TOPPERS 活用アイデア・アプリケーション開発

コンテスト

部門	：	活用アイデア部門　アプリケーション開発部門
作品のタイトル	：	TOPPERS Realtime System Sample (RSS) – LPCXpresso GPS Clock
作成者	：	中村 晋一郎（個人）
対象者	：	RTOS 初心者向け、リアルタイム・システム初心者向け
使用する開発成果物	：	TOPPERS/ASP for LPC (酔漢氏による成果物)

目的・狙い

TOPPERS には sample1 なるサンプルが存在します。 ユーザは、sample1 という名前から TOPPERS を使ったアプリケーションのサンプルである事を期待します。 しかし、実際はリアルタイムシステムを構築する上で必要不可欠なタスクの分割設計や、タスク間通信の使い方などの情報を得る事ができません。 TOPPERS は教育用途としても提供されますが、ソフトウェア開発を学ぶ人たちにとって有益な実装形態とは言い難いものがあります。

そこで、TOPPERS のサンプルとして機能するシンプルなリアルタイム・システムの事例として、簡単な GPS 時計アプリケーションを実装しました。 固定長メモリプールとメールボックスを組み合わせたタスク間通信、優先度設計のポイントなど、リアルタイム・システムの基礎を学習できます。 複数ファイルから構成されるプロジェクトのコンフィギュレータ使用方法も同時に示すことで、プロジェクトの雛形としても使用できるようにしました。

アプリケーションの概要

この GPS 時計アプリケーションは、初期化タスク、シェルタスク、GPS タスク、ディスプレイ・タスクの 4 つのタスクから構成されています。 システムにはダイナミック点灯制御が必要な LED 表示器が搭載されており、六桁の数字を一桁ずつ数ミリ秒毎に切り替えながら表示制御します。 一桁毎の表示時間長は輝度に影響を与えるので常に一定間隔で表示制御する必要があります。 ここに TOPPERS のリアルタイム性が活かされています。 この表示制御より低いプライオリティで動作しているのが GPS タスクとシェル・タスクです。

はじめに

今回の成果物のターゲットについてはじめに簡単に説明します。LPCXpresso は、NXP セミコンダクターズの ARM Cortex-M シリーズのマイコン評価基板で、Embedded Artists 社が設計しているものです。 LPCXpresso GPS Clock は、高輝度 7 セグメント LED 駆動回路と GPS モジュールを LPCXpresso に接続して実現しました。

システム全景

図 0 にシステム全景を示します。LPCXpresso GPS Clock 基板の拡張端子に GPS モジュールと USB シリアル変換モジュールを接続している様子です。

