK　正常終了
　　E_PAR　バッファサイズが不正
　　E_OBJ　既に ftp_connect()された後、または、FTP クライアントが未選択

ftp_option

機能　FTP オプションの設定

形式　ER ftp_option(INT optname, VP_INT optval);

引数　optname　オプションの種別
　　optval　オプションの値

解説　次の optname と optval 組み合わせで、オプションを 1 つずつ設定できます。

optname	optval（型）
OPT_FTP_NIF	ネットワークインタフェース（T_NIF *型）
OPT_FTP_VERBOSE	応答メッセージを抑制（BOOL 型）
OPT_FTP_PASSIVE	パッシブモード（BOOL 型）
OPT_FTP_DEBUG	デバッグモード（BOOL 型）
OPT_FTP_PORTNO	ポート番号（UH 型）
OPT_FTP_CALLBACK	イベント通知用のコールバック関数（FTP_CALLBACK 型）

本 API は、ログイン中は使用できませんが、ログアウト後再度ログインする前は再設定できます。

戻値　E_OK　正常終了
　　E_NOSPT　未サポートのオプション
　　E_OBJ　既にログインしている、または、FTP クライアントが未選択

ftp_connect

機能　FTP サーバへの接続

形式　ER ftp_connect(UW ipaddr, const char *user, const char *pass);

引数　ipaddr　接続先の IP アドレス
　　user　ユーザーID 文字列
　　pass　パスワード文字列

解説　FTP サーバに接続し、ログインまでを実行します。ログインに失敗した場合は、FTP サーバとの接続を切断します。

戻値　E_OK　正常終了
　　E_PAR　ユーザーID、または、パスワードが不正
　　E_NOMEM　メモリ不足
　　E_CLS　ログイン失敗で FTP サーバとの接続を切断
　　E_OBJ　既にログインしている、または、FTP クライアントが未選択
　　その他　内部で発行している TCP/IP API のエラー

ftp_command

機能　FTP コマンドの実行

形式　ER ftp_command(const char *command);

引数　command FTP コマンド文字列

解説　次の FTP コマンドを直接指定して実行できます。コマンドに引数を付加する場合は 1 つ以上のスペースで区切ってください。
"dir", "ls", "cd", "rm", "get", "put", "bye", "quit", "ascii","bin", "passive"
前もって ftp_connect()で FTP サーバと接続しておく必要があります。
"get"、または、"dir"コマンドを実行した後は、続けて、ftp_read()でデータを読み出してください。
"put"コマンドを実行した後は、続けて、ftp_write()でデータを書き込んでください。

戻値　E_OK　正常終了
E_PAR　コマンドが不正(FTP サーバとの接続は継続)
E_CLS　通信エラーで FTP サーバとの接続を切断
E_OBJ　FTP クライアントが未選択

ftp_open

機能　FTP サーバのファイルオープン

形式　ER ftp_open(const char *path, const char *mode);

引数　path　ファイル名
mode　モード　("r" : 読み出し, "w" : 書き込み)

解説　ftp_command()による"get"や"put"コマンドの代わりに利用できます。
path で指定されたファイルを開きます。
mode に"r"を指定すると、続けて ftp_read()でデータを読み出せます。
mode に"w"を指定すると、続けて ftp_write()でデータを書き込めます。

戻値　E_OK　正常終了
E_PAR　ファイル名やモードが不正(FTP サーバとの接続は継続)
E_CLS　通信エラーで FTP サーバとの接続を切断
E_OBJ　FTP クライアントが未選択

ftp_read

機能　FTP サーバからのデータ読み出し

形式　ER ftp_read(void *buf, int size);

引数　buf　読み出したデータを格納するバッファへのポインタ
　　size　バッファのサイズ

解説　FTP サーバからデータを読み出します。
　　前もって ftp_open()の"r"モードでファイルをオープンしておくか、ftp_command()により、"dir"、または、"get"コマンドが実行されている必要があります。
　　戻値が size の値未満の場合は、全てのデータの読み出しが終了したことを示します。

戻値　0 以上　正常終了(取得したデータ長)
　　E_OBJ　ファイル未オープン、または、FTP クライアントが未選択
　　E_CLS　通信エラーで FTP サーバとの接続を切断

ftp_write

機能　FTP サーバへのデータ書き込み

形式　ER ftp_write(const void *buf, int size);

引数　buf　書き込むデータが格納されているバッファへのポインタ
　　size　データの長さ

解説　FTP サーバへデータを書き込みます。
　　前もって ftp_open()の"w"モードでファイルをオープンしておくか、ftp_command()により、"put"コマンドが実行されている必要があります。

