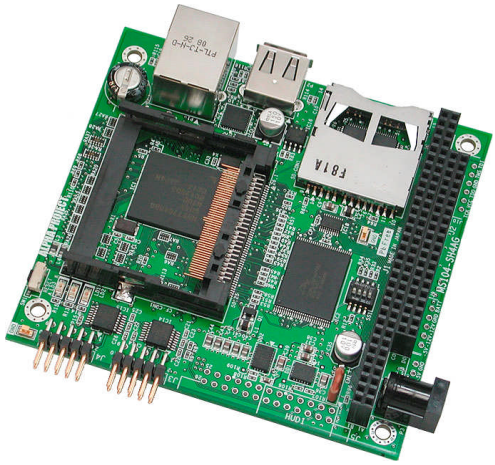
PC/104 規格準拠 SH-4A CPU ボード

Linux 開発キット

ソフトウェアマニュアル
Linux 編
1 版
ダイジェスト版



ALPHAPROJECT
http://www.apnet.co.jp

ご使用になる前に

このたびは MS104-SH4AG Linux 開発キットをお買い上げいただき誠にありがとうございます。
本製品をお役立て頂くために、このマニュアルを十分お読みいただき、正しくお使い下さい。
今後共、弊社製品をご愛顧賜りますよう宜しくお願いいたします。

梱包内容

本製品は、下記の品より構成されております。梱包内容をご確認のうえ、万が一、不足しているものがあれば
お買い上げの販売店までご連絡ください。

MS104-SH4AG Linux 開発キットに付属するもの

●DVD-ROM	1 枚	●LAN ストレートケーブル	1 本
●RS232C クロスケーブル	1 本	●AC アダプタ	1 本
●CF カード	1 枚	●保証書	1 枚

■本製品の内容及び仕様は予告なしに変更されることがありますのでご了承ください。

参考資料

本製品に付属している DVD-ROM には、下記の参考資料が収録されておりますので、本マニュアルと合わせてご覧ください。

■MS104-SH4AG　ソフトウェアマニュアル　U-Boot 編
■MS104-SH4AG　ソフトウェアマニュアル　VMware Player 編

なお、巻末に参考文献も掲載されておりますので是非ご覧ください。

目　次

1. はじめに

MS104-SH4AG は、SH7764 を搭載したボードコンピュータで、標準 OS に Linux を採用しています。
Linux を採用することにより、標準のネットワークプロトコルを利用して容易にネットワーク機器の開発することができます。
また、世界中のプログラマによって日々開発される膨大なオープンソースソフトウェア資産をロイヤリティフリーで利用することができます。
本製品では VMware Player を使用するため、Windows 上で MS104-SH4AG のソフトウェア開発が可能です。

ご注意
本書は VMware Player が WindowsPC にインストールされていることが前提となっています。VMware Player をインストールされていない場合は、『**Linux 開発キット　ソフトウェアマニュアル　VMware Player 編**』をお読みください。

1.1 Linux について

Linux とは 1991 年に Linus Torvalds 氏によって開発された、オープンソースの UNIX 互換オペレーティングシステムです。
Linux はオープンソース、ロイヤリティフリーという特性から、世界中のプログラマたちにより日々改良され、今では大手企業のサーバーや、行政機関などにも広く採用されています。
また、Linux の特長として CPU アーキテクチャに依存しないということがあげられます。これは、GNU C コンパイラの恩恵にもよるものですが、数多くのターゲット(CPU)に移植されており、デジタル家電製品を中心に非 PC 系製品にも採用されるようになりました。
Linux は、カーネルと呼ばれる OS の核となる部分とコマンドやユーティリティなど多くのソフトウェアから構成されます。これらのソフトウェアの多くは FSF の GNU プロジェクトによるフリーソフトウェアです。
本書では Linux のごく一部の機能と使い方のみを説明しています。
Linux の詳細については、一般書籍やインターネットから多くの情報を得られますので、それらを参考にしてください。

1.2 U-Boot について

U-Boot は DENX Software Engineering 社の Wolfgang Denk 氏が保守を行っているオープンソフトウェアの汎用ブートローダです。多くの開発者によって支援され、現在最も機能が豊富で柔軟性に富み、開発が活発に行われています。対応しているアーキテクチャは、SuperH の他に、PPC、ARM、AVR32、MIPS、x86、68k、Nios、MicroBlaze などです。またプログラムのダウンロードに関しても、ネットワークを介した TFTP の他に、CF カード、SD メモリカードなどのストレージデバイスからのダウンロードにも対応しています。

