EPSON

携帯情報機器へのITRON適用事例

1999年6月30日

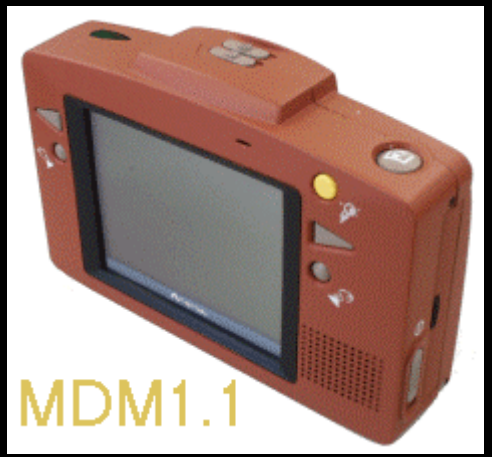
舘 義宏 Tachi.Yoshihiro@exc.epson.co.jp

セイコーエプソン株式会社

EPSON

マルチデータメモ(MDM)とは

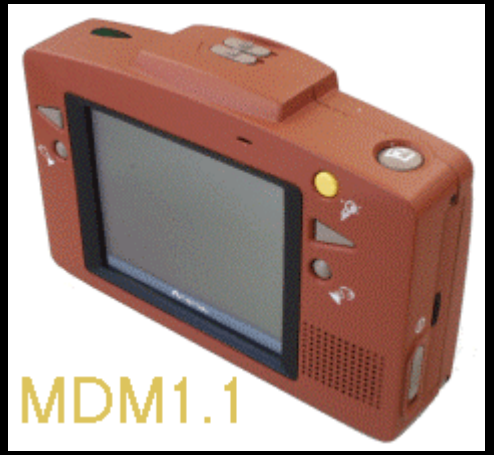
- デジタルデータを扱うメモ帳
  - 画像・音声データ・手書きメモを自由に記録
  - データ/機能をページに貼り付け
  - すべての基本操作がワンアクション。
- RISCマイコン+ITRONで多機能を実現
  - デバイス制御
  - データ圧縮伸長(JPEG/ADPCM)
  - PDA風アプリケーション



EPSON

MDMのハードウェア

- CPU: SH2 20MHz
- ROM: 1Mバイト(プログラム+フォント)
- RAM: 1Mバイト
- 記録媒体: 内蔵2Mバイト(+PCカード)
- 液晶表示: 320×240画素 モノクロ16階調
- カメラ: 27万画素カラー
- インターフェース: RS232C, PCカードスロット
- 電源: 単3乾電池2本(20時間)、またはACアダプタ
- その他: タッチパネル、マイク、スピーカ、スイッチなど



EPSON

MDMのソフトウェア

アプリケーション

リアルタイム性を要求する処理
- カメラ処理
- ボイスレコーダー処理
- 手書きメモ処理

システムサービス
- ファイルシステム
- GUI ・通信
- そのほかの資源管理

デバイス制御
- 画面
- CCD
- マイク・スピーカ
- タッチパネル ・スイッチなど

電源管理

μITRON

ハードウェア

EPSON

携帯情報機器に必要な仕組み

- タスクの安全な終了
- 外部プログラムの実行
- 電源管理(低消費電力の実現)
- そのほか
  - アプリケーション開発環境
  - GUIの搭載
  - 通信プロトコルの搭載
  - 複数ファイルシステムのサポートなど

} 今回の話題

EPSON

タスクの安全な終了(1/3)

- この仕組みの目的
  - ユーザーによる処理の中止
  - 時間的制約による処理の打ち切り
- 方法
  - 1. 資源獲得の記録を取る。
    - 資源獲得・解放手順を明確にし、記録に基づいて解放する。
    - 標準的資源をあらかじめ獲得し、自動的に解放する。

EPSON

タスクの安全な終了(2/3)

- 方法
  - 2. 終了ハンドラを定義する。
    - タスクが終了ハンドラを定義し、終了時には終了ハンドラが呼び出されるようにする。
  - 3. タスクテンプレートにのせる。
    - タスクのテンプレートを作成し、テンプレートの制約下でタスクを安全に動作させる。
    - テンプレートごとに、動作の記録や終了ハンドラの定義を実施する。

EPSON

タスクの安全な終了(3/3)

- 結果
  - タスク記述方法を制約することで、安全な動作が可能になる。
    - 具体的な制約は機器の性格に従う。
    - 制約自体はテンプレートを用いて再利用する。

EPSON

テンプレート例: イベント駆動型

- GUIと協調して動作するタスクのテンプレート例。
  - メッセージバッファを見ながら動作する。
  - 終了要求方法==終了型イベントの送付。

```
{
    handler=find_handler(this_task, 起動型イベント);
    do{
        handler(); // イベントハンドラの実行
        trcv_mbf(mbfid, msg, tmout);
        handler=find_handler(this_task, event_type(msg));
        if(event_type(msg)==終了型イベント) break;
    } while(1);
    handler(); // 終了イベントハンドラの実行
}
```

EPSON

外部プログラムの実行(1/4)

- 仕組みの目的
  - ハードウェアチェックプログラムの実行
  - アプリケーションの追加
- 仕組みの内容
  - 1. タスク優先度の割り当て
    - あらかじめ、タスク優先度の層別や予約をしておく。
      - 例: 優先度16以下はリアルタイム性能を特に要求しないタスク。奇数の優先度は外部プログラム用。
    - 外部プログラム内に優先度を記述する。

EPSON

外部プログラムの実行(2/4)

- 仕組みの内容
  - 2. プログラムのロード
    - テンプレートを利用して、内部のタスク同様cre_tskでテンプレートタスクを生成する。
    - テンプレートは自力でプログラムロード、初期化処理、エントリへのジャンプ、終了処理などを行う。