戻値　E_OK　正常終了
　　E_CLS　通信エラーで FTP サーバとの接続を切断
　　E_OBJ　ファイル未オープン、または、FTP クライアントが未選択

ftp_close

機能　FTP サーバのファイルのクローズ

形式　ER ftp_close(void);

引数　buf　書き込むデータが格納されているバッファへのポインタ
　　size　データの長さ

解説　必ず、ftp_read()、ftp_write()の後に呼び出してください。
　　データの読み出し、書き込みを終了します。

戻値　E_OK　正常終了
　　E_CLS　通信エラーで FTP サーバとの接続を切断
　　E_OBJ　FTP クライアントが未選択

ftp_exit

機能　FTP サーバとの接続終了

形式　ER ftp_exit(void);

引数　なし

解説　FTP サーバとの接続を切断します。
オープン中のファイルがあれば先にクローズ(実行中のコマンドがあれば中断)します。

戻値　E_OK　正常終了
E_CLS　通信エラーで FTP サーバとの接続を切断
E_OBJ　FTP クライアントが未選択
その他　内部で発行している TCP/IP API のエラー

コールバック

機能　イベント通知用のコールバック関数

形式　void callback(int event, VP parblk, int len);

引数　event　イベントコード(TEV_MESSAGE)
parblk　パラメータ
len　パラメータの長さ

解説　FTP クライアントの処理中に発生したイベントを通知します。現在サポートされているイベントは、応答メッセージの通知(TEV_MESSAGE)のみです。

戻値　なし

5.5. コンフィグレーション

FTP クライアントは、下記のコンフィグレーションを変更することができます。コンフィグレーションを行う場合は、noccfg.h をインクルードする前に各マクロを定義してください。

FTP クライアントのタイムアウト

コマンドポート、及び、データポートからの送受信待ち、データポートの能動接続待ち、コマンドポート、及び、データポートの接続終了待ちの時間を変更できます。これらの待ち時間はデフォルトで 60 秒ですが、FTP_TMOUT マクロにより変更できます。タイムアウト時間の単位は、カーネルのタイマの割り込み周期(Tick)ですので、60000/MSEC のように指定してください。

コマンドポートからのレスポンスの 2 回目以降の受信のタイムアウト

コマンドポートからレスポンスを受信した場合、レスポンスの続きを受信待ちします。この時のタイムアウトがデフォルトで 500 ミリ秒ですが、FTP_TMOUT2 マクロにより変更できます。

第 6 章 SNTP クライアント

時間を合わせるためのプロトコルに SNTP(Simple Network Time Protocol)があります。NORTi Oceans は SNTP クライアントをサポートしており、システムの時刻を NTP サーバと同期させることができます。

6.1. ファイル構成

INC¥nocsntp.h … SNTP クライアントのヘッダファイル
INC¥sntptime.h … 日付/時間のヘッダファイル
SRC¥nocsntp.c … SNTP クライアントのソースファイル
SRC¥sntptime.c … 日付/時間のソースファイル

SNTP クライアント機能の API を発行するユーザープログラムで、nocsntp.h と sntptime.h をインクルードしてください。

nocsntp.c と sntptime.c はコンパイルされて、NORTi Oceans のライブラリに結合されていますので、カスタマイズをされる場合にだけ、nocsntp.c と sntptime.c をプロジェクトに加えてビルドしてください。(ライブラリ内の nocdns と sntptime モジュールに優先してリンクされます)

6.2. 使用するリソース

タスク	0
セマフォ	0
イベントフラグ	0
メールボックス	0
固定長メモリプール	0
データキュー	0
周期ハンドラ	0
TCP 受付口	0
TCP 通信端点	0
UDP 通信端点	1

6.3. 注意事項

これらの API が複数のタスクで同時に発行されることを考慮した排他制御は、省略されています。待ち状態の発生しない API(注 1)については、複数のタスクで発行することが可能ですが、各タスクの優先度を同じにして、同時となるのを抑制してください。

(注 1) sntp_connect()、sntp_get_tim()以外

6.4. API

sntp_ini

機能　SNTP クライアント初期化

形式　ER sntp_ini(T_SNTP *sntp);

引数　sntp　SNTP クライアント管理ブロックへのポインタ

解説　SNTP クライアント用の変数領域(管理ブロック)を、T_SNTP 構造体としてユーザープログラム側に定義し、その先頭アドレスを sntp に指定して最初に一度だけ呼び出してください。

戻値　E_OK　正常終了

sntp_connect

機能　NTP サーバの設定

形式　ER sntp_connect(const char *ntp_srv, UW dns_ip);