1.3 VMware Player について

VMware Player は VMware Inc によって開発された、仮想マシン実行ソフトウェアであり、無料で使用することが可能です。
VMware Player は Windows/Linux 上で動作する PC/AT 互換機エミュレータです。VMware Player を用いて Windows 上で Linux を動作させたり、Linux 上で Windows を動作させたりすることができます。

本書では VMware Player が動作する WindowsOS を『ホスト OS』、VMware Playaer 上で動作する LinuxOS (Fedora9) を『ゲスト OS』と表現します。また、ホスト OS が動作する PC を『ホスト PC』と表現します。

1.4 GNU と FSF について

GNU プロジェクトとは、UNIX ライクなソフトウェアの自由な流通を目的として 1980 年代半ばに Emacs や GCC の作者である Richard Stallman 氏により立ち上げられました。GNU プロジェクトは、非営利団体の FSF(Free Software Foundation)によって管理運営されており、世界中のボランティアと寄付から成り立っています。GNU ツールのオリジナルはこの FSF を通じて配布されています。

1.5 GPL と LGPL について

Linux を使用する前に GPL(GNU General Public License) と LGPL(GNU Lesser General Public License) について触れておく必要があります。GPL はソフトウェアの自由な流通（コピーレフト）を保証するためのライセンスで、FSF から配布される全てのソフトウェア及び派生物に適用されています。FSF に関連しないソフトウェアでも適用しているものが数多くあります。
以下に概要を記載します。

・頒布、複製、改造等が自由であること
・頒布したソフトウェアにはソースコードを添付するか、提供手段を用意すること
・GPL が適用されたソフトウェアからの派生物（コードの一部を埋め込んだり、改変した著作物）についても GPL を適用すること
・著作権の表示義務
・無保証であること

なお、フリーソフトウェアだから必ず無償でなければいけないということではなく、頒布等のサービスに対して対価を請求することも認められています。ただし、有償で頒布されたものでも、所有者は複製を自由に頒布することができます。
GPL と LGPL の詳しい内容については、GNU プロジェクトのホームページ(http://www.gnu.org)　をご覧ください。

本製品に付属している DVD-ROM 内に GPL と LGPL の原文が収録されておりますので、ご利用になる前に必ずご一読ください。

1.6 保証とサポート

弊社では最低限の動作確認をしておりますが、Linux および付属ソフトウェアの性能や動作を保証するものではありません。
また、これらのソフトウェアについての**個別のお問い合わせ及び技術的な質問は一切受け付けておりませんのでご了承ください。**
なお、疑問点がある場合には、弊社ホームページに設置されております専用掲示板の利用をお勧めします。
個別サポートをご希望されるお客様には、別途有償サポートプログラムをご用意しておりますので、弊社営業までご連絡ください。

付属する GPL ソフトウェア等のソースコードは弊社ホームページより全てダウンロードすることができます。また、これらソフトウェアは不定期にバージョンアップをおこない、ホームページ上で公開する予定です。

専用掲示板 及び ダウンロード用の Web ページアドレス
http://www.apnet.co.jp/e-linux/index.html

2. システム概要

MS104-SH4AG の Linux システムの概要について説明します。

2.1 システム概要

MS104-SH4AG はルネサス社製 SH7764(SH4A)を搭載した PC/104 バス準拠 CPU ボードです。
Linux システムはブートローダと Linux カーネル、ルートファイルシステムから構成されます。ブートローダに U-Boot、Linux カーネルに Linux2.6.25.10、ルートファイルシステムには RAM か CF カードで動作する専用パッケージを使用します。

