HI.CommunicationEngine
SNTPクライアント
リファレンスマニュアル

ルネサスセミコンダクタパッケージ＆テストソリューションズ株式会社

ご注意

1. 本製品(ソフトウエア製品及びその関連ソフトウエア製品を含む。以下、同じ。)の使用に際しては、「外国為替及び外国貿易法」等、技術輸出に関する日本及び関連諸国の関係法規の遵守が必要となります。
2. 弊社は、本製品の使用に際しては、弊社もしくは第三者の特許権、著作権、商標権、その他の知的所有権等の権利に関し、別途、個別の契約書等(マニュアルの記載を含む。以下、同じ。)にて弊社による明示的な許諾がある場合を除き、その保証または実施権の許諾を行うものではありません。また本製品を使用したことにより第三者の知的所有権等の権利に関わる問題が生じた場合、弊社はその責を負いませんので予めご了承ください。
3. 本製品およびその仕様、またはマニュアルに記載されている事柄については、将来、事前の予告なしに変更することがありますので、最終的な設計、ご購入、ご使用に際しましては、事前に最新の製品規格または仕様書(マニュアルを含む)をご確認ください。
4. 本製品の使用(マニュアル記載事項に基づくものも含む)により直接または間接に生ずるいかなる損害についても、弊社は一切の責任を負いません。また、本製品の配布に使用される搭載機器や媒体が原因の損害に対しましても、弊社は一切の責任を負いません。
5. 本製品を、宇宙、航空、原子力、燃焼制御、運輸、交通、各種安全装置、ライフサポート関連の医療機器等のように、特別な品質・信頼性が要求され、その故障や誤動作が直接人命を脅かしたり、人体に危害を及ぼす恐れのある用途向けには使用できません。お客様の用途がこれに該当するかどうか疑問のある場合には、事前に弊社営業担当迄ご相談をお願い致します。
6. 本製品を使用してお客様のシステム製品を設計される際には、通常予測される故障発生率、故障モードをご考慮の上、本製品の動作が原因での事故、その他の拡大損害を生じないようにフェールセーフ等の充分なシステム上の対策を講じて頂きますようお願い致します。
7. 本製品およびマニュアルの著作権は弊社が所有しております。お客様は、弊社から提供された本製品を、別途、個別の契約書等にて定める場合を除き、いかなる場合においても全体的または部分的に複写・解析・改変することはできないものとします。
8. お客様は、別途、個別の契約書等にて定める場合を除き、本製品のマニュアルの一部または全部を無断で使用、複製することはできません。
9. 弊社は、本製品を1台のコンピュータで使用する権利をお客様に対してのみ許諾します。よって、本製品を第三者へ譲渡、貸与、賃借することは許諾しないものとします。但し、別途、個別の契約書等にて定められる場合はその条件に従います。
10. 本製品をはじめ弊社製品およびその関連製品についてのお問い合わせ、ご相談は弊社営業担当迄お願い致します。

はじめに

このマニュアルは、HI.CommunicationEngine TCP/IPマネージャ上で動作するTCP/IPネットワークアプリケーション「SNTPクライアント(HI.CommunicationEngine SNTP)」について説明します。

本製品ではタスクとしてSNTPクライアント機能を実現しています。
SNTPクライアントは、TCP/IPマネージャを経由してネットワーク上のNTPサーバから時刻情報を受け取り、システム時計を更新する機能を提供します。

このリファレンスマニュアルではSNTPクライアントの使用方法および関連事項を説明します。TCP/IPマネージャについては関連マニュアルを参照してください。

【関連マニュアル】
- HI.CommunicationEngine TCP/IPマネージャ リファレンスマニュアル
- 使用するμITRON のユーザーズマニュアル

