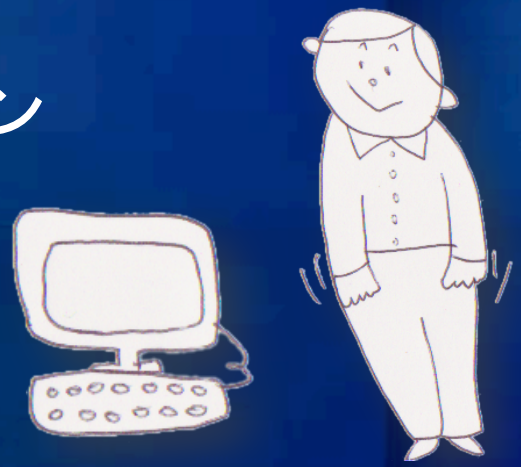
TOPPERS Realtime System Sample (RSS) LPCXpresso GPS Clock

2013年度
TOPPERS活用アイデア・アプリケーション
開発コンテスト
中村 晋一郎



TOPPERS Realtime System Sample (RSS)
内容

- RSSの狙い
- RSSの紹介
- RSSの先に…

RSSの狙い

TOPPERS Realtime System Sample(RSS)
狙い

- リアルタイム・オペレーティング・システムの使い方をサンプルとして提供する事
- ユーザがサンプルを雛型にプロジェクトを始められるようにする事
- その他

リアルタイム・オペレーティング・システムの使い方をサンプルとして提供する事

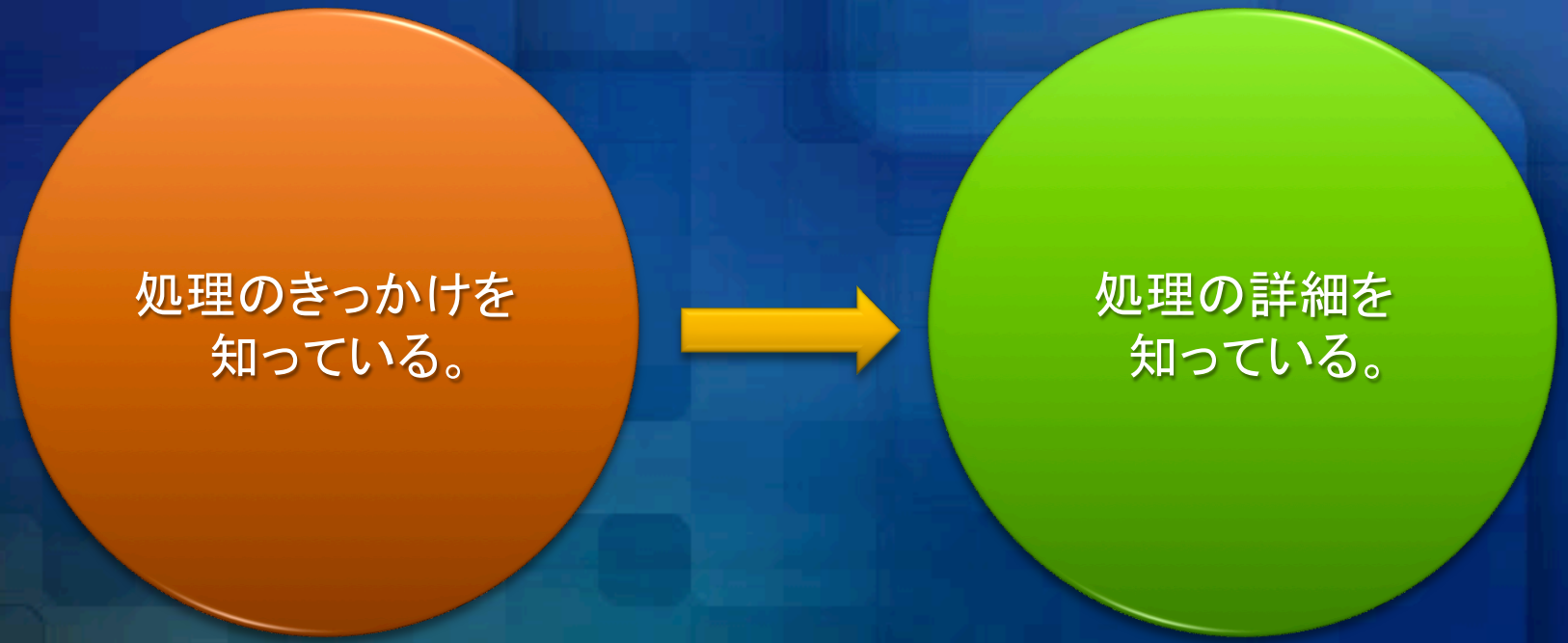
システム全体の処理構造はメッセージパッシングで構成

送信側

- 処理が必要になったタイミングで「やってね！」と要求するだけ。

受信側

- 「やってね！」と要求されたタイミングで処理を実行するだけ。



かつ、送信側と受信側は、互いに動作をブロックしない。

送信側と受信側は
互いに動作をブロックしない事！

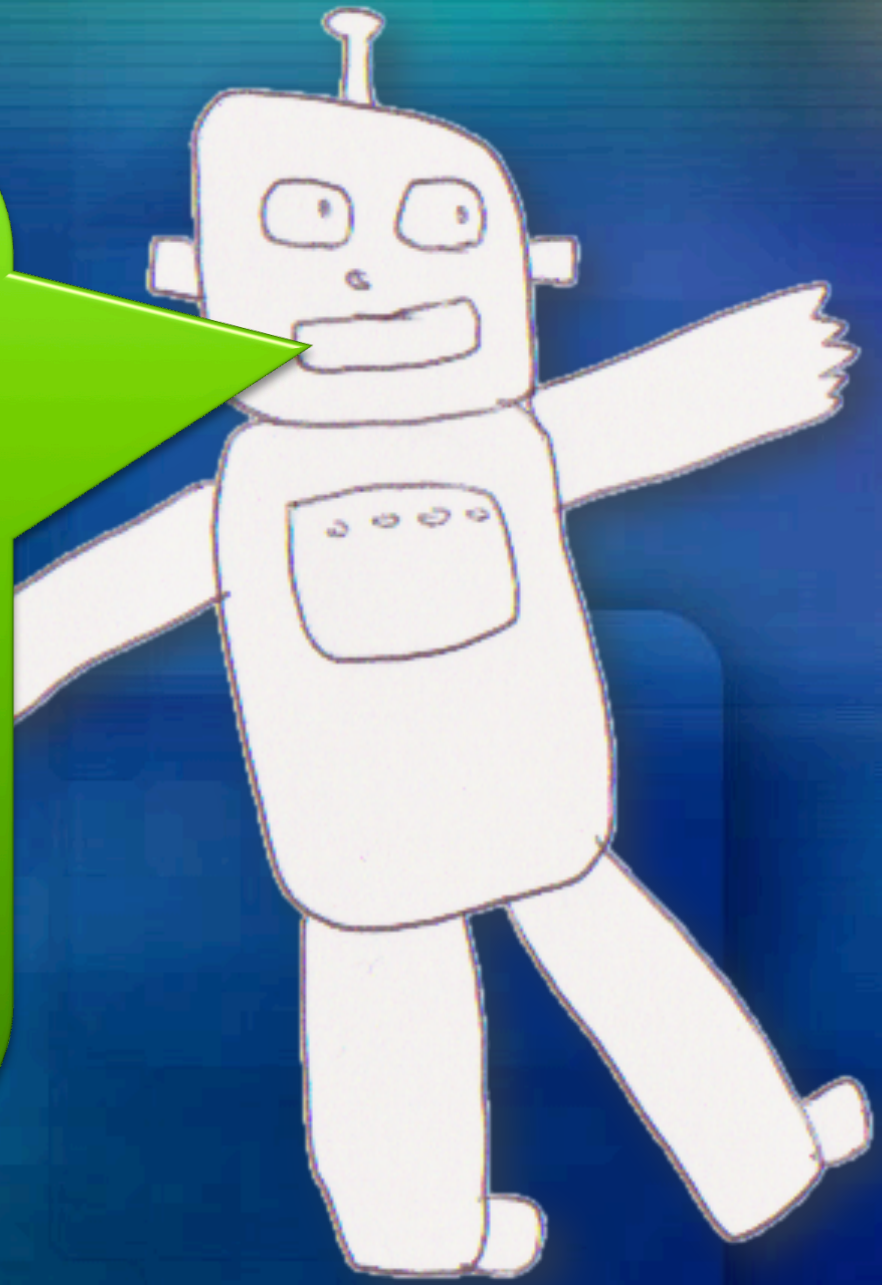
RTOSを使っているからリアルタイムシステムになるという事は無い。
タスクに分けているからリアルタイムシステムになるという事は無い。

意外に多い、
巷の不適切実装例。

教育上の何らかの
配慮かもしれないが…

でも、やっぱり
これは困る。

(ある書籍の例を紹介)



TOPPERSにおける誤った メッセージパッシングの実装例

```
#include <kernel.h>
#include <t_syslog.h>
#include <syssvc/syslog.h>

#include "kernel_cfg.h"
#include "read_multi_io.h"
#include "target_io.h"

/* 読み出した値の格納領域 */
typedef struct
{
    T_MSG *msg;
    uint_t vr2_value;
    uint_t psw_state;
} IO_DATA;

IO_DATA read_value;

void main_task(intptr_t exinf)
{
    ER ercd;

    /* syslog の設定*/
    syslog_msk_log(LOG_UPTO(LOG_INFO) , LOG_UPTO(LOG_EMERG));

    /* 起動画面表示*/
    syslog(LOG_NOTICE, "I                    ");

    ercd = act_tsk(READ_T
    ercd = act_tsk(DISP_T

    ercd = ext_tsk();
    assert(ercd == ercd);
}
```

受信側

資源が唯一！

送信側

某技術系月刊誌より抜粋

メールボックス特有の問題を
タスク状態の遷移で回避。

RTOSの使い方として
筋の良い例とは言えない。

送信タスクと受信タスクの
処理が直列化されている。

この実装は第3のタスクが