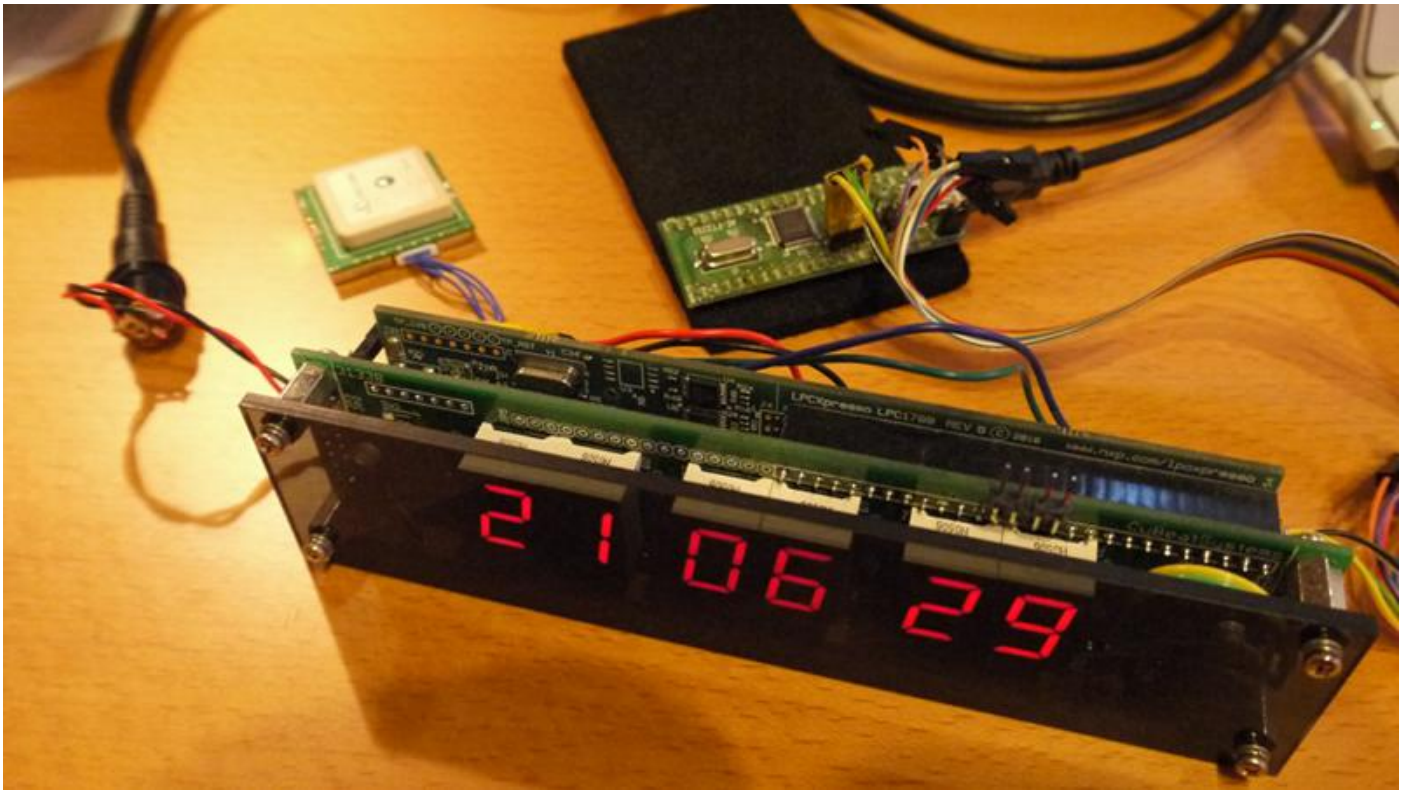

図 0．システム全景

アプリケーション

アプリケーションのタスク間の関係を図１に示します。

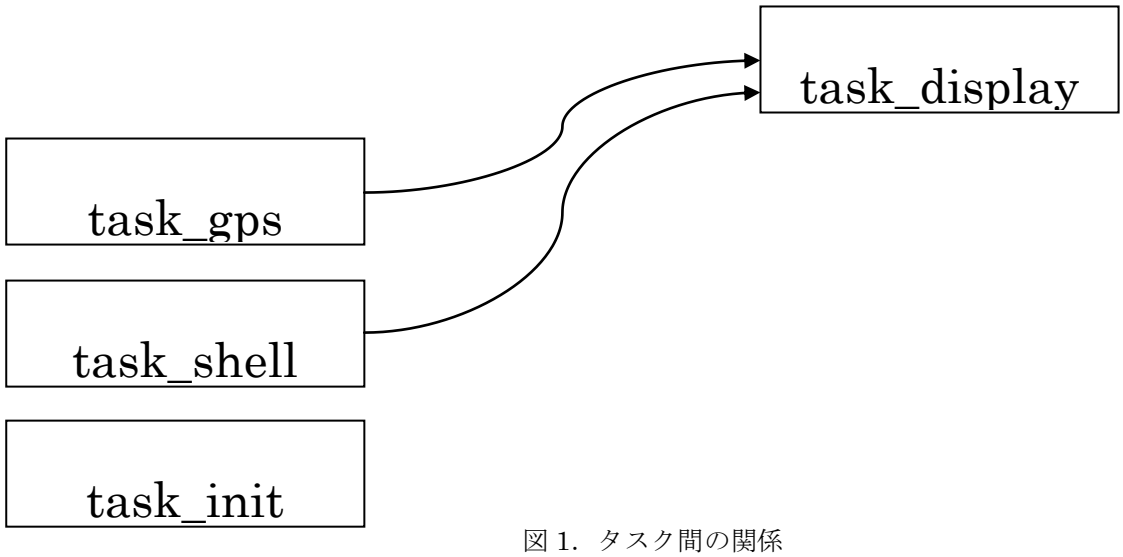

図 1. タスク間の関係

GPS タスク(task_gps)は、GPS モジュールから時刻情報を取り出し、ディスプレイタスク(task_display)にタスク間通信で表示内容を渡します。この表示内容を渡す方法は、固定長メモリプールとメールボックスを使い、固定長メモリプールの資源の範囲であればこれら二つのタスク（GPS タスクとディスプレイタスク）の動作を阻害しないように構成してあります。リアルタイム・システムにおいて、適切なタスク間通信を用いる事が非常に重要です。このサンプルを示すことで初学者がリアルタイム・システムの設計を学べるようにしました。これが本アプリケーションのサンプルプログラムとしての第１のポイントです。

次に、シェルタスク(task_shell)は、シリアルポート経由でユーザからコマンドを受け付け、システムに制御を加えます。今回はシステム情報を取得するための「info コマンド」と、ディスプレイに所望の表示を行なうための「display コマンド」を用意しました。ディスプレイタスクへのリクエストは、GPS タスクで使用していたものと同じメカニズムを使用します。