引数　ntp_srv　NTP サーバのアドレス
　　dns_ip　DNS サーバのアドレス

解説　NTP サーバを指定します。
　　sntp_get_tim()の実行前に最低 1 回は本関数で NTP サーバを指定してください。最初に指定されたサーバが、プライマリ NTP サーバとなり、2 回目に指定されたサーバが、セカンダリ NTP サーバになります。sntp_connect()を呼び出せるのは 2 回までです。
　　本関数では、ntp_srv にドメイン名を指定した場合、DNS クライアントを使用します。DNS クライアント(dns_resolve())が使用できる状態で本関数を呼び出してください。

戻値　E_OK　正常終了
　　E_OBJ　3 回以降の呼び出し
　　その他　dns_resolve()の戻値

sntp_option

機能　SNTP オプションの設定

形式　ER sntp_option(INT optname, VP_INT optval);

引数　optname オプションの種別
optval オプションの値

解説　次の optname と optval 組み合わせで、オプションを 1 つずつ設定できます。

optname	optval (型)
OPT_SNTP_NIF	ネットワークインタフェース (T_NIF *型)

各 API 発行後でもネットワークインタフェースを設定し、各 API を再発行できます。

戻値　E_OK 正常終了
E_NOSPT 未サポートのオプション

sntp_get_tim

機能　時刻の取得

形式　ER sntp_get_tim(T_NTP_TIM *pk_tim);

引数　pk_tim　NTP フォーマットの時刻を格納するポインタ

解説　指時刻を NTP サーバに問い合わせ、システムの時刻を校正します。
問い合わせでエラーがあった場合は設定されている他の NTP サーバがあればリトライを行います。
pk_tim に NULL を指定して呼び出した場合は取得した時刻は返しません。

戻値　E_OK　正常終了
E_TMOUT　タイムアウト
その他　内部で発行している TCP/IP API のエラー

my_asctime

機能　時刻を文字列に変換（出力バッファ指定なし）

形式　char *my_asctime(const my_tm_t *tms);

引数　tms　my_tm_t 形式の時刻

解説　時刻を“Wday Mon dd hh:mm:ss yyyy¥n”形式の文字列に変換します。

戻値　変換した文字列へのポインタ

my_asctime_r

機能　　時刻を文字列に変換（出力バッファ指定あり）

形式　　char *my_asctime_r(const my_tm_t *tms, char *buf);

引数　　tms　　my_tm_t 形式の時刻
　　buf　　変換結果を格納する文字列へのポインタ

解説　　時刻を“Wday Mon dd hh:mm:ss yyyy¥n”形式の文字列に変換します。
　　文字列バッファは 26 文字分必要です。本関数はスレッドセーフです。

戻値　　変換した文字列へのポインタ

my_ctime

機能　　暦時間から現地時間の文字列を算出（出力バッファ指定なし）

形式　　char *my_ctime(const my_time_t *timer);

引数　　timer　　暦時間を受け取る変数へのポインタ

解説　　暦時間を現地時間に変換し、文字列として返します。
　　文字列は“Wday Mon dd hh:mm:ss yyyy¥n”形式となります。

戻値　　変換した文字列へのポインタ

my_ctime_r

機能　　暦時間から現地時間の文字列を算出（出力バッファ指定あり）

形式　　char *my_ctime_r(const my_time_t *timer, char *buf);

引数　　timer　暦時間を受け取る変数へのポインタ
　　buf　　変換結果を格納する文字列へのポインタ

解説　　暦時間を現地時間に変換し、文字列として返します。
　　文字列は “Wday Mon dd hh:mm:ssyyyy¥n” 形式となります。
　　文字列バッファは 26 文字分必要です。本関数はスレッドセーフです。

戻値　　変換した文字列へのポインタ

my_time

機能　　システムの現在の暦時間を取得

形式　　my_time_t my_time(my_time_t *t);

引数　　t　　暦時間を受け取る変数へのポインタ

解説　　システムの現在の暦時間を返します。
　　暦時間は 1/Jan/1970 AM0:00 UTC からの経過秒数です。
　　SNTP で時刻設定してご利用ください。
　　暦時間を戻り値で受け取る場合、引数 t には NULL が指定できます。

戻値　　暦時間の値

my_clock

機能　起動時からのシステムクロックを取得

形式　my_clock_t my_clock(void);

引数　なし

解説　起動時からのシステムクロックを返します。
MSEC 単位の tick count の下位 32bit のみが返ります。

戻値　システムクロックの下位 32 ビット

my_mktime

機能　現地時間を暦時間に変換

形式　my_time_t my_mktime(my_tm_t *t);