現われた時に対応不能。

TOPPERSにおける正しいメッセージパッシングの実装例

送信側

```
void 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;

    /*
     * メモリプールからメモリを取得する。
     */
    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);
}
```

資源獲得

送信

受信側

```
/*
 * メールボックスの状態を観察
 * メッセージがあれば処理を実
 */
if (rcv_mbx(MBX_DISPLAY, (T_MSG **)&p) == E_OK) {
    uint8_t cmd = ((display_msg_t *)p)->cmd;
    void *param = ((display_msg_t *)p)->par
    switch (cmd) {
        case DISPLAY_CMD_UPDATE:
            cmd_update((display_para e_t *)param);
            break;
        default:
            syslog(LOG_NOTICE. wn command 0x%x.", cmd);
            break;
    }
    rel_mpf(MPF_DISPLAY, (VP)p);
}
```

受信

資源解放

メモリプールのバッファを
メッセージバッファとして使用。

資源が使える限り両タスクは
ブロック状態になる事はない。

ユーザがサンプルを雛型にプロジェクトを始められるようにする事

sample1は、カーネル動作確認用

```
43 /*-
44  *  サンプルプログラム(1)の本体
45  *
46  *  ASPカーネルの基本的な動作を確認するためのサンプルプログラム.
47  *
48  *  プログラムの概要:
49  *
50  *  ユーザインタフェースを受け持つメインタスク（タスクID: MAIN_TASK，優
51  *  先度: MAIN_PRIORITY）と，3つの並列実行されるタスク（タスクID:
52  *  TASK1?TASK3，初期優先度: MID_PRIORITY）で構成される．また，起動周
53  *  期が2秒の周期ハンドラ（周期ハンドラID: CYCHDR1）を用いる．
54  *
55  *  並列実行されるタスクは，task_loop回空ループを実行する度に，タスクが
56  *  実行中であることをあらわすメッセージを表示する．
57  *
58  *  周期ハンドラは，三つの優先度（HIGH_PRIORITY，MID_PRIORITY，
59  *  LOW_PRIORITY）のレディキューを回転させる．プログラムの起動直後は，
60  *  周期ハンドラは停止状態になっている．
61  *
62  *  メインタスクは，シリアルI/Oポートからの文字入力を行い（文字入力を
63  *  待っている間は，並列実行されるタスクが実行されている），入力された
64  *  文字に対応した処理を実行する．入力された文字と処理の関係は次の通り．
65  *  Control-Cまたは'Q'が入力されると，プログラムを終了する．
```