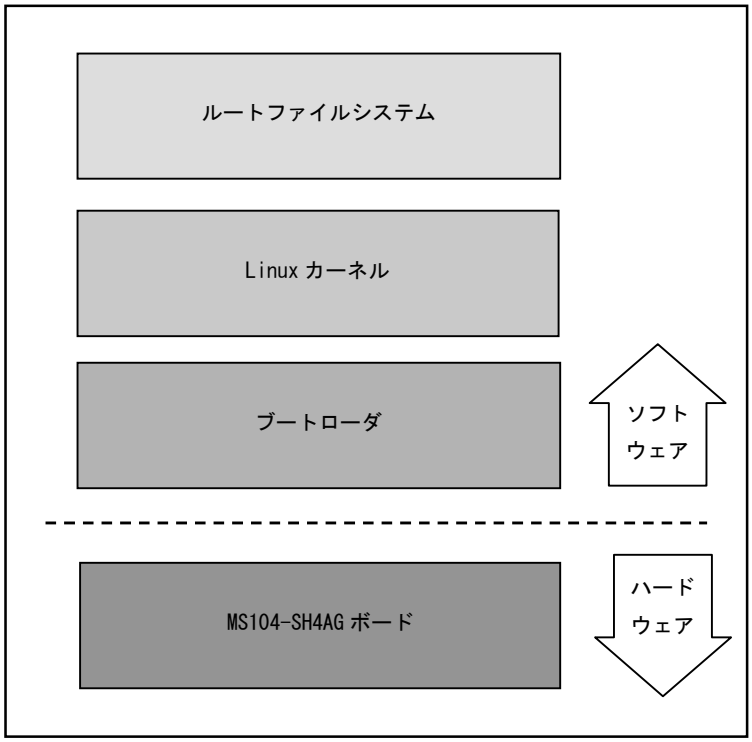


Fig 2.1-1 MS104-SH4AG システム概要図

2.2 ブートローダ

Linux カーネルは RAM 上で動作しますが、カーネル自体は RAM 上に自身をロードする機能を有していません。そのため、Linux カーネルをロードする何らかの手段が必要となります。この手段を提供するのがブートローダです。ブートローダは CPU やメモリ、周辺ハードウェアの初期化を行い、カーネルを RAM 上に展開したあとにカーネルをブートさせます。

MS104-SH4AG のブートローダには U-Boot を使用します。U-Boot の詳細に関しては、同梱の『**Linux 開発キット　ソフトウェアマニュアル　U-Boot 編**』をご覧ください。

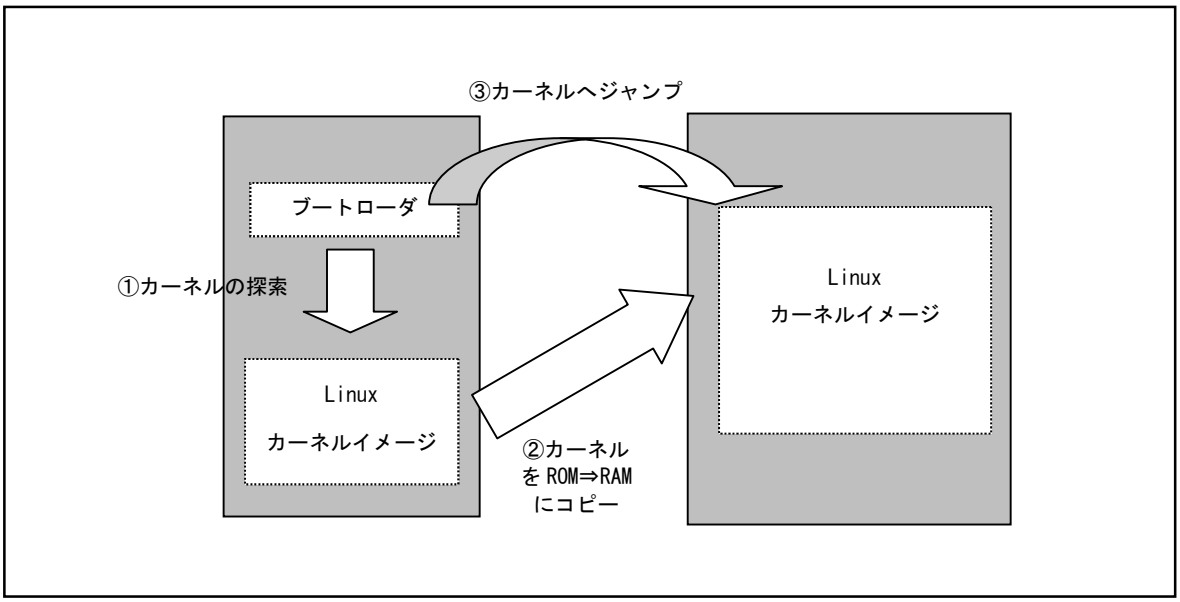


Fig 2.2-1 ブートローダ動作イメージ

2.3 Linux カーネル

Linux カーネルにはプロセス管理、メモリ管理、各種ファイルシステム、ネットワーク機能などがあり、デバイスドライバ自身もカーネルに組み込まれます。Linux カーネル上ではシェル（コマンドプロンプト）や Web サーバなどの多くのアプリケーションが動作します。

