Digital CameraへのLinuxの搭載

Porting Linux onto Digital Camera

大内　茂樹*　　アラン ボルマ*
Shigeki OUCHI　　Alain VOLMAT

要　　旨

アプリケーションを実装しやすい環境を作り，ビジネスユースとしてのデジタルカメラの可能性を検証する事を目的として，RDC-i700にLinuxの搭載を行なった．近年，組込機器へのLinuxの搭載が増えているが，大半はネットワーク機器やPDAなどPCに近いアーキテクチャが対象であり，デジタルカメラのようにタイミングクリティカルな機器への搭載は例がない．本研究ではデバイスドライバをレイヤ化し，アルゴリズム処理をカーネル内で行なう事などにより，リアルタイム拡張のないLinuxカーネルでデジタルカメラを動作させる事に成功した．また，デモ用アプリケーションをプロトタイプする事で，アプリケーションプログラムの実装容易性も実証した．

さらに，こういったプラットフォームは臨界点を越えないと普及しないという特性を持っているため，プロジェクトメンバ自身が普及活動も行なっている．本論文内でそれらもあわせて紹介する．

ABSTRACT

We've ported Linux onto RDC-i700 in order to study the feasibility of digital camera as business use. Recently Linux is running on a more and more wide range of embedded devices, but almost all of them target PC-like devices, for example, router, PDA, HDD video recorders. This means timing critical equipment like digital camera remains to be unexplored. In this study, we successfully ported Linux kernel to digital camera by bring in new driver layer which processes several algorithms, prototyped several application software, and verify the ease of implementing application software.

Also, this kind of platform never becomes poplar unless it reaches a critical point. For that reason , since we've been promoting our project by ourselves, we also introduce this activity in this paper.

* ソフトウェア研究開発本部　ネットワークシステム研究所
Network and System Lab., Software R&D Group

1. 背景と目的

Fig.1に示されるRDC-i700は2000年にリコーから発売されたデジタルカメラである．著名なRTOS（RealTime Operating System）の一つであるVxWorksをOSとして搭載し，すでに十分な動作実績がある．PCMCIAなど拡張性にすぐれ，ビジネスユース用として秀れたカメラである．

それでは，なぜわざわざ別のOSを搭載する必要があるのであろうか．それは，我々がこの既存のデジタルカメラを「プログラマブルカメラ」にしたいと考えているからである．プログラマブルカメラになれば，多くのVAR（Value Add Reseller）やSIer（System Integrator），あるいは個人のプログラマがこのデバイス用の有用なプログラムを書いてくれる事を期待できる．そして，それが加速していけば，デジタルカメラのビジネスユースプラットフォームにする事が可能となるのである．

すなわち，Linuxを搭載する事により，アプリケーションプラットフォームとしてのデジタルカメラを実現する事が本プロジェクトの目的である．

Fig.1　RDC-i700.

2. 技術

2-1　RDC-i700

RDC-i700はCCDをはじめとしたカメラ関連のデバイス以外に，PCMCIA及びCF（Compact Flash）スロットを持っている．これによりユーザは撮影した画像をE-mailで送ったり，インターネットを閲覧したりする事が可能となっており，世界初の「Internet-readyデジタルカメラ」と称している．Fig.2に示すようなデバイスを装備しており，これらをPC にも同様の機能があるデバイスと，カメラ独自のデバイスとに分けて，以下に機能と制御する際に問題となる点を説明する．

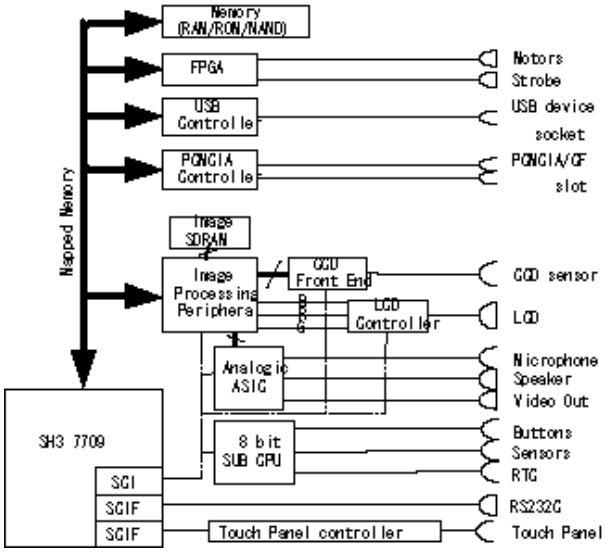

Fig.2　Devices of RDC-i700.