sample1は、怒涛の一本書き

初学者がsample1をサンプルにしたら･･･と思うと将来が怖い

- インターフェース定義と実装詳細の関係を混同するエンジニアが育ったらどうしよう･･･
- グローバル変数を多用するエンジニアが育ったらどうしよう･･･
- 適切にモジュール分割出来ないエンジニアが育ったらどうしよう･･･
- もっと言うと、日本のソフトウェアエンジニアのレベルが低下したらどうしよう･･･

sample1に対する所感

- カーネルユーザにとってのサンプルでない
  - 設計意図通りだから指摘すべき事ではないが・・・
- 初学者にとっては悪例となりかねない
  - TOPPERSが提供する資源として好ましくなさそう
- TOPPERSを本当に使って欲しいと考えている？
  - ・・・と疑問に思ってしまう御節介

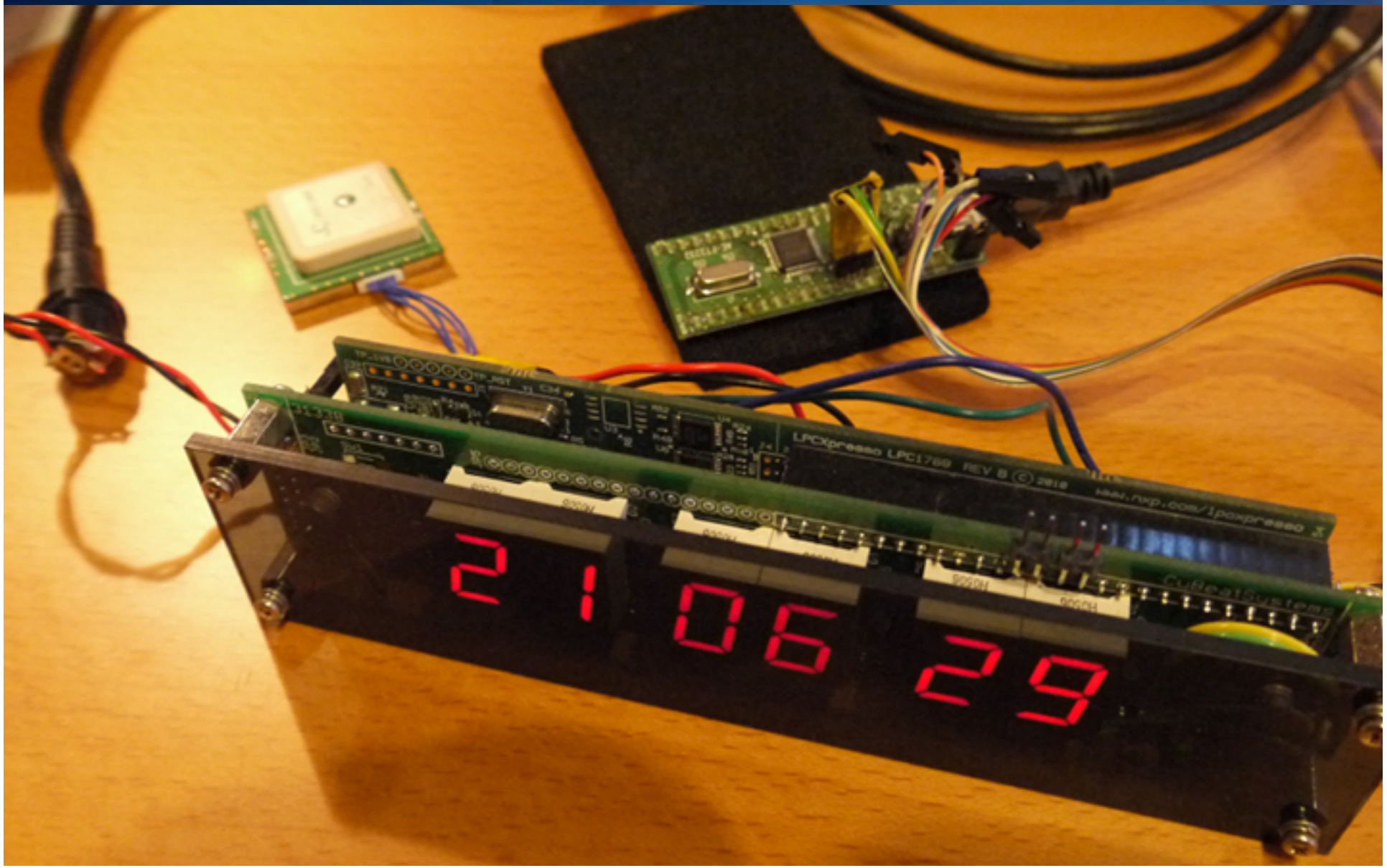
sampleではやっぱりマズイ気がする。
sample1でなく、kernel_testと命名すれば良かったのかも。