目次

図表目次

1. 概要

1.1 SNTP の機能概要

ネットワーク上のコンピュータ同士で時刻を同期させるためのプロトコルです。正確な時刻情報源を階層的に構築するための NTP プロトコルをベースとし、規定されたプロトコルです。本製品は SNTP version3 です（NTP サーバからの時刻情報は version4 形式まで受信可能です）。**本製品では、RFC 4330 で提唱する方法に準じた時刻補正処理を実装しており、2036 年以降も使用可能です。※1**

SNTP では、NTP を簡略化して、SNTP クライアントから NTP サーバに対する時刻情報の要求とそれに対する応答だけを使っています。NTP サーバのような長時間に渡る統計的な誤差の処理機能などを持たず、単発的な時刻合わせ機能しかないので、一時的にネットワーク トラフィックが混み合っていると精度が下がる可能性があります。SNTP パケットのやり取りを複数回使用して誤差を減らすなどの工夫が必要です。

※1 使用可能な範囲は、UTC 時刻で 1968 年 1 月 20 日から 2104 年 2 月 26 日までとなります。

1.2 SNTP クライアント

SNTP クライアントは「タスク」として実現します。

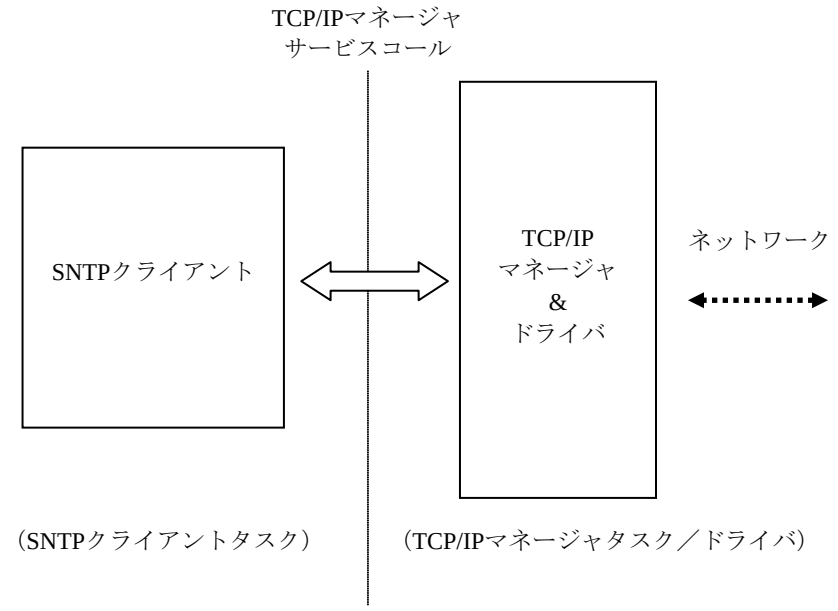

図 1.1 SNTPクライアントタスク

1.3　サポートする SNTP 機能

SNTP のプロトコル記述書では、下記の表のような動作モードが定義されています。
本 SNTP は、クライアント機能（CLIENT モード，BROADCAST モード，DUAL モード）のみサポートします。 サーバーとしての機能（SERVER モード）はサポート対象外※2 です。
下記の表は本製品のクライアント機能において、送信または受信可能な NTP パケットモードの対応表です。

表 1.1　送信／受信可能メッセージ

モード	受信	送信
3：クライアント	×	○
4：サーバー	○	×
5：ブロードキャスト	○	×

※1 認証の機能は、各 NTP サーバ毎に異なるために実装していません。
※2 本 SNTP の SERVER モードは、テスト用として実装されたものであり、製品の動作保証範囲外です。

1.4　時刻同期手順

SNTP では、次の４つのタイムスタンプ情報により時刻の同期を行います。

開始タイムスタンプ	T1	クライアントにより要求が送信された時刻
受信タイムスタンプ	T2	サーバにより要求が受信された時刻
送信タイムスタンプ	T3	サーバにより応答が送信された時刻
終了タイムスタンプ	T4	クライアントにより応答が受信された時刻

