NTP for NORTi

User's Guide

2008 年 5 月版

MiSPO
株式会社ミスポ

目次

2006 年 5 月版で訂正された項目

更新箇所	更新内容
4.1	usr_set_tim を追加
4.2	mktime の[解説]を更新
3.2	マクロに NTP_MIN_POLL, NTP_MAX_POLL を追加
4.1	ntp_stop を追加

2006 年 9 月版で訂正された項目

更新箇所	更新内容
1.5	次の制限事項を削除。「NTP はサーバーから獲得した時間の誤差が 128 ミリ秒以上、1000 秒 未満の場合にのみシステムクロックを調整します」
3.2	NTP_MIN_OFFSET,NTP_MAX_OFFSET を追加
4.1	「システム時間が 1000 秒を超えた」を「システム時間が NTP_MAX_OFFSET(秒)を超えた」に変更

2007 年 9 月版で訂正された項目

更新箇所	更新内容
1.3	「IPv6(RFC4330)に対応。」を追加
1.6	ファイル構成に md5.h、md5.c を追加
4.1	ntp6_ini の説明を追加
4.1	ntp_start の説明に IPv6 用のパラメータ例を追加
全体	「システムクロック」を「システム時刻」に変更

2008 年 5 月版で訂正された項目

更新箇所	更新内容
4.1	ntp_ini の初期化用パラメータ：tskId の説明を追加
4.1	ntp_ini で使用される t_ntp_ini 構造体についての説明を追加

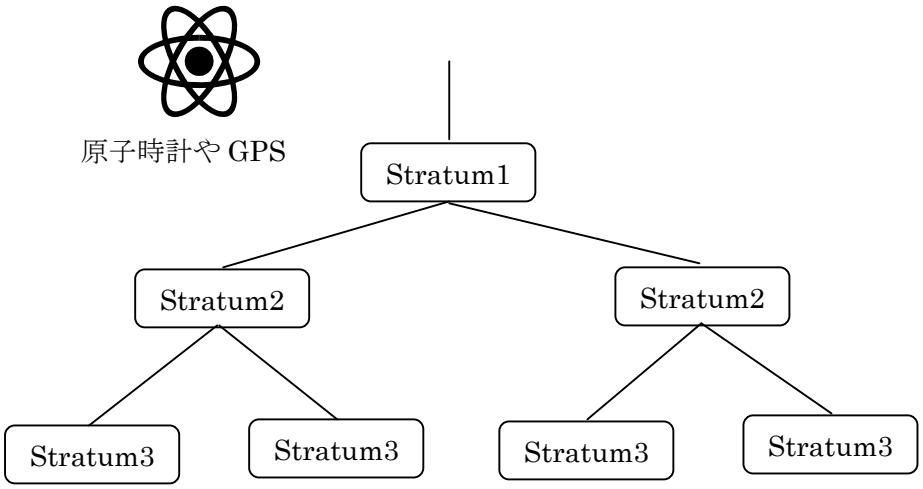
第 1 章 導入

1.1 はじめに

NTP for NORTi は NORTi TCP/IP プロトコルスタックのアプリケーションレイヤーで NTP (Network Time Protocol)を実現します。本書では NTP の仕様に関してのみ説明を行っています。カーネルや TCP/IP プロトコルスタックの使用方法については各ユーザーズガイドを参照してください。

1.2 NTPの概要

NTP(Network Time Protocol)はタイムサーバーとクライアント間で管理して保持しているシステム時刻の同期を行うプロトコルです。NTP は UDP のポート 123 番を使って、通信を行います。NTP サーバーは下図のように Strutum といわれる階層に分類されており、この階層によって精度が異なります。最も高い精度は Stratum1 のサーバーで、これは原子時計や GPS などを使って精度が保たれています。



1.3 特長

・NTP Version3.0(RFC1305)に対応しています。
・NTP は正確な時刻情報を維持するデーモンとして動作します。
・アプリケーションは NTP の初期化以外に特別な呼び出しは不要です。
・NTP for NORTi はクライアントとデーモン(サーバー)の機能が並行して動作します。
・IPv6(RFC4330)に対応しています。