RSSの紹介

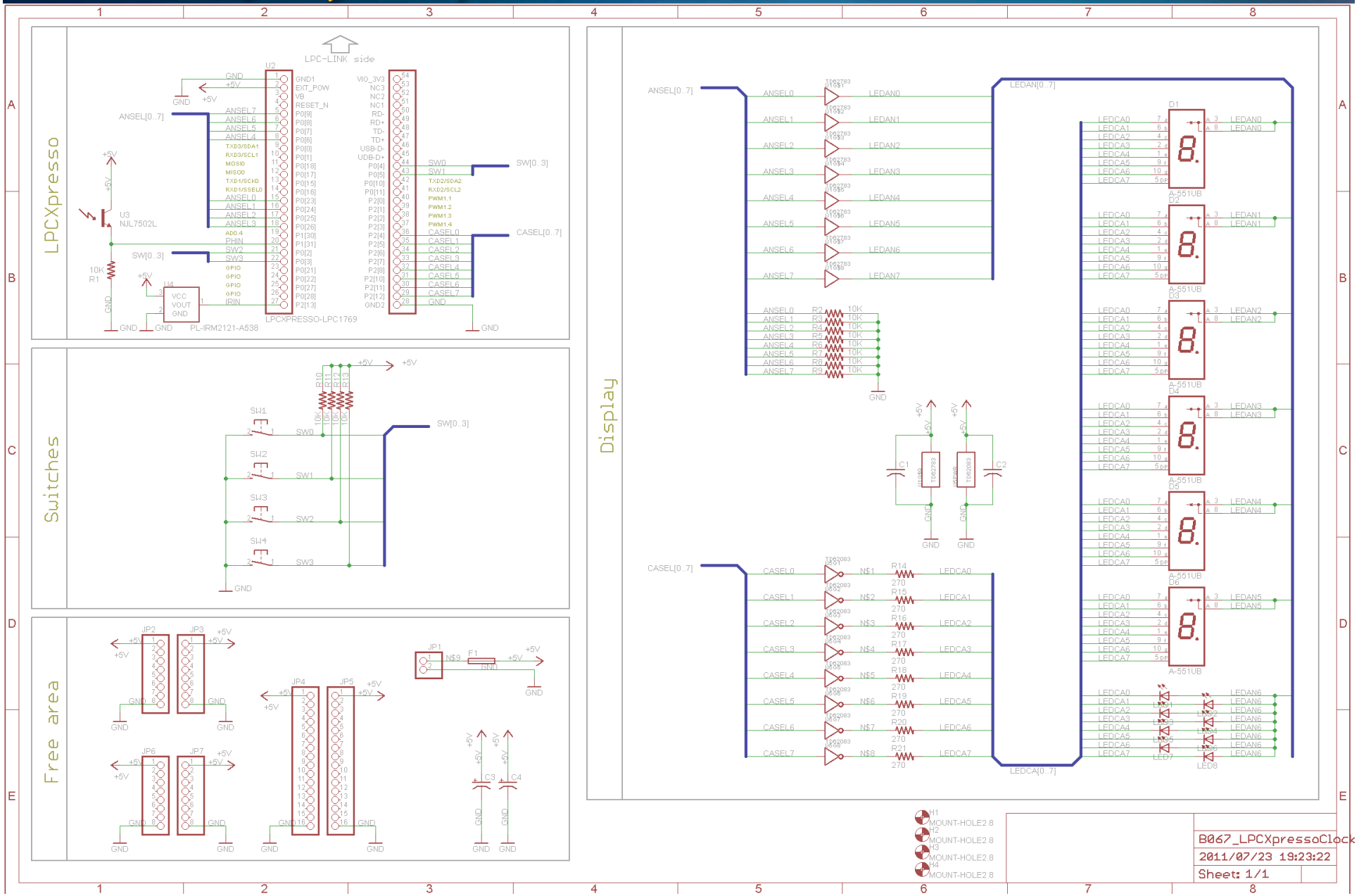
基本コンセプト

・シンプルなハードウェア
・シンプルなファームウェア
・実際に役立つテンプレート

ハードウェア



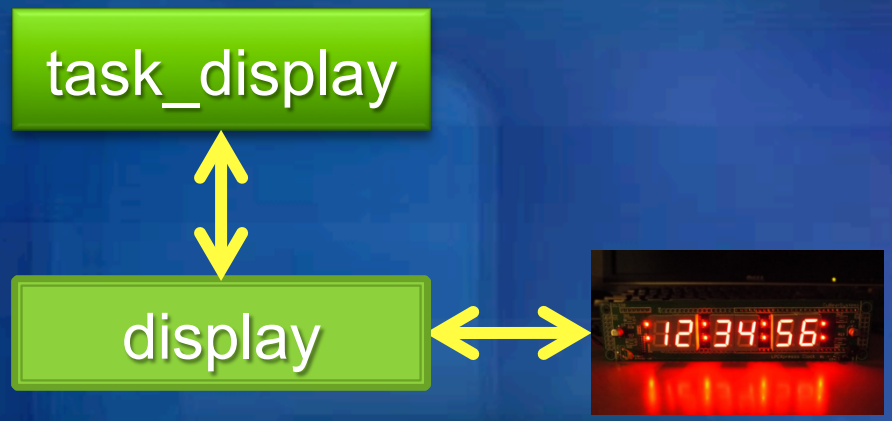
ハードウェア


LPCXpresso
LPC-LINK side
U2
LPCXPRESSO-LPC1769
U3
NJL7502L
U4
PL-IRM2121-A538
ANSEL[0..7]
SW[0..3]
CASEL[0..7]
Switches
SW1
SW2
SW3
SW4
Free area
JP1
JP2
JP3
JP4
JP5
JP6
JP7
F1
C3
C4
Display
LEDAN[0..7]
LEDCA[0..7]
D1
D2
D3
D4
D5
D6
A-551UB
LED7
LED8
H1
H2
H3
H4
MOUNT-HOLE2.8
B067_LPCXpressoClock
2011/07/23 19:23:22
Sheet: 1/1

タスク構成

下層ライブラリとの関係

タスクは下層ライブラリを使ってデバイスを制御するという当たり前の関係を実装で表現



タスクの実装が煩雑にならないメリットをサンプルによって学習可能

適切なレベルの抽象化と階層化

タスク

```
└── task
    ├── ntshell_config.h
    ├── ntshell_usrcmd.c
    ├── ntshell_usrcmd.h
    ├── task_display.c
    ├── task_display.cfg
    ├── task_display.h
    ├── task_gps.c
    ├── task_gps.cfg
    ├── task_gps.h
    ├── task_init.c
    ├── task_init.cfg
    ├── task_init.h
    ├── task_ntshell.c
    ├── task_ntshell.cfg
    └── task_ntshell.h
```

下層ライブラリ

```
├── library
│   ├── display
│   │   ├── display.c
│   │   └── display.h
│   ├── gt_720f
│   │   ├── gt_720f.c
│   │   └── gt_720f.h
│   ├── nmea
│   │   ├── nmea.c
│   │   └── nmea.h
│   └── ntshell
│       ├── ntconf.h
│       ├── ntint.h
│       ├── ntlibc.c
│       ├── ntlibc.h
│       ├── ntopt.c
│       ├── ntopt.h
```