本製品では Linux カーネル 2.6.25.10 を使用しています。本製品に添付される Linux カーネルは TCP/IP によるネットワーク機能、各種ファイルシステム(ext2、ext3、NFS、FAT)をサポートしています。

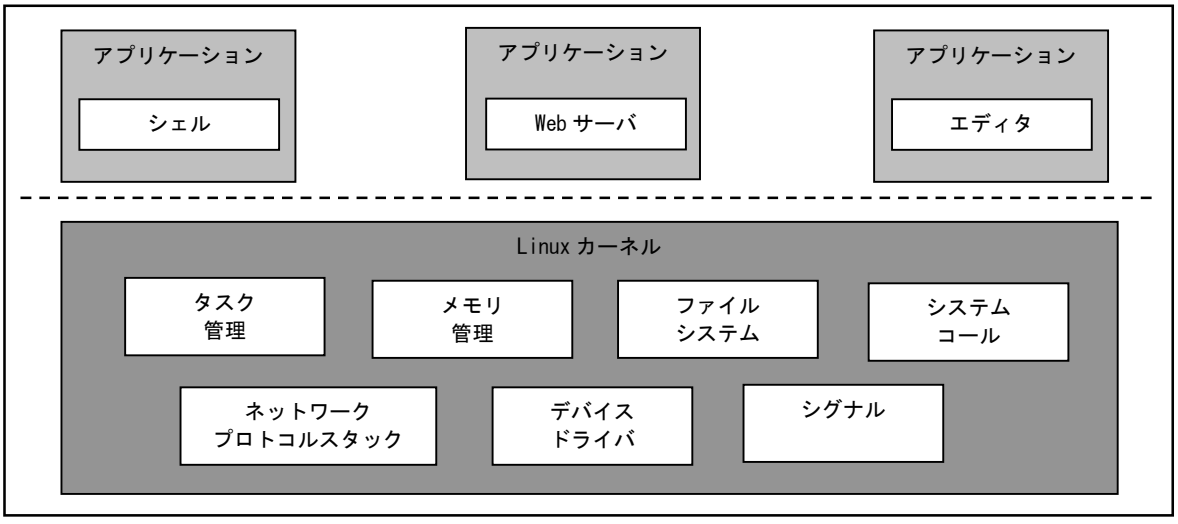


Fig 2.3-1 Linux システム概要

2.4 ルートファイルシステム

Linux は、カーネルとファイルシステムという 2 つの要素から構成されます。
Linux では、全てのデータがファイルという形で管理されています。アプリケーションプログラムやデバイスドライバをはじめ、HDD や COM ポートなどの入出力デバイスもファイルとして扱われます。
Linux では全てのファイルがルートディレクトリを起点としたディレクトリ構造下に管理されており、これら全てのファイル構造のことをファイルシステムと呼びます。また、システム動作に必要なシステムファイル群のこともファイルシステムと呼びます。
本書では、これらの意味を明確にするため、ファイル管理構造(ext2 や ext3)のことをファイルシステム、システム動作に必要なファイル群のことをルートファイルシステムと表現しています。

Linux のルートファイルシステムは、そのシステムが必要とする機能に合わせて構築する必要があります。
MS104-SH4AG では、利用形態に合わせて 2 種類のルートファイルシステムを用意しています。

●ramfs ルートファイルシステム
RAM 上で動作するように構成されたオリジナル Linux パッケージで、Linux カーネル内に組み込まれています。ファイルシステムは tmpfs が使用され、サイズは Linux カーネルと合わせては約 2.5Mbyte で、FlashROM に収まります。CF カードにルートファイルシステムを書き込むときに使用します。ルートファイルシステムが RAM 上に展開されるため、電源を落とすと変更した内容は破棄されます。

●cf ルートファイルシステム
CF カード専用に構成されたオリジナル Linux パッケージで、ファイルシステムには ext2 を採用しています。サイズは約 8.5Mbyte となっております。MS104-SH4AG のすべてのデバイスを動作を確認するためのアプリケーションが含まれています。ルートファイルシステムが CF カード上に展開されるため、電源を落しても変更した内容は破棄されません。

本書では ramfs ルートファイルシステムを利用した Linux システムを RAMFS-Linux システム、cf ルートファイルシステムを利用した Linux システムを CF-Linux システムと表現します。