時刻同期処理の流れを示します。

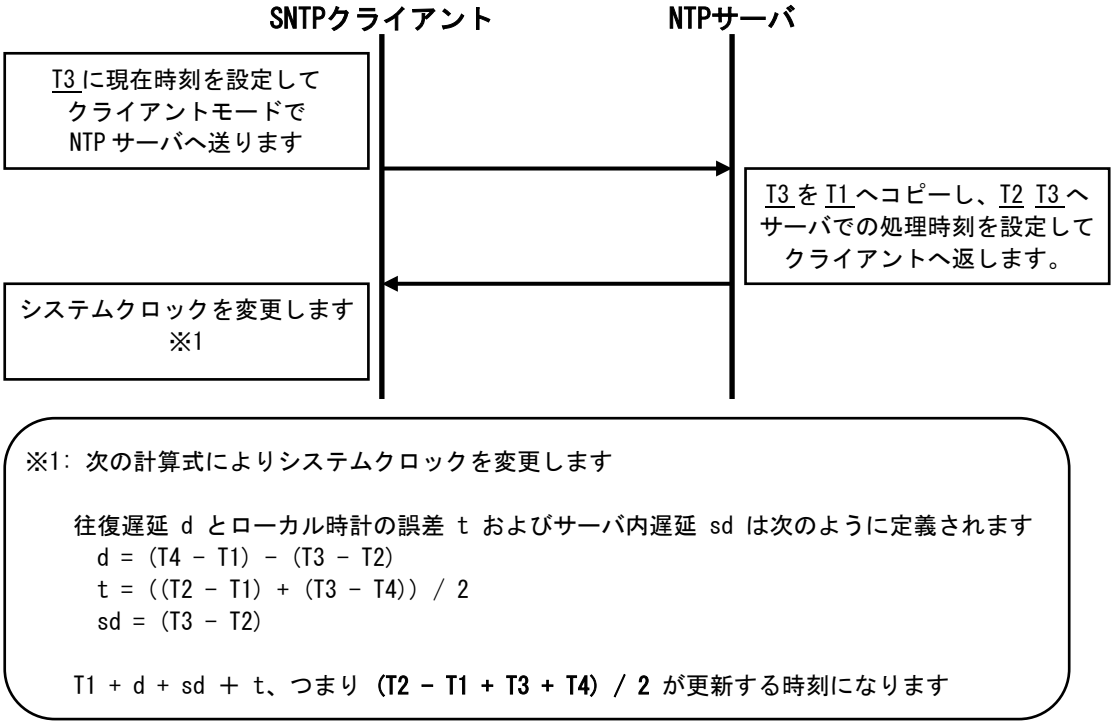

図 1.2　時刻同期処理フロー

1.5　時間の更新について

本 SNTP クライアントで標準提供している時間更新機能はシステムクロックに対して行われます。そのため、時間更新部分ではシステムクロックを直接操作（μITRON の set_tim）します。したがってユーザがシステムクロックを直接操作した場合、システムクロックの正しい時間更新は保証できません。

時間更新部分はソースファイルとして公開しています。システムクロック以外のクロックソースを SNTP クライアントに更新させたい場合は、時間更新部分を変更してください。

また、時間更新処理は組み込むシステム事に微調整を行って下さい。詳細は、2.2 時刻更新インターフェースで、説明します。

2. SNTP 使用方法

2.1 SNTP クライアントのタスク情報

SNTP クライアントには、CLIENT モード、BROADCAST モード、DUAL モードの 3 種類の製品モードと、SERVER モードの動作確認テスト用（無保証）モードがあります。

SNTP クライアントを使うためには、SNTP タスク情報テーブル（sntpConfTbl）に各情報を登録しなければなりません。sntpConfTbl には SNTP クライアントが動作上必要とする動作情報、OS リソース情報が含まれます。これらは使用する OS が有する機能に応じて意味合いが異なります。使用する OS に合わせて登録してください。