小規模リアルタイム・システムにおいて、使いやすいシェルの存在は貴重です。シェルを経由してユーザのリクエストを受け取ったシステムが、他のタスクでも使用しているタスク間通信のメカニズムを再利用して必要な機能を実現しています。組み込みシステムにおいて、同じ機能を実現する場合に再利用可能なメカニズムを用いて機能実現する事は、リソース面、品質面などで見て非常に有効な方法です。こういったサンプルを提供する事が初学者にとって学習効果の高いものと考えました。これが本アプリケーションのサンプルプログラムとしての第２のポイントです。

参考までに、図２に実行中のシェルの様子を示しました。

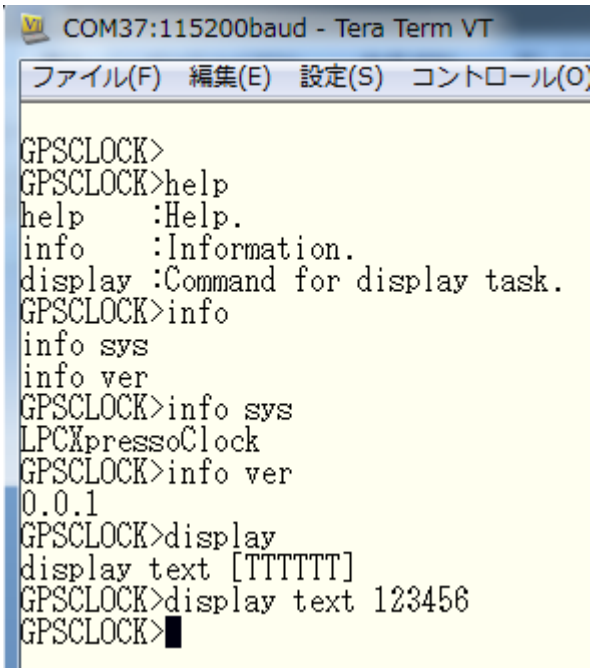

図２．コマンドを実行している様子

実装上の工夫

sample1 は、一本のソースに全てが詰め込まれていました。提案アプリケーションでは、幾つかの工夫を凝らしてあります。図３に、タスク毎に分割されたファイルの様子について示します。TOPPERS カーネルを初めて使うユーザがリアルタイム・システムを構築する場合、どのようにタスク分割をするのか？、どのようにタスク間通信を行なうのか？、など様々な疑問にぶつかる事が考えられます。提案アプリケーションでは、タスク毎に分割実装されたプロジェクトで機能を実装してあります。提案アプリケーションを雛型なすれば、TOPPERS カーネルの初めてのユーザでも、簡単にタスクを追加、削除する事ができるようになります。