定義と実装詳細の適切な分離

```c
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);
```

定義

実装詳細

```c
/**
 * @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);
}
```

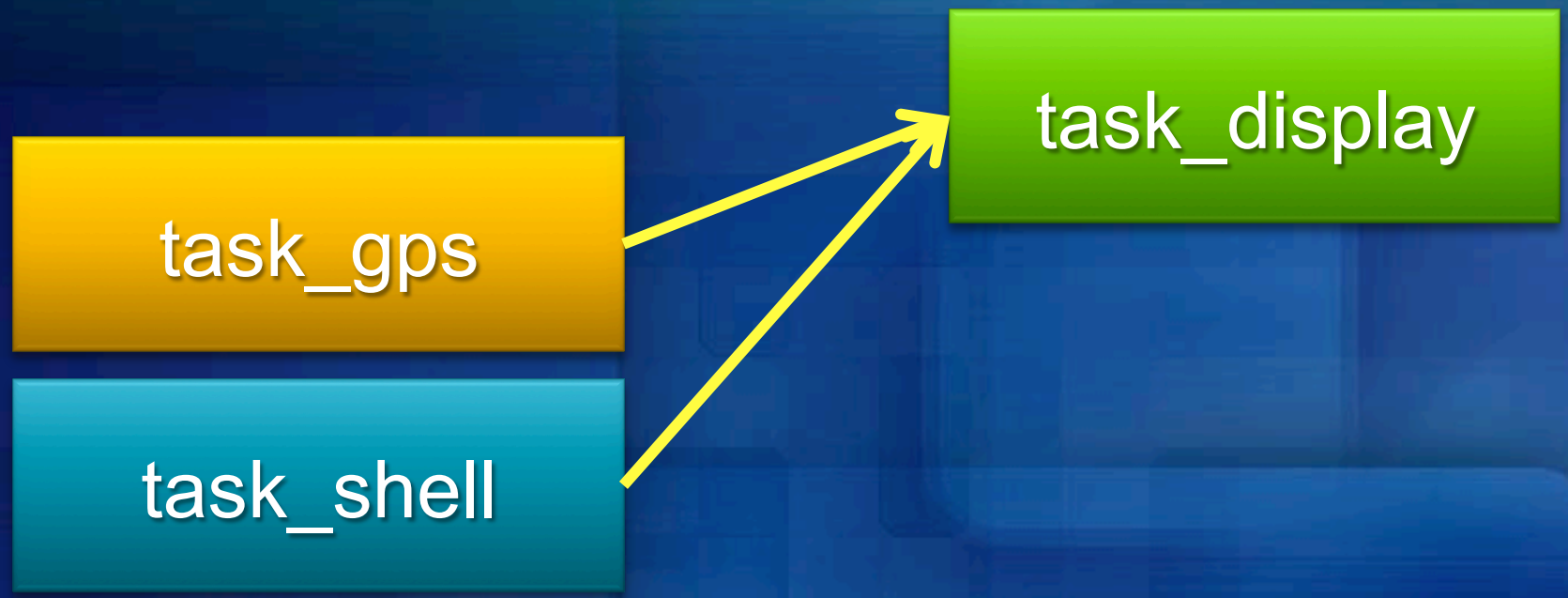
シェルタスクによる
システムインターフェースも装備

COM37:115200baud - Tera Term VT

ファイル(F)　編集(E)　設定(S)　コントロール(O)

```
GPSCLOCK>
GPSCLOCK>help
help    :Help.
info    :Information.
display :Command for display task.
GPSCLOCK>info
info sys
info ver
GPSCLOCK>info sys
LPCXpressoClock
GPSCLOCK>info ver
0.0.1
GPSCLOCK>display
display text [TTTTTT]
GPSCLOCK>display text 123456
GPSCLOCK>
```

APIの再利用例を提示



task_gpsもtask_shellも、「表示を更新する」作業に対して、task_displayの同一APIを使用している。
メッセージ・パッシングは、task_display自身がメカニズムを隠蔽しており、task_displayにリクエストを出す各タスクが、メッセージ・パッシングの実装詳細を知る必要もない。

RSSのメリット

- RTOS使用方法サンプルとして使用可能
  - RTOS固有の使用方法を内包
- RSSを雛型にプロジェクトを始められる
  - 複数ファイルに分割統治された設計
- RTOSやソフトウェア初学者に提示可能な内容
  - ちょっとした工夫が大きな差を生む実例を提示