2-1-1　PC-likeデバイス

CPUやPCMCIA，CF，入力デバイスとしてのタッチパネル，出力デバイスであるLCD などがこれにあたる．

- CPU…日立（現在はルネサス）のSH7709Aを使用している．これは SH3 アーキテクチャの32bit RISC CPUであり，MMU（Memory Management Unit）及びシリアル通信，I/Oポート，AD/DAコンバータなどのいくつかの周辺機器コントローラを内蔵している．SH3アーキテクャのCPUはビッグエンディアンでもリトルエンディアンでも動作可能であり，エンディアン選択ピンによって決定されるが，RDC-i700はビッグエンディアンで動作している．これはハードウェア的な選択でありソフトウェアによる変更はできない．
- PCMCIAコントローラ…丸文社製のMRSHPC-02が採用されており，このコントローラによって，PCMCIAスロット及びCFスロットが制御されている．
- 入力デバイス…タッチパネルといくつかのボタン類が用意されている．ボタンは数が少ないため，PCにおけるキーボードとは根本的に違う制御が必要であり，タッチパネルはPCで使用されるものとほぼ同様で，SH3 CPUにシリアルインターフェースを通じて接続されている．
- 出力デバイス…視覚情報の出力デバイスとしてLCDを備えており，LCDのコントローラはIPP（Image

Processing Peripheral）を通じてSH3と接続されている．IPPは機能の一部として，PCにおけるグラフィックスチップと同様の役割を果たすが，インターフェースは全く異なるものとなっている．

PCMCIAを介してEthernetカード等を接続する事によって実現されるネットワーク接続や，タッチパネル付きのLCDなどにより，様々なアプリケーションを動作させる事が可能となっている．これらには，静止画の撮影や，サウンドや動画の記録など，カメラ特有のアプリケーションも含まれるが，通常PCで動かすようなアプリケーションも含まれる．例えば，撮影したばかりの写真をワイヤレスLANで共有したり，添付ファイルとしてE-mailやFAXで送ったりなどである．このようなアプリケーションにはこれらのPC-likeデバイス欠かせない．

2-1-2　カメラ特有のデバイス

カメラ特有のデバイスの中で，最も重要なデバイスは，IPPである．このチップがこの機器をデジタルカメラたらしめており，その周辺にCCDコントローラ，CCD F/E（Front End）チップ，画像用SD-RAMが配置されている．

- CCD及びコントローラ…CCDはイメージをキャプチャする単なるセンサであり，取得されたイメージはIPPを通じて，SD-RAMに送られる．
- メカニカルパーツ…メカニカルパーツには，ズームやフォーカスが含まれる．これらのパーツは高精度の動作が必要なため，ステッピングモータが使われている．モータを高速に動作させるためには，CPUは非常に高速にIOポートに値のセットを行なわなければならない．その場合のタイマーの間隔は，数msec程度でしかない．
- ストロボ…ストロボの制御に際に，問題となるのは，高速に動作させる事ではなく，非常に高精度な同期である．ストロボがまさにフラッシュする瞬間と，CCDが画像をキャプチャする瞬間を同期させなければならない．このケースで必要な精度も，数msecである．
- 自動露出とホワイトバランス…これらは実際には実在するデバイスではなく，アルゴリズムなのであるが，カメラたらしめるために必須の機能を提供するものとして，ここで説明する．

  自動露出は，撮影場所の明るさによらず，適切な光量を得るために，CCDのシャッターが開いている時間や絞りの開度を調整するためのアルゴリズムである．ホワイトバランスは，撮影場所の光源によらず，撮影された映像を適切な色合いに調整するためのアルゴリズムであり，IPPにパラメータをセットする事で調整を行なう．

  これらの2つのアルゴリズムには，いくつかの実装が可能であり，それらの中には重い計算が必要だが高性能なものや，逆に性能はそこそこだが，軽くて速いものなどがある．問題となるのは，これらのアルゴリズム部が呼び出される回数と間隔であり，非常に短かい間隔で何度も呼ばれうる．最も多いケースでは，CCDの一フレーム毎に呼ばれる可能性もある．