1.4 NTP for NORTiの動作

NTP for NORTi はリモート NTP サーバーの時間を使って NORTi のシステム時刻を初期化します。NTP が起動された後、周期的に特定のサーバーと同期がとられます。NORTi のシステム時刻はシステム起動後からのシステムクロックの 48bit の周期割り込みのカウンターが管理されています。NTP は 64bit の固定長小数点フォーマットでサーバーから時間を取得します。その時間には 1900, Jan 1st 12:00:00 UTC からの経過時間が含まれています。ANSI C では主に 32bit の整数型フォーマットで 1970, Jan 1st 12:00:00 UTC からの経過時間を扱います。個々の書式変換は NTP 内部で行います。NTP はいくつかのサーバーとから得られた情報から、最適な時間を計算しシステム時刻と同期します。また、NTP は他のクライアントからのリクエストに応答し、内部で管理している時間情報を他のクライアントに返します。

1.5 制限事項

・NORTi Version4 以外での動作は保証しません。
・NTP で扱う時間の精度は MSEC(ミリ秒)単位です。
・NTP はサーバーから獲得した時間の誤差が 128 ミリ秒以上、1000 秒 未満の場合にのみシステム時刻を調整します。
・ユーザーアプリケーションは NTP デーモンの初期化後に set_tim 関数を使用しないでください。

1.6 ファイル構成

NTP は次のファイルから構成されます。

/INC

- nonntp.h
  - NTP 標準ヘッダ NTP で使用される構造体などが定義されています
- ntpcfg.h
  - NTP コンフィグレーションヘッダ。アプリケーションはどこか一箇所で必ずこのファイルを定義してください
- nontime.h
  - ANSI C Time 関数用ヘッダファイル
- md5.h
  - MD5 メッセージダイジェストアルゴリズムヘッダ

/SRC

- nonntp.c
  - NTP プロトコルソースコード
- ntpinfc.c
  - Time 変換用ユーティリティ関数が含まれています
- nontime.c
  - ANSI C Time 関数が含まれています
- md5.c
  - MD5 メッセージダイジェストアルゴリズム

リトルエンディアン動作時はnontp.cを**LITTLE_ENDIAN**マクロ付きでコンパイルしてください。

第 2 章 NTPの実装

2.1 概要

NTP の機能はタスクとして動作します、NTP タスクは UDP の受信コールバックから起床されます。
NTP サーバーへのリクエストする、インターバル時間は 64 秒～1024 秒でこの周期は NTP_MIN_POLL と NTP_MAX_POLL で変更できます。

2.2 リソース

タスク×1

NTP 送受信用タスク

周期起動ハンドラ×1

NTP クライアントが周期的に要求を行うために使用

メイルボックス×1

UDP の受信コールバックが NTP タスクに通知するために使用

UDP 通信端点×1

NTP パケットの送受信用に使用(ポート番号 123)

第 3 章 コンフィグレーション

本章では NTP を使用するために必要なコンフィグレーションを説明します。

3.1 マクロ

アプリケーションは ntpcfg.h を必ずどこか 1 箇所で#include してください。ntpcfg.h は NTP のコンフィグレーションを行うためのヘッダファイルです。ntpcfg.h を#include する前に次の様な定数を記述することにより、コンフィグレーションを行うことができます。
()はデフォルト値で、指定がない場合に使用されます。

#define NTP_SRV_MAX	1	同期を取る NTP サーバーの数
#define PRI_NTP_TSK	6	NTP タスクの優先度
#define SSZ_NTP_TSK	1024	NTP タスクのスタックサイズ
#define LOCAL_TIME_ZONE	32400	UTC(協定世界時間)と Tokyo の ローカルタイムとの誤差(秒)
#define NTP_MIN_POLL	6	NTP リクエスト最小周期(2 の 6 乗= 64 秒)
#define NTP_MAX_POLL	10	NTP リクエスト最大周期(2 の 10 乗= 1024 秒) 必ず NTP_MAX_POLL > NTP_MIN_POLL で設定してください。
#define NTP_MIN_OFFSET	10	最小オフセット時間(ミリ秒)
#define NTP_MAX_OFFSET	1000	最大オフセット時間(秒)