リアルタイム・システムの
雛型として活用可能な
生きたサンプルを提供

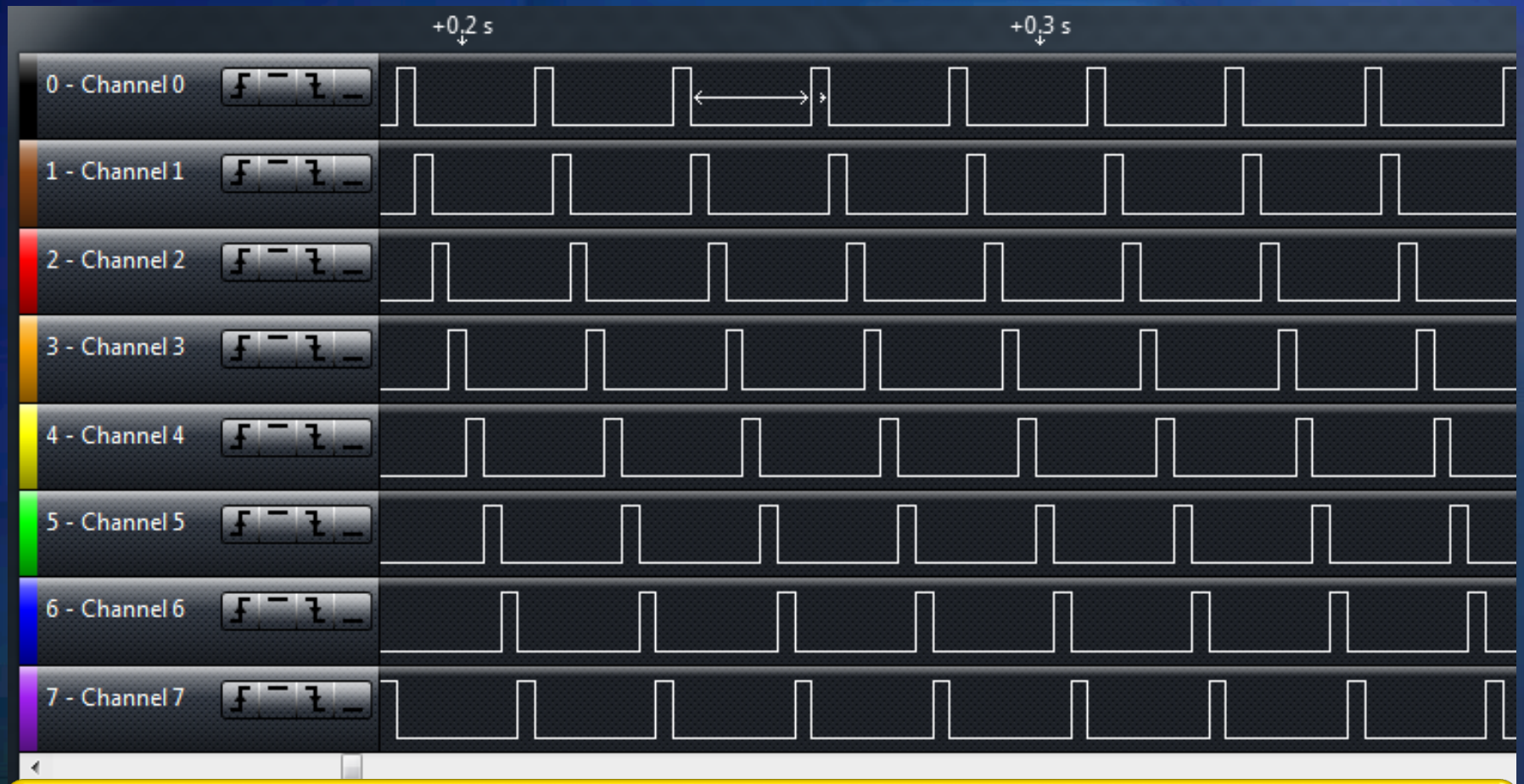
RSSの先に…

組み込みシステムは検証までが設計です

- 「本当にリアルタイム性を有していますか？」
- 「それはどうやって確認しましたか？」
- 「検証は再現可能な方法ですか？」

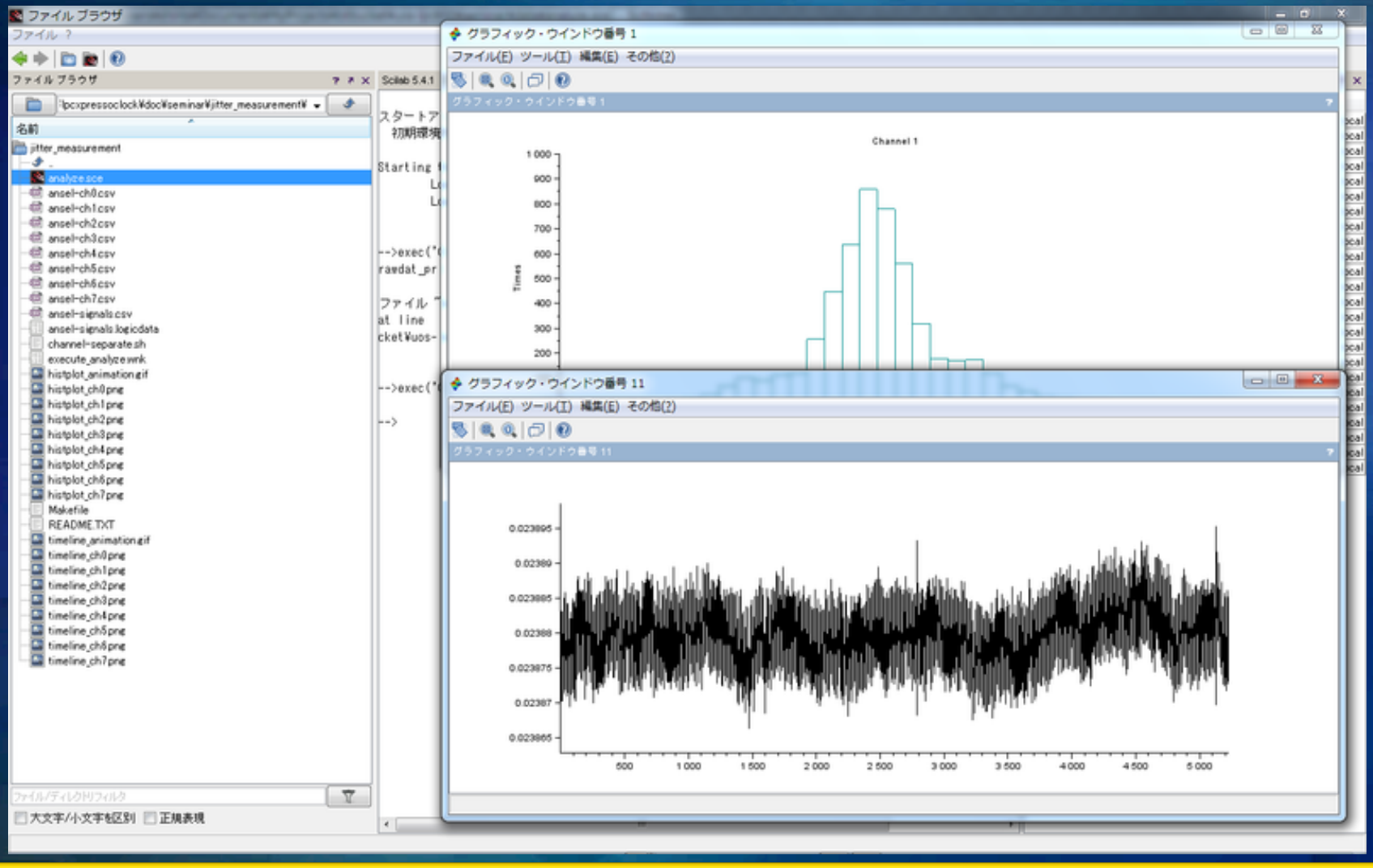
TOPPERSの教材で
ここまで網羅する物を
私は見た事が無い。

ロジック・アナライザによるデータ収集



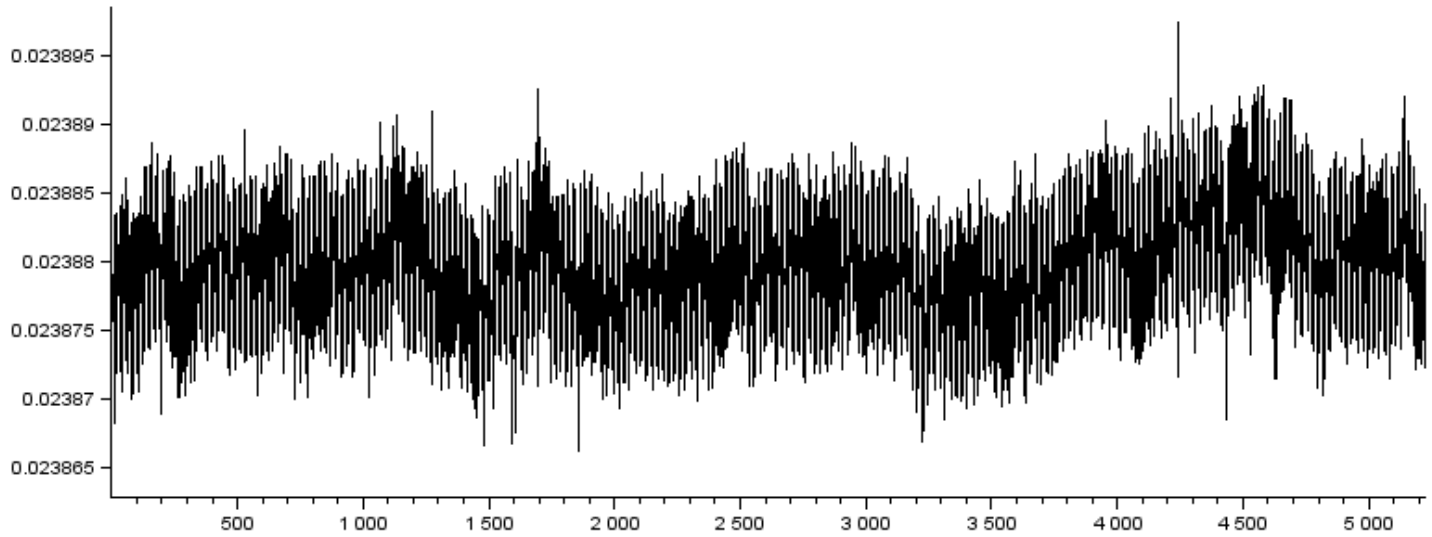
リアルタイム性をロジック・アナライザで判定するのは不可

収集データをスクリプトで解析



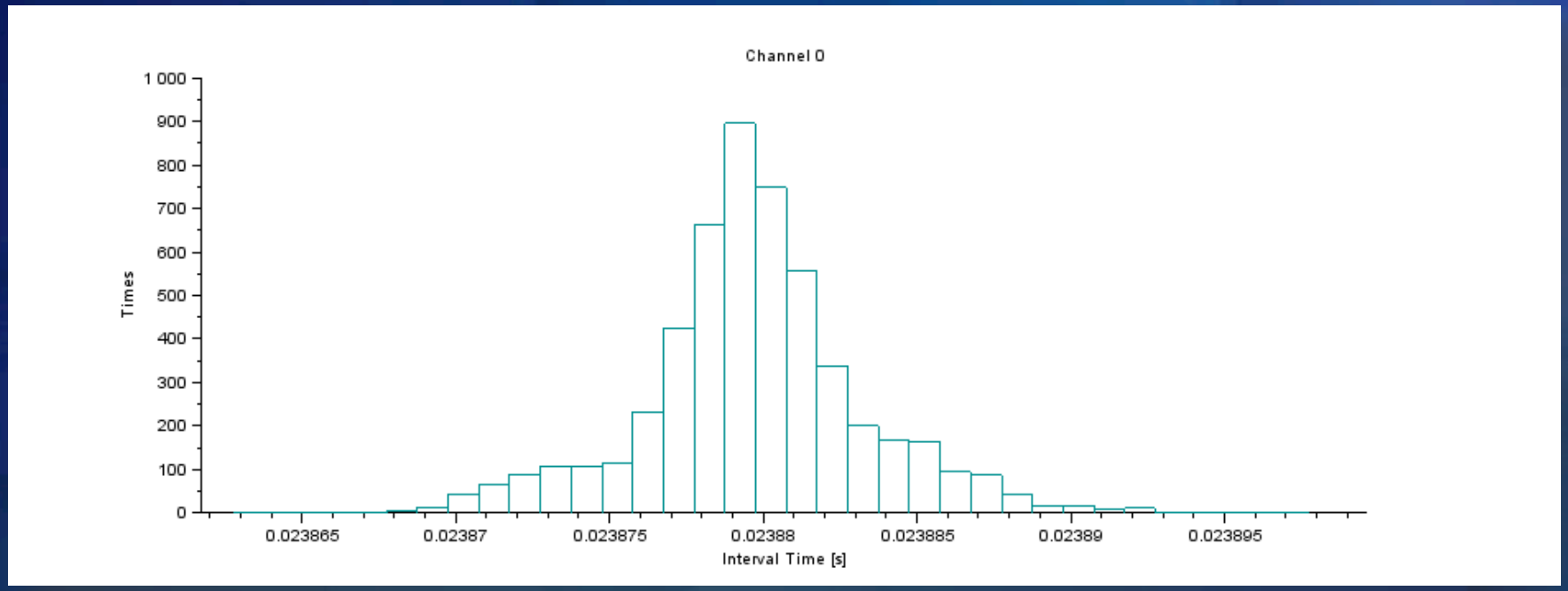