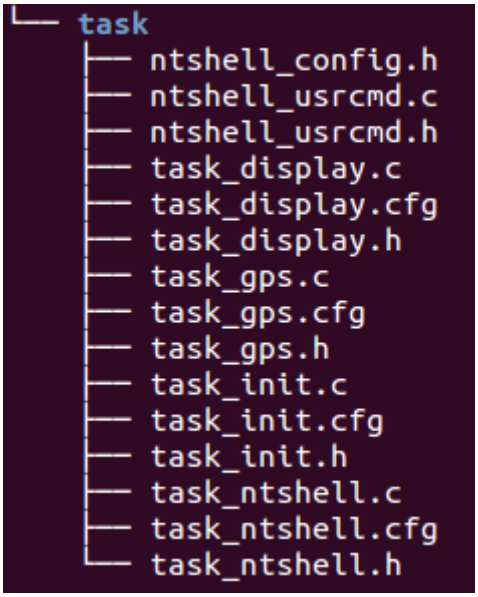

図３．タスク毎に分割されたファイル

また、タスクで使用する細かなロジックはライブラリとして別のディレクトリ構成としました。このようにすることで、タスクの実装が単純になり、再利用性も高まります。また、下層ライブラリとその上位に位置するタスクという論理的な階層構造をディレクトリ分割で表現する事で、アプリケーション設計における階層設計を意識できるようにしてあります。実際に RTOS の初心者は、ソフトウェアの初心者である事も多く、このような細かな配慮で提供されているコードを見て RTOS と一緒にソフトウェアを学習してもらうのが一番効率的と考えました。

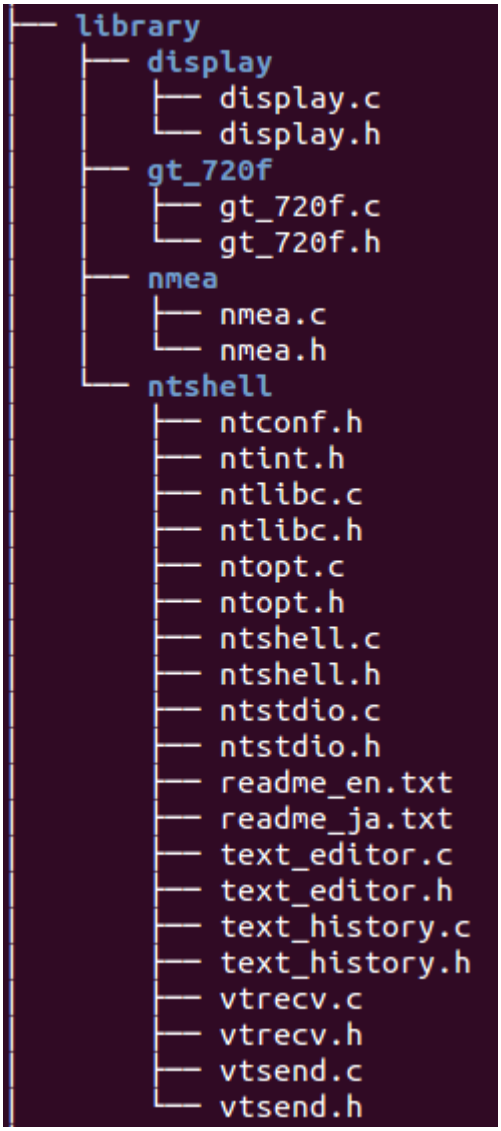

図４．階層化されたライブラリ

提案アプリケーションでは、タスクとそのタスクへのリクエスト用 API の定義を同一ヘッダで提供するようにしました。図５に示したのが、ディスプレイ・タスクにおける API です。このコードは、ディスプレイタスクのヘッダです。task_display 関数は、ディスプレイタスクの本体です。tskapi_display_update 関数は、ディスプレイタスク以外のタスクがディスプレイタスクに要求をするための API です。内部では、固定長メモリプールとメールボックスが組み合わせて使われていますが、その事を意識せずにシステムを構成できるように API 内部に隠ぺいしてあります。

```
void task_display(intptr_t exinf);

void tskapi_display_update(
        uint8_t h10,
        uint8_t h01,
        uint8_t m10,
        uint8_t m01,
        uint8_t s10,
        uint8_t s01,
        uint8_t col);
```