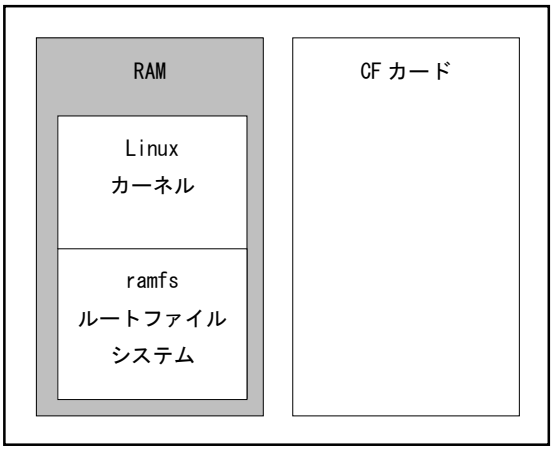


Fig 2.4-1 RAMFS-Linux システム

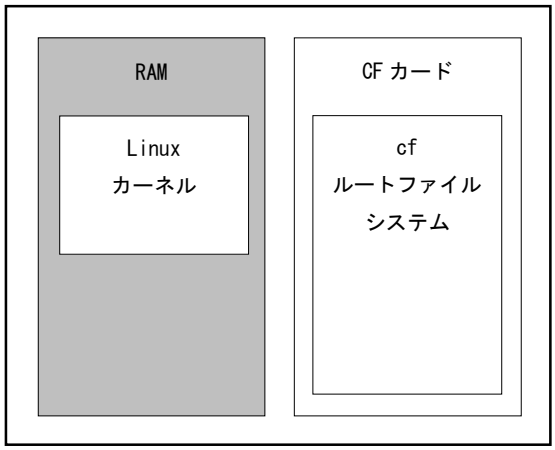


Fig 2.4-2 CF-Linux システム

3. クロス開発環境

本章ではクロス開発環境の構築方法について説明します。

ご注意

『Linux 開発キット　ソフトウェアマニュアル　U-Boot 編　3. クロス開発環境』を実行されている場合、クロス開発環境はインストールされていますので、本章の内容を実行する必要はありません。

3.1　クロス開発環境概要

MS104-SH4AG 上で動作する Linux カーネルやアプリケーションプログラムを作成するには Linux の動作する PC/AT 互換機上でクロス開発環境を構築する必要があります。クロス開発環境を構築するには LinuxOS ※1 上にターゲット用の下記のパッケージをインストールする必要があります。

GNU binary utilities（アセンブラ、リンカ等）

GNU Compiler Collection（クロスコンパイラ・プリプロセッサ等）

uClibc（C 標準ライブラリ等） ※2

上記のパッケージによりターゲット用の実行ファイルを作成することができます。実行ファイルは LinuxOS からターゲットシステムにダウンロードし、動作を確認します。

※1　本書では LinuxOS として VMware Player 上で動作する『Fedora9』を使用します。詳細は『Linux 開発キット　ソフトウェアマニュアル　VMware Player 編』を参照してください。

※2　『uClibc』は組込み用途向け C 標準ライブラリで、通常 Linux システムで使用される『Glibc』よりも容量を必要としないためメモリサイズに制限がある場合などに使用されます。