この節では，デジタルカメラの特殊性について説明した．次節では，これらの機能をどのようにLinux上に実装したかを議論していく．

2-2　利用可能な既存リソース

Linuxはオープンソースのプロジェクトであり，我々はこの新しいハードウェアに対してでも，数多くの既存のリソースを使用する事ができる．RDC-i700に用いられているSH3 CPUに対応したLinuxカーネル（SH-Linux）や，クロスコンパイル用のツール類はWeb上から入手可能である．(*1，*2) むろんCPU自体に対応していても，デジタルカメラという特異なアーキテクチャでそのまま動作するわけではないが，新規にCPUに対応させるよりはずっと短期間で移植が可能である．

SH3 CPUは内蔵のシリアルインターフェースを持っており，SH-Linuxはこれに対応している．CPU内部にある事から，アーキテクチャに依存せずに，そのままRDC-i700のケースでも使用可能である．この内蔵シリアルポートがあるおかげで，カーネル開発の初期のうちから，シリアルコンソールが使用でき，これは効率的なデバッグに貢献している．

CPU内蔵デバイス以外にも，RDC-i700で使用されているPCMCIAコントローラは日立の評価ボードであるSolution Engineと同シリーズのものを使用しており，これもSH-Linuxでサポートされている事から，少しの変更で移植する事がで

きた.

RDC-i700はPCではなく組込機器である事から，当然PCのようなBIOSが最初からあるわけではない．IPL（Initial Program Loader）も自身で組み込む必要がある．SH-Linuxプロジェクトは SH-IPL+Gと呼ばれるIPLや，SH-Liloと呼ばれるブートローダも公開している．我々は，開発の初期段階はCFをPCにおけるハードディスクのように，ブートデバイスとして使用していた．これはSH-IPL+G及びSH-Liloが想定する環境と同じであり(*3)，これら2つのソフトウェアを流用する事ができた．ソースコードの変更点は，メモリマップの変更など，ごくわずかであった．現在はカーネルはROM化されているため，IPLのみを用いている.

さらに，当然の事ながら，数多くのユーザランドで動作するプログラムが使用可能であり，これらには組込機器用の小さなウィンドウシステムなども含まれる.

2-3　デザイン上の選択

上記のように，数多くのソフトウェアリソースが使用可能ではあるが，LinuxをOSとして使用可能なデジタルカメラは他にない事から，我々は今までにないデザイン上の選択を追られるケースが多い．ウィンドウシステムや，ファイルシステムや，リソースの削減などである.

Table.1 Comparison resources of window systems

	Qtopia	Microwindows	Shikigami
ROM	6-8MB	1-2MB	N/A
RAM	4-8MB	1-2MB	4-8MB

2-3-1　ウィンドウシステム

組込機器用にデザインされたウィンドウシステム，あるいはGUI環境はいくつか存在している．Qtopia，Microwindows，式神などがよい例である．式神は日本の会社であるAXEが作ったものであり，日本語の扱いにすぐれている．たとえば日本語の手書きOCR入力などである．これは大変魅力的な機能であったが，商品でありソースが公開されていないという問題があった.

一方，Qtopia及びMicrowindowsはソースコードが公開されており，テスト的な使用や変更が可能な事などが利点となる．QtopiaはQt/Embeddedツールキットをベースにしたもので，TrollTech社が開発した．非常にパワフルな環境であり，実際シャープのLinuxを搭載したZaurusなどでも採用されている．しかしながら，メモリをはじめとして必要リソース量が多く，ウィンドウシステムのみならともかく，JavaやWebブラウザなど，他の魅力あるアプリケーションと共に16MBのRAMのみを持つRDC-i700上で動作させるのは困難である.

その点，Microwindowsは非常に小さいウィンドウシステムで，比較的リソースの貧弱なものを含む数種類のアーキテクチャでの動作実績もあり，この点で非常に有利に思えた．RDC-i700はSH7をビッグエンディアンで使用しているが，これはLinuxの世界では非常に少数派であり，移植性は選択における最重要項目の一つだからである.

Microwindowsのソースコードを精査した結果，移植の際に変更しなければならない部分は非常に小さい事が分かった．また，プログラミング用のAPIもX11のそれと非常に似ており，この点もアプリケーションの実装の容易さという点で魅力的な事から，MicrowindowsをRDC-i700用のGUI環境として選択した．フレームバッファがCPUからリニアに見えないという点で，苦労した点もあるが，非常に小さいコード改変量で動作させる事に成功した.