・オブジェクトの動的な生成（ID番号番号自動割付け機能を含む）・削除機能を有するOSの場合
cTskPri, cStkAdr, cStkSz をパラメータとしたタスクを ID=taskId としてまた、イベントフラグを ID=flgId として動的に生成し使用します。

・オブジェクトの動作な生成・削除機能を有さないOSの場合
cTskPri, cStkAdr, cStkSz をパラメータとしたタスクを ID=taskId としてまた、イベントフラグを ID=flgId として予め OS に登録されている前提で使用します。

```
typedef struct {
        UINT    cMskLvl;        SNTPクライアント割込みマスクレベル
        PRI     cTskPri;        SNTPクライアントタスクの優先度
        ID      taskId;         SNTPクライアントタスクのタスクID
        VP      cStkAdr;        SNTPクライアントタスクのスタック領域先頭アドレス
        INT     cStkSz;         SNTPクライアントタスクのスタックサイズ
        ID      flgId;          SNTPクライアントが使用するイベントフラグID
} sntpConfTbl;
```

（1） cMskLvl　　SNTPクライアント割込みマスクレベル

SNTPクライアント内部の割り込みマスクレベルを設定します。

1 以上でカーネル割り込みマスクレベル以下の値を設定してください。

1 ≦ SNTPクライアント割り込みマスクレベル ≦ カーネル割り込みマスクレベル

（2） cTskPri　　SNTPクライアントタスクの優先度

SNTPクライアントタスクの優先度を設定します。

TCP/IPマネージャのタスク優先度より低く設定してください。

SNTPクライアントタスクの優先度 ＜ TCP/IPマネージャタスクの優先度

TCP/IPマネージャの優先度より低い優先度でなるべく高い優先度に設定してください。TCP/IPマネージャの優先度より高い場合の動作は保証されません。また、SNTPクライアントタスクより高い優先度のタスクが時間更新処理に割り込んで実行されると時間の制度が低下するので、ご注意願います。

（3） taskId　　SNTPクライアントタスクのタスクID

SNTPクライアントタスクのタスクIDを設定します。

taskIdに0を指定すると、空いているタスクIDを探して登録します（オブジェクトの動的生成機能およびID番号自動割付け機能を有するOSの場合のみ）。

（4） cStkAdr　　SNTPクライアントタスクスタック領域の先頭アドレス

SNTPクライアントタスクスタックに使用する領域の先頭アドレスを設定します。

(5)　cStkSz　　SNTPクライアントタスクのスタックサイズ

SNTPクライアントタスクのスタックサイズを設定します。

(6)　flgId　　SNTPクライアントが使用するイベントフラグID

SNTPクライアントタスク用のイベントフラグIDを設定します。

flgIdに0を指定すると、空いているイベントフラグIDを探して登録します（オブジェクトの動的生成機能およびID番号自動割付け機能を有するOSの場合のみ）。

2.2　時刻更新インターフェース

SNTP クライアントは NTP サーバから取得した情報をもとに、時刻を更新します。
SNTP クライアントは、システムの時刻を取得する場合は外部関数 sntp_getTime を、更新する場合は外部関数 sntp_setTime を呼び出します。
sntp_setTime と sntp_getTime は sntptime.c にありますので、必要に応じて調整してください。

```
/*=========================================================================
 *  Name:       sntp_setTime
 *  Function:   Settting time API
 *  Return:     0 : normal end (clock synchronized)
 *              1 : unavailable time stamp
 *              -1: unavailable packet received
 *=========================================================================
 */
W   sntp_setTime( double tick, T_SNTP_NTIME *tp )
{
    SYSTIM  pk_tim;
    double  tim48 = 0xffffffffL;
    double  tmp_tim;

    if(tp->li == 3)
        return -1;

    tim48 = tim48 + 1;
    tmp_tim = tick / tim48;
    pk_tim.utime = (unsigned long)tmp_tim ;

    tmp_tim = tick - ((double)pk_tim.utime * tim48);
    pk_tim.ltime = (unsigned long)tmp_tim ;

    set_tim( &pk_tim );

    return 0;
}

/*=========================================
 *  Name:       sntp_getTime
 *  Function:   Getting time API
 *  Return:     none
 *=========================================================================
 */
void sntp_getTime( double *timer )
{
    SYSTIM  pk_tim;
    double  tim48 = 0xffffffffL;
    double  tmp_tim;

    get_tim(&pk_tim);

    tim48 = tim48 + 1;
    tmp_tim = (double)pk_tim.ltime;
    tmp_tim = tmp_tim + ((double)pk_tim.utime * tim48);

    *timer = tmp_tim;
}
```