NTPはサーバーから獲得した時間の誤差がNTP_MIN_OFFSET（10）ミリ秒以上、NTP_MAX_OFFSET(1000)秒の場合にのみシステム時刻を調整します。

3.2 時間に関連する構造体

NORTi で使用する時間と NTP で扱う時間の相違を述べます。

3.2.1 NORTi で使用する SYSTIM フォーマット

NORTi はシステム時刻を秒またはミリ秒単位で管理していません。NORTi のシステム時刻は、システム起動時に 0 クリアされ以後、周期割込みごとにカウントアップされます。NORTi では get_tim,set_tim システムコールでこの値を読み込み、書き込みを行うことができます。この値は 48bit の構造体 SYSTIM に格納されています。

```
typedef struct    16,32 ビット CPU の場合
{       H       utime;  上位 16 ビット
        UW      ltime;  下位 32 ビット
```

```
} SYSTIM;
```

3.2.2 NTP タイムスタンプフォーマット

NTP のタイムスタンプは 1900 年 1 月 1 日の 0 時からの相対秒数による 64 ビット符号なし整数で表されており、前半 32 ビットが整数部、後半 32 ビットが小数部となっています。したがって、非常に精緻な要求に対しては 200 ピコ秒までの精度が表現できます。

```
typedef struct t_ntp_tim
{    union{
        UINT    uip;        /* unsigned 整数部 */
        INT     ip;         /* signed 整数部    */
     }U_ip;
     union{
        UINT    ufp;        /* unsigned 小数部 */
        INT     fp;         /* signed 小数部    */
     }U_fp;
}T_NTP_TIM;
```

第 4 章 関数仕様

4.1 NTPのAPI

NTP は NTP の初期化とカレンダーやシステム時間をアクセスするために time.h に定義されているいくつかの ANSIC の関数をサポートしています。

---

ntp_ini
ntp6_ini

[機 能] NTP の初期化を行う(ntp6_ini は IPv6/IPv4 デュアルスタック用)

[形 式] ER ntp_ini(T_NTP_INI * ntp_ini);
ER ntp6_ini(T_NTP_INI *ntp_ini);
T_NTP_INI * ntp_ini　　NTP の初期化用パラメータのポインタ

```
typedef struct t_ntp_ini{
        ID tskid;                       /* NTPで使用するタスクID */
        ID mbxid;                       /* NTPで使用する*/
        ID cycid;                       /* NTPサイクリックハンドラID */
        ID cepid;                       /* NTP UDP通信端点ID */
        NTP_CALLBACK callback;          /* NTPコールバック関数  */
}T_NTP_INI;
```

タスク ID は、NTP が使用するタスク ID を指定します。タスク ID に、0 を指定した場合は、NTP が自動的にタスク ID を割り当てます。

コールバック関数はユーザーアプリケーション側で用意するプログラムです。NTP のイベントがコールバック関数に通知されます。コールバックによる通知が不要な場合、callback には NULL を設定してください。

void function-name(UB event, UW data);
UB event　　NTP のイベントコード

NTP_CLK_INI(0x01)
　　システム時刻が最初に初期化されたときに、このイベントで呼ばれます。
　　UW data　　時間を取得した IP アドレスが含まれます。
NTP_CLK_SYNC(0x02)
　　システム時刻に設定が行われたときに、このイベントで呼ばれます。
　　UW data　　未使用
NTP_CLK_RCV(0x03)
　　NTP パケットが受信されたときに、このイベントで呼ばれます。
　　UW data　　NTP パケットの送信元アドレス
NTP_CLK_OFFSET_ERROR(0x04)
　　タイムサーバーとローカルな NTP_MAX_OFFSET(秒)を超えた場合、このイベントが呼ばれます。
　　UW data　　オフセット値