図５．ディスプレイタスク本体関数とリクエスト用関数

図６にディスプレイタスクへのリクエスト用関数の実装を示します。固定長メモリプールとメールボックスを組み合わせたタスク間通信は、TOPPERS を使ったシステムをリアルタイム・システムとしてきちんと動作させるための定石とも言えますが、これらを知らないあまり、きちんとリアルタイム・システムとして実現されていないケースも見受けられます。サンプルではこういった事例も示し、リアルタイム・システムの特徴を最大限にシステムに取り入れられる人材を育成したいと考えました。

```
/**
 * @brief ディスプレイの更新を要求する。
 * @details この関数はディスプレイタスクにリクエストを要求したいタスクが呼び出す為のものである。
 */
void tskapi_display_update(
        uint8_t h10,
        uint8_t h01,
        uint8_t m10,
        uint8_t m01,
        uint8_t s10,
        uint8_t s01,
        uint8_t col)
{
    VP vp;

    /*
     * メモリプールからバッファを取得する。
     */
    get_mpf(MPF_DISPLAY, &vp);

    /*
     * コマンドとパラメータを設定する。
     */
    ((display_msg_t *)vp)->cmd = DISPLAY_CMD_UPDATE;
    display_param_update_t *param =
        (display_param_update_t *)&(((display_msg_t *)vp)->param);
    param->h10 = h10;
    param->h01 = h01;
    param->m10 = m10;
    param->m01 = m01;
    param->s10 = s10;
    param->s01 = s01;
    param->col = col;

    /*
     * メールボックスに送信する。
     */
    snd_mbx(MBX_DISPLAY, vp);
}
```

図６．ディスプレイタスクへのリクエスト用関数

ディスプレイタスクの本体側では、メールボックスから受信したメッセージを処理し、固定長メモリプールに資源を戻す処理を実装してあります。

```
display_msg_t *p;
while (1) {
    /*
     * メールボックスの状態を観察し、
     * メッセージがあれば処理を実行する。
     */
    if (prcv_mbx(MBX_DISPLAY, (T_MSG **)&p) == E_OK) {
        uint8_t cmd = ((display_msg_t *)p)->cmd;
        void *param = ((display_msg_t *)p)->param;
        switch (cmd) {
            case DISPLAY_CMD_UPDATE:
                cmd_update((display_param_update_t *)param);
                break;
            default:
                syslog(LOG_NOTICE, "Unknown command 0x%x.", cmd);
                break;
        }
        rel_mpf(MPF_DISPLAY, (VP)p);
    }
    display_7seg_clock(&(work.display));
    tslp_tsk(2);
}
```

図７．ディスプレイタスクの本体

まとめ

TOPPERS の sample1 に代わる「ユーザのためのサンプル」を提示すべく、簡単な GPS 時計アプリケーションを提案しました。 本アプリケーションを提供する事で、優先度設計の重要性、タスク間通信のポイント、実アプリケーションのひな型としての使用などが可能です。 従来の sample1 では難しかった初学者のための学習材料として最適です。

謝辞

酔漢氏がポーティングされている TOPPERS/ASP for LPC には、ユーザが使うための工夫が多く盛り込まれていました。 今回の TOPPERS Realtime System Sample (RSS) - LPCXpresso GPS Clock も、氏の成果物に強く影響を受けています。 素晴らしいポーティング成果物にこの場を借りてお礼申し上げます。

参考文献

TOPPERS/ASP for LPC　http://sourceforge.jp/projects/toppersasp4lpc/