2-3-2　ファイルシステム

ファイルシステムを決定するに際しては，二つの使い方を考慮しなければならない．一つはカーネルの上，すなわちユーザ空間で動作するアプリケーションプログラムを格納するLinuxのディレクトリツリー用，もう一つは撮影した写真のストレージとして用法である．ユーザ空間用に関しては，最終的にはROM化する予定のため，現在はPC用のLinuxで標準的に使用されているEXT2と呼ばれるファイルシステムを使用している.

ストレージ用としては，CFカードと内蔵のNANDフラッシュメモリを考慮する必要がある．CFカードは他のデジタルカメラとの互換性を考慮して，FATフォーマット以外選択の余地がないため，問題は内蔵メモリのフォーマットという事になる．Linuxカーネル2.4.xでは，フラッシュメモリ用のドライバとして，JFFS及びJFFS2の二つが存在している．JFFS2はNANDメモリをサポートしているが，常にファイルを圧縮して格納する．一方，JFFSは圧縮は行なわないが，NANDをサポートしていない．内蔵メモリは写真のストレージとして使用するものであり，写真はJPEGファイルとして

圧縮されている事が多いため，格納時に圧縮する事は，時間のロスにつながるだけで使用率の削減には結び付かない．

我々は，JFF2の改良を考えだが，幸運にも YAFFS (*4)と呼ばれるLinuxカーネル標準添付ではないドライバがある事を知った．標準でないものは安定性などでリスクをともなうが，テストした所，性能，安定性共に満足の行くものであった．これによって，我々がJFFS2を改良する時間を節約できただけでなく，このカメラのユーザが40%セーブの時間を節約する事にもつながった．

2-4　新規実装が必要なもの

これまで，そのまま，あるいは小規模な変更で使用可能な既存リソース，及びリソースとしては存在しても，デザインの観点から選択をしなければならないものを説明した．

一方，我々のデジタルカメラは世界初のLinuxで動作するものであり，カメラ特有のデバイスを制御するためのLinux用ドライバは存在しない．実際，製品版のVxWorksを搭載したRDC-i700は何の問題もなくデバイスを制御しているにもかかわらず，これらのドライバをLinuxに移植するには相当な工数が必要となる．なぜなら，VxWorksをはじめとする組込用のOSとUNIX-likeなOSではドライバの構成，書き方等が完全に異なるからである．

我々のデザインによるデバイスドライバの構成をFig.3に示す．

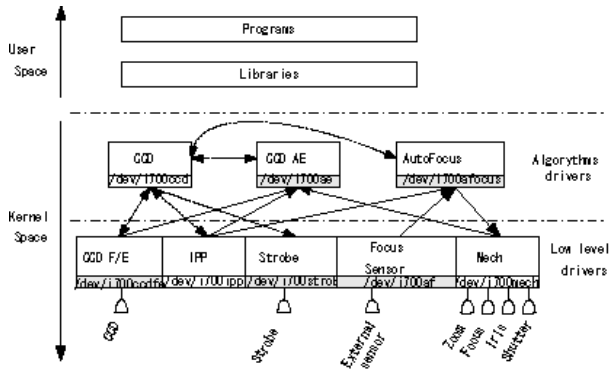

Fig.3　Structure of device drivers.

2-4-1　CCD

CCDは上述のように単なるセンサであり，実際にはIPP及びF/Eチップを介して制御される．IPPのドライバはデバイスの初期化を除いては，カーネル内ライブラリのように実装されている．これらの機能は他のデバイスから呼び出し可能であり，以下のような機能を提供している．

・JPEG圧縮/伸長
・YUV-RGB変換
・video出力
・イメージスケーリング

CCDのドライバはより上位のレイヤに位置し，すべての操作の起点となるものである．他のデバイスとの協調動作を担当し，ユーザランドのライブラリから呼び出してモニタリングモードや静止画撮影モードなどを制御する事ができる．本ドライバは，F/E，IPP，MECHなどの下位のハードウェアドライバを呼び出し，Linuxカーネルにより提供されるwait queueを用いて同期を取っている．

2-4-2　ストロボ

ストロボは，通常のUNIXで用いられるOPEN/READ/WRITE/CLOSEのアーキテクチャで制御する事は難しい．この方法でストロボを発光させる事は可能であるが，CCDから画像をキャプチャするタイミングとの同期をユーザランドで行なう事になり，正確な同期ができないからである．そこで，ストロボの発光はCCDの属性として実装し，カーネル内のCCDドライバから呼び出されるようにしている．

