B-004

Linux 用 USB デバイスドライバ生成支援に関する一提案
Support for Code generation of Linux USB device drivers

吉田 泰彦 †　　大原 茂之 ‡
Yasuhiko Yoshida　　Shigeyuki Ohara

1．はじめに

急速なネットワーク技術の進展によるユビキタス環境の普及に伴い，デジタル機器産業の競争力の源泉となる組込みソフトウェアの需要が急増している．現在では組込みシステム関連企業の従業員 480 万人のうち，組込みソフトウェア技術者数は 17 万 5,000 人にのぼり，組込みソフトウェア開発費は年間 2 兆 4,000 億円と推定されている．業界自体の伸びと比例し，組込みソフトウェア技術者に対する需要が増加の一途を辿る中，技術者不足が叫ばれるようになった．

組込みソフトウェアには，高い信頼性が要求される．しかし，組込みソフトウェア技術者の不足により，信頼性の確保が難しい状況になってきている．そこで，バグを出さないようにすることが必要である．そのためにはソースコードを書かずに生成していくような支援が必要である．

支援の題材として，今注目されている Linux と USB(Universal Serial Bus)を取り上げる．Linux はオープンソースであることから，バグの発見やセキュリティにもメリットがある．また，組込み機器向けに Emblix という RTOS(Real Time Operationg System)がある．USB には「高速/高信頼通信」「プラグ&プレイ」「小型コネクタ，電力供給による省スペース，省電力化」「ホスト/デバイス構成による機能実装の容易さ」というメリットがある．

よって本研究では，Linux 用 USB デバイスドライバの生成において，技術者の負担を軽減することを目的とする．

2．支援対象の Linux 用 USB デバイスドライバ

USB のホストとデバイスを Figure.1 に示す．Figure.1 は Linux においての USB デバイスドライバの構成である．USB 仕様としては大まかにホストとターゲットがある．本研究で支援する箇所は，ホストに位置する Kernel にある USB ドライバである．この USB ドライバが Linux 用 USB デバイスドライバである．

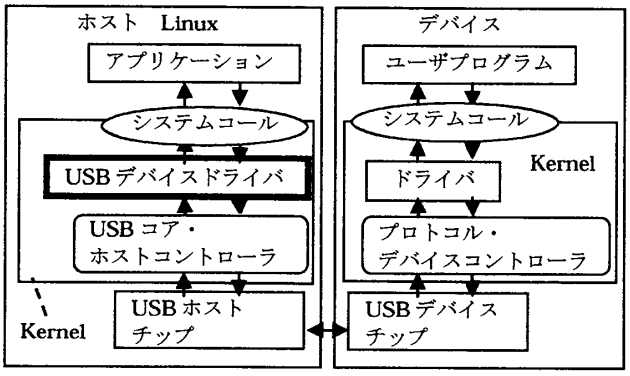

Figure.1　USB のホストとデバイス

USB ドライバはアプリケーションからシステムコールによって呼ばれ，USB コアの関数を呼び出して要求された動作をする．また，USB コアからも呼ばれ，要求された動作をする．

3．支援対象となるデバイスドライバの仕様

デバイスドライバは「入力されたデータをどこへどのように出力するか」が仕様となる．支援対象となるデバイスドライバの仕様を Figure.2 に示す．

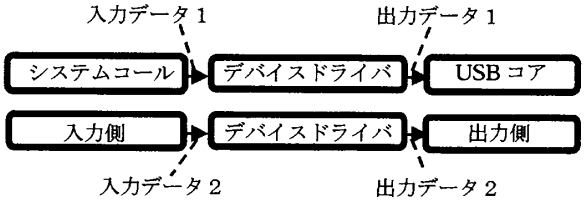

Figure.2　支援対象となるデバイスドライバの仕様

ただし，Linux と USB を対象としているので Linux デバイスドライバの枠組みと USB に合った形となる．ここで Linux デバイスドライバの枠組みとは，file_operation 構造体に定義されている関数である．Linux 用 USB のデバイスドライバ関数を Figure.3 に示す．ただし，これは一例である．

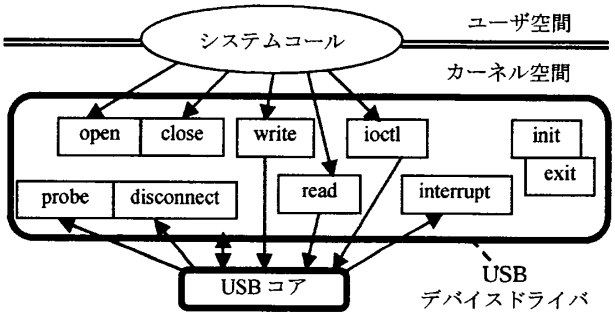

Figure.3　Linux 用 USB のデバイスドライバ関数

よって，デバイスドライバの仕様はデータを送る方向・Linux デバイスドライバの枠組み・データ送信のプロトコル・デバイスの扱い方・データの送信先があげられると考えられる．データを送る方向とは，データを送信するのか受信するのかである．Linux デバイスドライバの枠組みとは，Linux で使用されるデバイスドライバの関数である．データ送信のプロトコルとは，USB の転送方式である．USB の転送方式にはコントロール・バルク・インタラプ

† 東海大学大学院工学研究科電子工学専攻
‡ 東海大学電子情報学部情報メディア学科教授

ト・アイソクロナスの４つがある．デバイスの扱い方とは，送られてきたデータを解析し，加工することである．データの送信先とは，生成するデバイスドライバから，どのモジュールの関数を呼ぶのかということである．