閏秒指示子liが3の場合、サーバ側の時計が同期していない事を示します。
受信した時刻メッセージを無視する場合はリターン値"-1"で終了してください。
サーバからの応答受信が無効と判断されるとユーザが設定した条件のリトライ処理を行います。

SNTPクライアントはタスク起動時に自システムの時計が非同期状態として初期化されますが、リターン値"0"で終了すると自システムの時計が同期済状態として処理します。
もし、SNTPクライアントに同期済状態と認識させたくない場合はリターン値"1"で終了してください。

sntp_setTime 関数のパラメータについて

（1）T_SNTP_NTIME *tp

```
typedef struct {
    unsigned char   li;             閏秒指示子
    unsigned char   ver;            バージョン番号
    unsigned char   mode;           モード
    unsigned char   stt;            階層
    double          t1;             開始タイムスタンプ　T1 （ミリ秒）
    double          t2;             受信タイムスタンプ　T2 （ミリ秒）
    double          t3;             送信タイムスタンプ　T3 （ミリ秒）
    double          t4;             終了タイムスタンプ　T4 （ミリ秒）
    double          rd;             往復遅延　d （ミリ秒）
    double          lco;            ローカル時計の誤差　t （ミリ秒）
} T_SNTP_NTIME;
```

li：受信した NTP パケットの閏秒指示子

値	意味
0	警告無し
1	最後の 1 分が 61 秒
2	最後の 1 分が 59 秒
3	警告状態(時計が同期していない)

ver：受信した NTP パケットのバージョン番号

値	意味
3	NTP バージョン３
4	NTP バージョン４

mode：受信した NTP パケットのモード

値	意味
0	予約
1	対称能動
2	対称受動
3	クライアント
4	サーバー
5	ブロードキャスト
6	NTP 制御メッセージのため予約
7	私的使用のため予約

stt：受信した NTP パケットの階層

値	意味
0	不明または、有効でない階級
1	1 次参照(電波時計などの時刻を参照している)
2～15	2 次参照(NTP、SNTP を経由して時刻を参照している)
16～255	予約

t2 が非 0（サーバからの応答）の場合は、
　　往復遅延 rd = (t4-t1)-(t3-t2)　　ローカル時計の誤差 lco = ((t2-t1)+(t3-t4))/2
t2 が 0（サーバからのブロードキャスト）の場合は、
　　往復遅延 rd = 0　　ローカル時計の誤差 lco = 0

（2）double tick
t2 が非 0（サーバからの応答）の場合は、**更新時刻** tick = (t2-t1+t3+t4)/2
t2 が 0（サーバからのブロードキャスト）の場合は、**更新時刻** tick = t3

sntp_setTime では、引数として SNTP の時刻である１９００年からの通算時間（単位 msec）を受け取り、その値を ITRON の 48bit クロックへ変換してシステムクロックを更新します。

sntp_getTime では、ITRON のシステムクロックである 48bit のデータを取得して、その値を SNTP の時刻である１９００年からの通算時間（単位 msec）に変換して引数 timer の示すアドレスに設定します。

基準時刻を変更する場合は、オフセット値を加算してください。

sntp_setTime と sntp_getTime は SNTP クライアントタスクから呼び出されます。そのため、sntp_setTime や sntp_getTime を変更した場合は、変更によって増加したスタック使用量を SNTP タスク情報テーブルのタスクのスタックサイズ(sntpConfTbl の cStkSz)に加算してください。