[戻 値] ＝ E_OK 正常終了
≠ E_OK 異常終了

[解 説] NTP での初期化を行います。ntp_ini 内部ではタスク、メモリプール、サイクリックハンドラ、UDP 通信端点の生成を行っています。この関数で異常終了した場合、これらいずれかの生成でエラーになっています。NTP の初期化用パラメータを指定する t_ntp_ini 構造体は、ntp_ini 処理内で、t_ntp_ini 構造体の内容を書き換えることがあります。従って、t_ntp_ini 構造体は、書き換え可能な変数に割り当てるようにして下さい。

ntp_start

[機 能] NTP デーモンの開始

[形 式] ER ntp_start(char *srv[], UW dns_ip, TMO tmout);

char *srv[]	NTP サーバーIP アドレスのリスト IPv4 アドレスの文字列形式(“202.234.232.223") IPv4 ドメイン形式(“clock.nc.xxx.jp”) IPv6 アドレスの文字列形式(“3ffe:501:ffff:2:a00:46ff:febc:ba49") IPv6 ドメイン形式(“-6 ntp2.bit.nl”)
UW dns_ip	ドメイン形式使用時に使用する DNS サーバーのアドレス
TMO tmout	タイムアウト

[戻 値] ＝ E_OK 正常終了
≠ E_OK 異常終了

[解 説] NTP デーモンを開始し System タイマーの初期化を行います
この関数で接続する NTP サーバーを複数指定できます。指定できるサーバー数は NTP_SRV_MAX で定義されています。この値以上を設定できません。

ntp_stop

[機 能] NTP デーモンの停止

[形 式] ER ntp_stop(void);

T_NTP_TIM	*pk_tim システム時間を格納するポインタ

[戻 値] ＝ E_OK 正常終了
≠ E_OK 異常終了

[解 説] この関数はサーバーからタイムリクエストのポーリングを停止します。
NTP デーモンの再開は ntp_start で行うことができます。
この関数はシステム時刻に影響しません。

ntp_get_tim

[機 能] NTP フォーマットのシステム時間を取得する

[形 式] void ntp_get_tim(T_NTP_TIM *pk_tim);
T_NTP_TIM　　*pk_tim システム時間を格納するポインタ

[戻 値] なし

[解 説] NTP フォーマットのシステム時間を取得します。pk_tim には 12:00:00 Jan 1st 1900 UTC からの経過時間が含まれています。

ntp_set_tim

[機 能] NTP フォーマットのシステム時間を設定する

[形 式] void ntp_set_tim(T_NTP_TIM *pk_tim);
T_NTP_TIM　　*pk_tim システム時間を格納するポインタ

[戻 値] なし

[解 説] NTP フォーマットのシステム時間を NORTi SYSTIME フォーマットに変換しシステム時間として設定します。pk_tim には 12:00:00 Jan 1st 1900 UTC からの経過時間が含みます。

usr_set_tim

[機 能] アプリケーションからの時間設定

[形 式] void usr_set_tim(time_t t, UW msec, W zone);

time_t	t	12:00:00 Jan 1st 1970 UTC からの経過時間(秒単位).
UW	msec	ミリ秒
W	zone	UTC(協定世界時間)とローカルタイムとの誤差(秒)

[戻 値] ＝ E_OK 正常終了

≠ E_OK 異常終了

[解 説] この関数はアプリケーションからシステム時刻を設定します。

t は mktime 関数を使って計算することができます。システム時刻は NTP デーモンを実行ぜず、この関数によって設定することができます。この関数を呼び出す前に ntp_stop で NTP デーモンを停止してから使用してください。

4.2 ANSI C Time API

NTP for NORTi は時間に関連する ANSI C の関数をサポートしています。

4.2.1 構造体と型

clock_t

unsigned int 型のプログラム実行時間（システム時刻）を表します

time_t

unsigned int 型の暦時間（12:00:00 Jan 1st 1970 UTC からの経過秒）を表します

struct tm

この構造体には次の要素別の情報が含まれています

int tm_sec	秒(0-59)
int tm_min	分(0-59)
int tm_hour	時(0-23) 深夜零時からの時間
int tm_mday	月の中の日付(0-31)
int tm_mon	月(0-11) 1 月からの月数
int tm_year	西暦 1900 年からの年号
int tm_wday	曜日(0-6) 日曜からの曜日
int tm_yday	1 月 1 日からの日数(0-365)
int tm_isdst	サマータイムのフラグ