2-4-3　メカニカルパーツ

メカニカルパーツで問題となるのは，ステッピングモータの制御である．RDC-i700で使用されるモータは1ステップ毎に1.8msecのウェイトが必要であるが，Linuxカーネルの分解能は10msecであり，デフォルト値のまま動作させると1ステップ毎の動作が非常に遅くなる．Linuxのカーネルは分解能をHZというマクロ定数で定義しており，これを変更し分解能を1msecとして動作させる事で解決した．なおいくつかのドライバやユーザランドプログラムには分解能が10msecである事が前提の，すなわちマクロを用いない行儀の悪いものが存在しており，ソースコードの修正が必要であった．

以上，カメラ特有のデバイスを中心に，新規に実装する必要のあるものについて説明を行なった．

3. 成果

これまでの研究開発により，RDC-i700上のLinuxは安定して動作するようになり，基本機能である静止画撮影のみならず，ネットワークを用いたアプリケーションも動作するようになっている．

現在は，「アプリケーションを実装しやすいデジタルカメラ」というコンセプトを実証すべく，インターンシップ学生によるアプリケーションの実装や，大学等研究機関への貸し出しによるキラーアプリケーションの発掘にステージを移している．

4. 今後の展開

実装の段階は，ほぼ終了したもののいくつかの課題が残っている．

4-1 技術的課題

4-1-1 パワーマネージメント

現在残っている最大の技術課題はパワーマネージメントである．現在はカーネル内において，全く管理をしておらず，そのため無線LANカードを用いた場合，バッテリの持続時間は，わずか20-30分ほどである．まず，各デバイスを制御するデバイスドライバに，低消費電力モードに遷移したり復帰するためのコードを追加する必要がある．これは，それほど難しい事ではない．最大のポイントは，各デバイスが，いつ低消費電力モードに入るか（そしていつ復帰するか）を協調動作させるための仕組みである．これに関しては最近，組込Linuxが盛んになっている事から，いくつかの提案があり，それらをよく調査して進めていく予定である．

4-1-2 パッケージング

現在開発用のCFには非常にたくさんのプログラムを格納しているためサイズが巨大化している．ROM化する際には「ダイエット」しなければならない．アプリケーションカメラは，最終的には，ある種の特注対応カメラとして動作する事が想定されるため，それらの需要毎に手作業でこの作業を行なう事は，非常に骨の折れる作業となる．そのため，特定のアプリケーションに必要なツールやドライバだけを抜き出してROM化用ファイルシステムイメージを容易に作成する機構が必要である．

4-2 非技術的課題

我々のゴールは単にプログラマブルカメラを作る事ではなく，ここから利益を得る事である．ホビーユーザは「おもちゃ」として現在のLinuxカメラで遊んでくれるかもしれないが，ビジネスとするためにはVARやSIerの参入がかかせない．そのためにいくつかの課題が考えられる．

4-2-1 配布

我々の活動を加速させるために，ソースコードを配布する準備を進めている．法律的な面でまだ解決しなければならない点が残っているが，技術的な面でのチェックは終了している．

4-2-2 コミュニティ形成

オープンソースを用いた開発活動では，ある種のサポートコミュニティの存在が欠かせない．初期にはおもちゃとしてとびついてくれるホビーユーザが必要であり，コミュニティの大きさがある臨界点をこえると，ビジネスユーザ/デベロッパが参入してくるという仮説のもとに，コミュニティを形成していく計画である．ソースの配布もこれに必要なものであり，またLinuxを冠した学会やユーザズ会での発表なども行なっていく．

4-2-3 キラーアプリケーションの探索

VARやSIerの参入のためには，ビジネスに使える事の証明が必要であり，そのために必要なもう一つものが規模ではなく，魅力的なキラーアプリケーションの存在であると考える．現在はインターンシップの学生に細々とやってもらっている段階であるが，もう少し大きな規模のものを含めて進めていく．

5. 謝辞

Linux及び多くのフリーソフトウェアを開発しているコミュニティのみなさまに感謝いたします．彼らの存在がなけ

れば本プロジェクトは意味をなさない事は言うまでもありません.

参考文献

1) SH-Linux kernel; http://sourceforge.net/projects/linuxsh

2) SH-Linux Cross development tools; http://www.sh-linux.org/

3) SH-Linux Boot Sequence ;
http://linuxsh.sourceforge.net/docs.php3

4) YAFFS; http://www.linuxdevices.com/articles/AT9680239525.html