スクリプトによる自動化で再現性の高い検証を提供

収集データから分布を調査
チャネル毎の処理時間推移



時系列的変化と全体傾向を把握

収集データから分布を調査
チャネル毎の処理時間推移



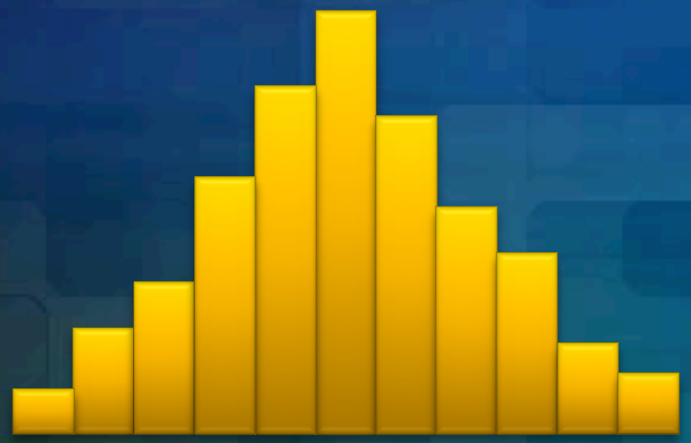
検証対象時間内におきるバラツキの傾向を把握

ヒストグラムの判定例

一定間隔に起きるイベントに対する解析例

一般型

- 左右対称の形
- 中央付近が最も高い
- 安定した特性



ふた山型

- 非左右対象の形
- ふたつの山が現れる
- 異質特性が混在した特性

現場で感じる事

- 波形を見ないで議論しようとする人が結構いる
  - 議論の対象は何？
- 実動作確認しないまま良しとする人が結構いる
  - そのシステム、本当に期待通りの動作してる？
  - たまたま動作しているだけという可能性はない？

設計と実際の対比が、システムが