EPSON

外部プログラムの実行(3/4)

- 仕組みの内容
  - 3. 内部の機能の呼び出し
    - あらかじめ、内部の機能、すなわち組み込み済みのAPIやデバイスドライバなどを管理する仕組みを用意しておく。
      - 例: 内部の機能は機能管理モジュールにみずからの機能とハンドル獲得用のインタフェースを通知しておく。
    - 外部プログラムは拡張SVCを介して、内部の機能へのハンドルを獲得する。
    - ハンドルを利用して、内部の機能を直接呼び出す。

EPSON

外部プログラムの実行(4/4)

- 結果
  - ITRON環境でも簡単な仕組みで外部プログラムを実行できる。
  - 次の仕組みは必要。
    - プログラムをロードするための、ファイルシステムや通信プロトコルのミドルウェア。
    - 内部の機能を呼び出すために、その機能のポインタを知る手段。

EPSON

電源管理(1/3)

- 目的
  - 消費電力を低減させ、電池で長時間動作させる。
  - 電源電圧を監視し、電圧低下による誤動作を防止する。
- 方法
  - 1. CPUを低消費電力モードに設定する。
    - タスク優先度最低のタスク(idleタスク)が常にCPUを停止させる。
    - 割り込み発生でCPUが動作を再開する。

EPSON

電源管理(2/3)

- 方法
  - 2. まめにデバイスの電源を切る。
    - open/closeの呼び出しでデバイスの利用状況を把握し、利用者がいないデバイスの電源は切る。
  - 3. オートパワーオフ
    - オートパワーオフカウンタが設定値を越えたとき、イベントキューに電源OFFイベントを押し込む。
    - ユーザーと対話的に動作するアプリケーションが、動作のたびにオートパワーオフカウンタをリセットする。

EPSON

電源管理(3/3)

- 方法
  - 4. 定期的に電圧を監視し、不十分なときには電源ONさせない。
    - 必要な電圧を得られないデバイスは電源ONさせない。
    - 電源ON時に電圧降下したら、ONをキャンセルする。
    - 動作中に必要な電圧を切ったら、イベントキューに電圧降下イベントを押し込む。
- 結果
  - きめこまかな電源管理で安定・長時間動作。

EPSON

アプリケーション開発環境

- 目的
  - より多くのアプリケーションを開発する。
  - より詳細にアプリケーションをデバッグする。
  - 実機やICEを用意する時間的・金銭的余裕がない。
- 方法
  - インタプリタなどの言語の活用。
  - シミュレーション環境の構築。

EPSON

シミュレーション環境の構築(1/3)

- ItIs-winで簡単なシミュレーション環境を構築
- 選択の理由
  - ItIsとPC・開発環境を含めてすぐに手に入り、極めて安価。
  - MDMにはリアルタイム性能を要求しないタスクが多い。
  - 基本アルゴリズムはもともとPCで開発している。
  - 言語とその動作環境を作成するより容易。

EPSON

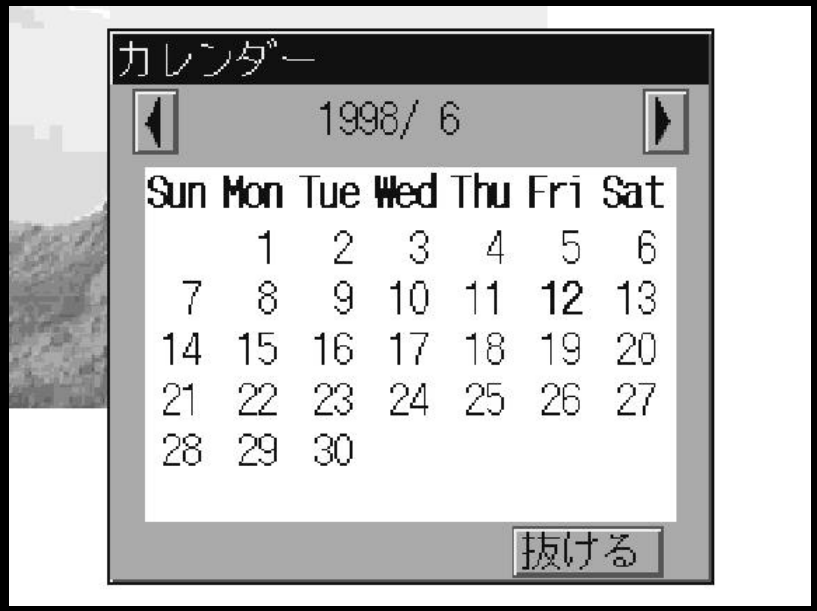
シミュレーション環境の構築(2/3)

- 構築手順:
  - ハードウェア初期化をItIs-Winの起動に置換。
  - デバイスの機能を呼び出す関数を、Win32上のシミュレーション環境を操作する関数(またはダミー関数)に置換。
  - エンディアン・ITRON仕様の違いを吸収するように修正。
  - ミドルウェア・アプリケーションをWin32上で再コンパイル。

EPSON

シミュレーション環境の構築(3/3)

- 結果
  - Windowsマシンで簡単に開発・デモできる。
  - ITRONタスク==Win32スレッドなので、デバッグも容易。
  - 今後、リトルエンディアンマシン(SH-CARDなど)とμITRON4.0を使えばさらに簡単に。



EPSON

まとめ

- タスクの安全な終了
  - 自由度を下げて安全な動作 終了。
- 外部プログラムの実行
  - 簡単な仕組みで実行。
- 電源管理
  - 機器の特性に応じたきめこまかな電源管理。
- アプリケーション開発環境
  - ItIs-winを用いて簡単にシミュレーション。