【注意】

標準の sntptime.c では ITRON のシステムクロックである 48bit のデータは、SNTP の時刻と同じく１９００年からの通算時間（単位 msec）として参照と設定を行っています。ただし、本製品において有効なシステムクロックの範囲は、UTC 時刻の 1968 年 1 月 20 日 03:14:08 から 2104 年 2 月 26 日 09:42:23 までであり、有効範囲外の時刻が設定されている状態で本製品を動作させた場合は、本製品内部での初期時刻データとして UTC 時刻の 1968 年 1 月 20 日 03:14:08 を使用します。

尚、システムクロックの値が範囲外の場合は、伝搬遅延時間を０として扱い、NTP サーバから取得した時刻を補正せずに設定します。

3. SNTP クライアント API 関数

本製品ではAPI(Application Interface)として、SNTPクライアントの初期化、起動、停止を行うためのサービスコールを準備しています。

3.1 SNTP サービスコール

本節では、SNTPサービスコールについての詳細な説明を以下の形式で行っています。

（No.）関数名　　機能

【発行可能なシステム状態*1】

C言語インタフェース

関数の呼出し形式

パラメータ

型	パラメータ	パラメータの意味
・	・	・
・	・	・

リターンパラメータ

型	パラメータ	パラメータの意味
・	・	・
・	・	・

リターン値

リターン値	リターン値の意味*2
・	・
・	・

パケットの構造

struct {
　　・・・
　　・・・

解　説

　・・・・・・・・・・・・

*1 発行可能なシステム状態を以下のアルファベットで示します
　Ｔ：タスク実行状態
　Ｄ：ディスパッチ禁止状態
　Ｌ：ＣＰＵロック状態
　Ｉ：非タスク部実行状態
　なお、各状態の詳細は各ITRONのユーザーズマニュアルを参照してください。
　発行可能なシステム状態以外の状態でインタフェース関数をコールした場合、システムの正常な動作は保証されません。
*2 エラーの理由として、アドレスが4の倍数以外、アドレスが奇数についてのエラーは、奇数アドレスからの16ビットや32ビットアクセスが可能なマイコン向けの製品では発生しません。

3.1.1 SNTP_init SNTP クライアント初期化

【T／D／L／I】

C言語インタフェース

void = SNTP_init (void);

パラメータ

無し

リターンパラメータ

無し

解　説

SNTPクライアントを初期化します。

SNTPクライアントの内部変数を初期化し、SNTPクライアントを使用する為の準備をします。

SNTP_init は、SNTPクライアント使用前に 1 回だけ必ず実行してください（初めてSNTP_startを呼び出す前に 1 回だけ実行し、その後は実行しないでください）。