4.2.2 API

time	現在の暦時間を得る
clock	プログラムの実行時間(システム時間)を得る
mktime	struct tm(要素別の時間)を time_t(暦時間)に変換する
asctime	struct tm(要素別の時間)を ASCII 文字列に変換する
gmtime	暦時間を UTC(協定世界時間)に変換する
localtime	暦時間を現地時間に変換する

time

[機 能] 現在の暦時間を得る

[形 式] time_t　time(time_t *tp)
　　time_t　*tp　　取得する時間を格納するポインタ

[戻 値] 暦時間

[解 説] 12:00:00 Jan 1st 1970 UTC からの経過時間(秒単位)が返ります

clock

[機 能] プログラムの実行時間を得る

[形 式] clock_t　clock(clock_t *cp)
　　clock_t　*cp　　取得する時間を格納するポインタ

[戻 値] プログラム実行時間

[解 説] 32bit のプログラム実行時間(システム時刻)を返します。単位は MSEC です。
　　NORTi の get_tim で得る ltime の値と同じです。

mktime

[機 能] struct tm を time_t に変換する

[形 式] time_t mktime(struct tm *tp);

struct tm *tp　　要素別時間が格納されたポインタ

[戻 値] 暦時間

[解 説] 要素別時間(struct tm)を暦時間(time_t)に変換します。

この関数では以下の struct tm **メンバーの設定が必要です。**

int tm_sec **秒**(0-59)
int tm_min **分**(0-59)
int tm_hour **時**(0-23) **深夜零時からの時間**
int tm_mday **月の中の日付**(0-31)
int tm_mon **月**(0-11) 1 **月からの月数**
int tm_year **西暦** 1900 **年からの年号**

asctime

[機 能] struct tm を ASCII 文字列に変換する

[形 式] char *asctime(const struct tm *tp);

struct tm *tp　　要素別時間が格納されたポインタ

[戻 値] 変換された文字列

[解 説] 構造体として格納された時刻を文字列に変換します。

gmtime

[機 能] 暦時間を UTC に変換する

[形 式] struct tm *gmtime(const time_t *tp);
time_t *tp 暦時間が格納されたポインタ

[戻 値] UTC に変換された要素別時間が格納されたポインタ

[解 説] 暦時間(time_t)を UTC の要素別構造体(struct tm)に変換します。

localtime

[機 能] 暦時間を現地時間に変換する

[形 式] struct tm *localtime(const time_t *tp);
time_t *tp 暦時間が格納されたポインタ

[戻 値] 現地時間に変換された要素別時間が格納されたポインタ

[解 説] 暦時間(time_t)を現地時間の要素別構造体(struct tm)に変換します。

第 5 章 補足

5.1 国内の主要なNTPサーバー

clock.nc.fukuoka-u.ac.jp	Stratum1
clock.tl.fukuoka-u.ac.jp	Stratum1
ntp.cyber-fleet.net	Stratum2
ntp1.jst.mfeed.ad.jp	Stratum2
ntp2.jst.mfeed.ad.jp	Stratum2
ntp3.jst.mfeed.ad.jp	Stratum2

Stratum1 と Stratum2 のリストは以下の URL で参照できます

http://www.eecis.udel.edu/~mills/ntp/clock1a.html

http://www.eecis.udel.edu/~mills/ntp/clock2a.html

5.2 WindowsXPのNTPクライアントについて

WindowsXP で使用している NTP クライアントは通常の NTP とは仕様が異なり、精度が低いため、NTP for NORTi とは接続できません。

5.3 テスト

NTP for NORTi は Linux の NTPd、Bytefusion NTP Client およびインターネットで接続可能ないくつかの Stratum1、Stratum1 のサーバーとテストを行いました。

NTP for NORTi ユーザーズガイド

株式会社ミスポ	http://www.mispo.co.jp
一般的なお問い合せ	sales@mispo.co.jp
技術サポートご依頼	norti@mispo.co.jp