4．Linux用USBデバイスドライバ生成工程

Linux用USBデバイスドライバはデバイスドライバの仕様に沿って生成していく．デバイスドライバ生成の流れをFigure.4に示す．

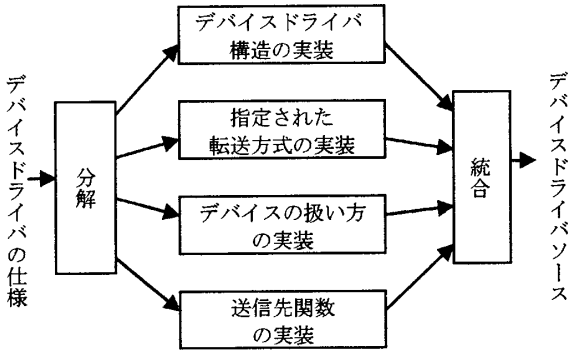

Figure.4　デバイスドライバ生成の流れ

デバイスドライバ生成工程は５つからなる．その５つとは，デバイスドライバ構造の実装・指定された転送方式の実装・デバイスの扱い方の実装・送信先関数の実装・統合である．

デバイスドライバ構造の実装では，デバイスドライバの仕様のデータ送信の方向・Linuxデバイスドライバの枠組みを元にしてデバイスドライバの構造を決定する．指定された転送方式の実装では，デバイスドライバの仕様のデータ送信のプロトコルを元にして指定されたプロトコルを実装する．デバイスの扱い方では，デバイスドライバの仕様のデバイスの扱い方を元にしてデータの解析や加工を実装する．送信先関数の実装では，データの送信先を元にしてモジュールの呼び出し関数を実装する．統合では，その他の４つの工程で出来たものを組合わせることによりデバイスドライバソースを作成する．

5．Linux用USBテンプレート

デバイスドライバ生成工程において，デバイスドライバを生成していく上ですべてを一から作り上げて行くのではなく，必要なものをテンプレートとして持ち，テンプレートを利用してデバイスドライバを生成していく方法を提案する．

テンプレートとするものは，デバイスドライバ構造・USBの転送方式・デバイスの扱い方・送信先の関数である．

デバイスドライバ構造のテンプレートとしては，Linuxデバイスドライバの枠組みをテンプレートとする．USBの転送方式では，USBのプロトコルであるコントロール・バルク・インタラプト・アイソクロナスの４つの転送形式をテンプレートとする．デバイスの扱い方では，エラー処理やデータ解析・加工といった，データの処理方法をテンプレートする．送信先の関数では，呼び出しモジュールの関数と呼び出し方をテンプレートとする．

Linux用USBテンプレートによるデバイスドライバを Figure.5に示す．デバイスドライバ内にある interrupt, read, write, ioctl は Linux デバイスドライバ構造のテンプレートである．Linux デバイスドライバ構造テンプレートの中にある点線で囲まれているものは USB の転送方式・デバイスの扱い方・送信先の関数のテンプレートである．

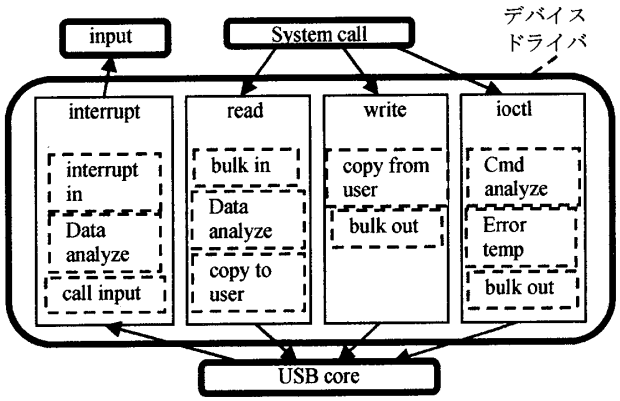

Figure.5　Linux用USBテンプレートによるデバイスドライバ

４つの工程のテンプレートができあがることから，デバイスドライバの開発では統合工程を考えればよいことになる．統合工程は，テンプレートを組み合わせていくことになる．

また，今あるテンプレートではできない処理が要求される場合があるが，テンプレートは追加することができ，処理の幅を広げることができる．

このように，デバイスドライバの生成工程において，テンプレートから仕上げていくことにより，ソースコードを書かずにLinux用USBデバイスドライバを作成していくことができる．

6．おわりに

本報告では，Linux用USBデバイスドライバを題材として生成方法を提案した．今後としては，USB以外の通信方式や，Linux以外のOSへの対応ができるようにしたい．また，技術者不足が叫ばれていることから，教育ツールとしての可能性もあると考えている．

参考文献

1)経済産業省商務情報政策局 情報政策ユニット情報処理振興課, “組込みソフトウェア産業実態調査 報告書”
2)山岡賢一著, ”USB ハード&ソフト開発のすべて”, 発行所 CQ出版株式会社
3)ALESSANDRO RUBINI,JONATHAN CORBET 著, 山崎康, 山崎邦子, 長原宏治, 長原洋子 訳, “Linux デバイスドライバ 第２版”, オライリー・ジャパン
4)ジャン・アクセルソン著, 玉井浩 訳, “USB コンプリート”, 発行所 エスアイビー・アクセス