3.1.2 SNTP_start SNTP クライアント起動

【T】

C言語インタフェース

```
ER ercd = SNTP_start(T_SNTP_PAR *par );
```

パラメータ

T_SNTP_PAR	*par;	SNTPクライアント初期化パラメータ

リターンパラメータ

ER	ercd	リターン値
UW	par->memlen;	使用したメモリの長さ
H	par->perrno;	E_PAR発生時の詳細エラーコード

パケットの構造

```
struct {
        UW      myipaddr;       SNTPクライアントのIPアドレス
        UW      svipaddr;       NTPサーバのIPアドレス
        UW      bcipaddr;       ブロードキャスト用のIPアドレス
        UH      mode;           SNTPクライアントの動作モード
        ID      cepid;          使用するUDP通信端点ID
        H       respwait;       応答待ち時間（単位：秒）
        H       retrycnt;       リトライ回数
        H       interval;       要求時間間隔
        H       perrno;         エラー詳細情報
} T_SNTP_PAR;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（詳細エラーコード参照）
E_OBJ	オブジェクト状態不正（すでに動作開始している）
E_ILUSE	サービスコール不正使用（TCP/IPマネージャが動作していない）

詳細エラーコード（E_PAR発生時）

ESNTP_PARADR	1	par が4の倍数以外
ESNTP_MYIP	2	myipaddr が不正（0またはffffffff，IPアドレス未登録）
ESNTP_RMIP	3	svipaddr が不正（svipaddr＝0）
ESNTP_MODE	4	mode が不正（0,1,2,10以外）
ESNTP_CEPID	5	cepidが不正（cepid＜0 または cepid＞128）
ESNTP_WAIT	6	respwaitが不正（respwait≦0）
ESNTP_RCNT	7	retrycntが不正（retrycnt＜0）
ESNTP_TIME	8	intervalが不正（interval≦0）
ESNTP_SYS	20	tcpipConfTbl の設定値が不正
ESNTP_CRECEP	30	UDP通信端点の生成エラー（使用中 または 空きなし）
ESNTP_CREFLG	31	イベントフラグの生成エラー（sntpConfTbl.flgId）
ESNTP_CRETSK	32	タスクの生成エラー（sntpConfTbl.taskId）
ESNTP_STATSK	33	タスクの起動エラー

解説

SNTPクライアントを起動します。
SNTPクライアントタスクを生成、起動し、SNTPクライアント動作を開始させます。
本サービスコールは、SNTPクライアントタスクの起動処理後、直ちにリターンします。

parが0または4の倍数以外の場合は、エラーコードとしてE_PARを返します。この場合はperrnoに詳細エラーコードの設定は行いません。E_PAR発生時に必ずperrnoを参照する場合は、本サービスコールを呼び出す側でperrnoに0を設定してください。

myipaddrには、SNTPクライアントのIPアドレスを指定します。
myipaddrに0を指定すると、TCP/IPマネージャに登録されているIPアドレスのうち、最も小さいIDを持つIPアドレスを自動で選択して使用します。
myipaddrで指定したIPアドレスがTCP/IPマネージャに登録されていない場合（0指定で登録されているIPアドレスが無い場合を含む）は、エラーコードとしてE_PARを返し、perrnoにESNTP_MYIPを返します。

svipaddrには、NTPサーバのIPアドレスを指定します。svipaddrが不正の場合は、エラーコードとしてE_PARを返し、perrnoにESNTP_RMIPを返します。svipaddrは、BROADCASTモードでは使用しません。

bcipaddrには、時刻情報をブロードキャストするIPアドレスを指定します。bcipaddrは、SERVERモード以外では使用しません。

modeには、SNTPクライアントの動作モードを設定します。

cepidには、使用するUDP通信端点のIDを指定します。cepidに0を指定すると空いているUDP通信端点を探して使用し、cepidを使用したUDP通信端点のID値に書き換えます。

(1) CLIENTモード

modeにSNTP_CLIENT(0)を指定した場合は、CLIENTモードとして動作します。
SNTP タスクを生成し、NTP サーバに対して時刻の要求を行います。
指定した NTP サーバより正しい応答を受け取ると、ユーザ作成の sntp_setTime 関数を呼び出してから SNTP タスクを終了します。