引数　t　　my_tm_t 形式の時刻

解説　現地時間の年、月、日、時、分、秒を暦時間に変換します。
暦時間は 1/Jan/1970 からの経過秒数となります。
夏時間はサポートされていません。

戻値　変換できた場合　変換した暦時間
変換できない場合 -1
パラメータが無効 -1

my_gmtime

機能　　暦時間を my_tm_t に変換（格納先指定なし）

形式　　my_tm_t *my_gmtime(const my_time_t *tms);

引数　　tms　　暦時間

解説　　暦時間を、年、月、日、時、分、秒に変換します。

戻値　　変換できた場合　変換した my_tm_t 形式の時刻へのポインタ
　　変換できない場合 NULL

my_gmtime_r

機能　　暦時間を my_tm_t に変換（格納先指定あり）

形式　　my_tm_t *my_gmtime_r(const my_time_t *tms, my_tm_t *tm_time);

引数　　tms　　暦時間
　　tm_time　変換する my_tm_t 形式の時刻へのポインタ

解説　　暦時間を、年、月、日、時、分、秒に変換します。
　　本関数はスレッドセーフです。

戻値　　変換できた場合　変換した my_tm_t 形式の時刻へのポインタ
　　変換できない場合 NULL

my_localtime

機能　　暦時間を現地時間に変換（格納先指定なし）

形式　　my_tm_t *my_localtime(const my_time_t *tms);

引数　　tms　　暦時間

解説　　暦時間を、現地時間に変換します。

戻値　　変換できた場合　変換した my_tm_t 形式の時刻へのポインタ
　　変換できない場合 NULL

my_localtime_r

機能　　暦時間を現地時間に変換（格納先指定あり）

形式　　my_tm_t *my_localtime_r(const my_time_t *tms, my_tm_t *tm_time);

引数　　tms　　暦時間
　　tm_time　変換する my_tm_t 形式の時刻へのポインタ

解説　　暦時間を、現地時間に変換します。本関数はスレッドセーフです。

戻値　　変換できた場合　変換した my_tm_t 形式の時刻へのポインタ
　　変換できない場合 NULL

set_timezone_ofst

機能　タイムゾーンによる時差を設定

形式　ER set_timezone_ofst(W tz);

引数　tz　タイムゾーンによる UTC から時間差(秒)

解説　タイムゾーンによる標準時からの時差を設定します。
設定できる範囲は-12*3600～+14*3600 です。
my_localtime, my_localtime_r, my_mktime を利用する場合に設定が必要です。

戻値　E_OK　正常終了
E_PAR　設定無効

例　W tz = 9 * 60 * 60;
set_timezone_ofst(tz); /* 日本の UTC からの時間差を設定 */

get_timezone_ofst

機能　タイムゾーンによる時差を取得

形式　W get_timezone_ofst(void);

引数　なし

解説　タイムゾーンによる標準時からの時差を取得します。

戻値　タイムゾーンによる標準時からの時差(秒)
無効な値の場合は-1

6.5. コンフィグレーション

SNTP クライアントは、下記のコンフィグレーションを変更することができます。コンフィグレーションを行う場合は、noccfg.h をインクルードする前に各マクロを定義してください。

SNTP クライアントのタイムアウト

NTP サーバへの問い合わせのタイムアウト時間はデフォルトで 1 秒ですが、SNTP_TMOUT マクロにより変更出来ます。タイムアウト時間の単位は、カーネルのタイマの割り込み周期(Tick)ですので、1000/MSEC のように指定してください。

6.6. データ型

my_tm_t

```
typedef struct my_tm
{
    int tm_sec;   /* 秒 (0..61) */
    int tm_min;   /* 分 (0..59) */
    int tm_hour;  /* 時 (0..23) */
    int tm_mday;  /* 日 (1..31) */
    int tm_mon;   /* 月 (1 月が 0) */
    int tm_year;  /* 年 (1900 年が 0) */
    int tm_wday;  /* 曜日(日曜日が 0) */
    int tm_yday;  /* 1 月 1 日からの日数 */
    int tm_isdst; /* 季節時間の有無 */
} my_tm_t;
```

my_time_t

```
typedef long my_time_t; /* 暦時間 */
```

my_clock_t

```
typedef long my_clock_t; /* システムクロック */
```

NORTi Oceans ユーザーズガイド

ミドルウェア編

2012 年　3 月　初版

株式会社ミスポ http://www.mispo.co.jp/
〒213-0002　川崎市高津区二子 5-1-1
TEL 044-829-3381　FAX 044-829-3382
一般的なお問い合わせ sales@mispo.co.jp
技術サポートご依頼　norti@mispo.co.jp