期待通りに動作しているのか確認する唯一の方法。

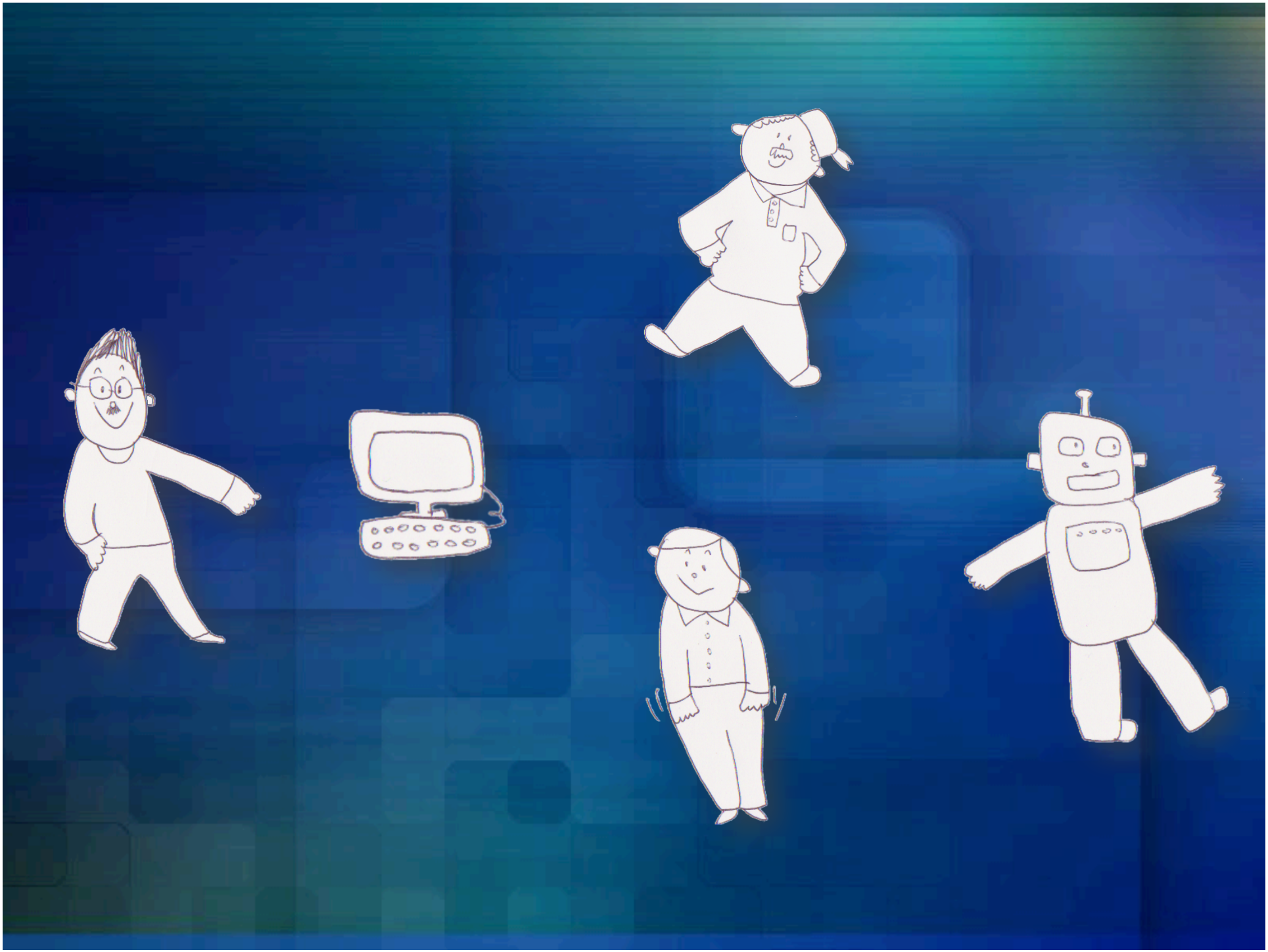
リアルタイム・システムは、
ソフトウェア由来のバグが特に潜みやすい。

組み込みシステムは検証までが設計です

- 「本当にリアルタイム性を有していますか？」
  - 統計的処理による検証が必須
- 「それはどうやって確認しましたか？」
  - 簡単な機材の組み合わせでも十分可能
- 「検証は再現可能な方法ですか？」
  - スクリプトによる自動化により再現可能

リアルタイム・システムの真の楽しさがここにある

RSSで

- TOPPERSを使ったリアルタイム・システム設計の基礎から、実業務にも応用可能な実装から検証までの一連の流れを学ぶ機会を提供。
- 教育のための教育でない、作って学べる、使って楽しめる仕組みを提供。

TOPPERS RSSで
楽しい組み込みライフ！