respwait 時間内に正しい応答を受け取れない場合はタイムアウトとなりますが、retrycnt 分のリトライを行った後、SNTP タスクを終了します。

(2) BROADCASTモード

mode に SNTP_BROADCAST(1)を指定した場合は、BROADCAST モードとして動作します。
SNTP タスクを生成し、NTP サーバからのブロードキャスト（IP マルチキャスト）で送られてくる時刻同期要求を受け取るたびにユーザ作成の sntp_setTime 関数を呼び出します。
尚、SNTP タスクは NTP サーバに対する時刻の要求を行いません。
SNTP_stop を実行すると、SNTP タスクを終了します。

(3) DUALモード

modeにSNTP_DUAL(2)を指定した場合は、DUALモードとして動作します。
SNTP タスクを生成し、NTP サーバからのブロードキャスト（IP マルチキャスト）で送られてくる時刻同期要求を受け取るたびにユーザ作成の sntp_setTime 関数を呼び出します。
また、一定時間経過毎に NTP サーバへ時刻の要求を行って、ユーザ作成の sntp_setTime 関数を呼び出します。
SNTP_stop を実行すると、SNTP タスクを終了します。

(4) SERVERモード**（動作確認用擬似サーバ）**

mode に SNTP_SERVER(10)を指定した場合は、SERVER モードとして動作します。
SNTP タスクを生成し、SNTP クライアントからの時刻要求に応答します。
svipaddr が 0 以外の場合、一定時間経過毎に NTP サーバへ時刻の要求を行って、ユーザ作成の sntp_setTime 関数を呼び出します（svipaddr が 0 の場合、自システムの時刻は同期済みとして動作するため sntp_setTime 関数を呼び出しません）。
bcipaddr が 0 以外の場合、一定時間経過毎に時刻情報をブロードキャストで送信します。
SNTP_stop を実行すると、SNTP タスクを終了します。
※ SERVER モードは動作確認用の擬似 SNTP サーバとしてのみ使用してください。
※ SERVER モードにおけるクライアントへの応答パケットとブロードキャストパケットの各フィールドは固定値として設定されます（精度：-16、ルート遅延：0、ルート分散：0、参照識別子：LOCL、ブロードキャストパケットのポーリング間隔：4）。

respwaitには、サーバに送信した時刻要求に対するサーバからの応答を待つ時間を秒単位で指定します。
respwaitに指定した時間内にサーバからの応答がない場合はタイムアウトとなり、リトライ処理を行います。尚、サーバからの応答があっても不正な受信として無視されるとrespwaitに指定した時間後にタイムアウトとなります。

retrycntには、応答待ちのタイムアウトが発生した場合のリトライ回数を指定します。

respwaitとretrycntは、BROADCASTモードおよびSERVERモードでsvipaddrに0を指定した場合は使用しません。

intervalには、DUALモードにおけるNTPサーバへ時刻要求の間隔、SERVERモードにおける上位のNTPサーバへ時刻要求の送信およびブロードキャストによる時刻送信の間隔を秒単位で指定します。この処理間の時間間隔とは別に前回の処理開始から処理終了までの時間が加算されて一連の処理サイクル時間となります（処理開始から処理終了までの時間は一定ではありません）。
intervalは、CLIENTモードとBROADCASTモードおよびSERVERモードでbcipaddrに0を指定した場合は使用しません。

SNTPクライアントは、NTPサーバから、各動作モードにあった正しいフォーマットのNTPパケットを受け取るとユーザ作成のsntp_setTime関数を呼び出しますが、NTPパケット内の詳細情報を判断して実際に時刻更新を行うかはユーザの任意です。その為、時刻更新が行われたかどうかについては、ユーザ作成のsntp_setTime関数によって判断してください。

TCP/IPマネージャが動作していない場合は、エラーコードとしてE_ILUSEを返します。
すでに、SNTPクライアントが動作を開始していた場合は、エラーコードとしてE_OBJを返します。

3.1.3 SNTP_stop SNTP クライアント停止

【T】

C 言語インタフェース

```
ER ercd = SNTP_stop ( void );
```

パラメータ

なし

リターンパラメータ

ER	ercd	リターン値

リターン値／エラーコード

E_OK	正常終了
E_OBJ	オブジェクト状態不正（すでに停止している）

解説

SNTPクライアントを停止します。

SNTPクライアントのサービスを停止後、SNTPクライアントタスクを終了して削除します。

HI.CommunicationEngine
SNTPクライアント リファレンスマニュアル
CM7000SNT02J-6

発行年月　　2014年　7月　　第6版
発　　行　　ルネサスセミコンダクタパッケージ＆テストソリューションズ株式会社
編　　集　　ルネサスセミコンダクタパッケージ＆テストソリューションズ株式会社