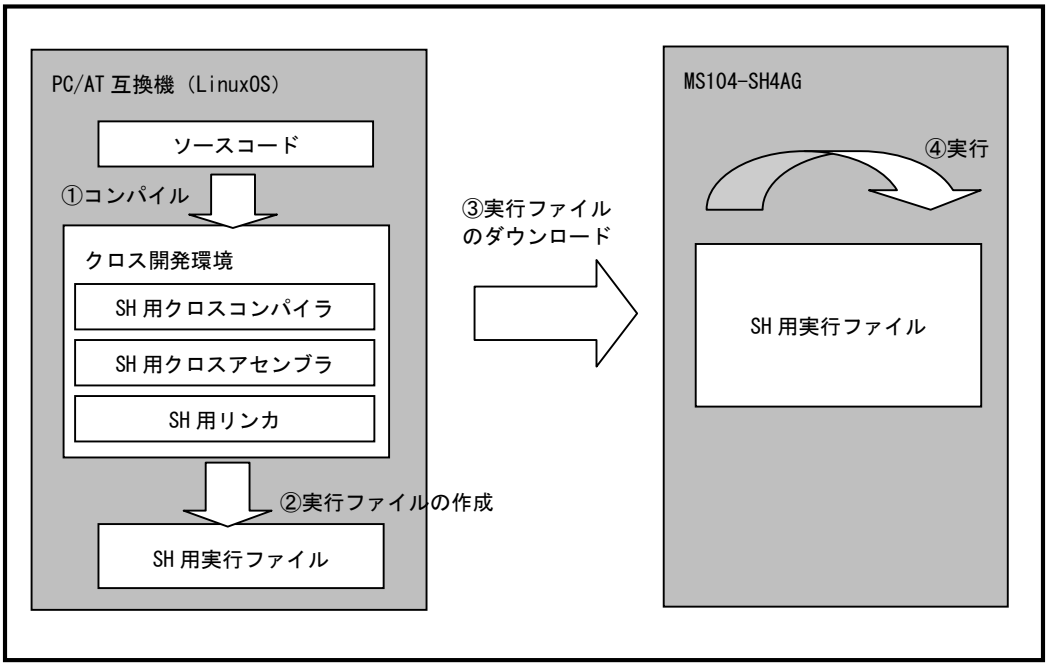


Fig 3.1-1 クロス開発環境

3.2 動作環境

クロス開発環境を構築するには、VMware Player がインストールされた Windows2000、WindowsXP もしくは WindowsVista が動作する PC/AT 互換機が必要になります。
VMware Player をインストールするには『**Linux 開発キット ソフトウェアマニュアル VMware Player 編**』をご覧ください。
また、VMware Player およびクロス開発環境をインストールするには、5GByte 以上のディスク容量が必要です。VMware Player のゲスト OS 用仮想ディスク容量の上限は 16GByte になりますので、16GByte 以上の空き容量を推奨します。

使用機器等	環　　境
PC	PC/AT 互換機
OS	Windows2000/XP/Vista（推奨 WindowsXP）
空き容量	5GByte 以上（推奨 16GByte）
メモリ	512MByte 以上
ソフトウェア	VMware Player ターミナルソフト
DVD ドライブ	DVD 読み込み可能なドライブ
その他	シリアルポート　　1ch LAN ポート　　1ch

Table 3.2-1 クロス開発環境の動作環境

3.3 添付 DVD-ROM の構成

MS104-SH4AG の Linux の開発には、Linux カーネルソース、Buildroot ソースファイル、クロスコンパイラ等が必要です。
これらは、弊社ホームページ及び関連リンクからダウンロードするか、添付 DVD-ROM から入手することができます。

```
L_KIT_C01_VX_X
|-- applicationnote                    : アプリケーションノート
|-- binaries
|   |-- vmlinuz-ms104sh4ag-ramfs       : Linux カーネルバイナリファイル（ramfs 付き）
|   |-- vmlinuz-ms104sh4ag-ramfs.img   : Linux カーネル U-Boot 用イメージ（ramfs 付き）
|   |-- vmlinuz-ms104sh4ag             : Linux カーネルバイナリファイル
|   |-- vmlinuz-ms104sh4ag.img         : Linux カーネル U-Boot 用イメージ
|   |-- cf-ms104sh4ag.tar.gz           : cf ルートファイルシステム
|   |-- ramdisk-ms104sh4ag.cpio.gz     : ramfs ルートファイルシステム
|   `-- u-boot-ms104sh4ag.bin          : U-Boot バイナリファイル
|-- index.html                         : インデックス HTML
|-- index_images                       : インデックス HTML イメージ
|-- license
|   |-- fdl.txt                        : GFDL 原文
|   |-- gpl.txt                        : GPL 原文
|   `-- lgpl.txt                       : LGPL 原文
|-- manual
|   |-- ALSA                           : ALSA マニュアル
|   |-- DirectFB                       : DirectFB マニュアル
|   |-- ms104sh4ag_linux.pdf           : Linux 開発キット　ソフトウェアマニュアル　Linux 編
|   |-- ms104sh4ag_uboot.pdf           : Linux 開発キット　ソフトウェアマニュアル　U-Boot 編
|   `-- ms104sh4ag_vmware.pdf          : Linux 開発キット　ソフトウェアマニュアル　VMware Player 編
|-- sources
|   |-- buildroot-ms104sh4ag.tar.gz    : Buildroot ソースファイル
|   |-- external-toolchain.tar.gz      : クロスコンパイラ
|   |-- install_ms104sh4ag.sh          : クロス開発環境インストールシェルスクリプト
|   |-- uninstall_ms104sh4ag.sh        : クロス開発環境アンインストールシェルスクリプト
|   |-- linux-2.6.25.10-alp.tar.gz     : Linux カーネルソースファイル
|   |-- u-boot-1.3.3-alp.tar.gz        : U-Boot ソースファイル
|   |-- ms104sh4ag-sample.tar.gz       : MS104-SH4AG サンプルプログラム
|   |-- ms104lcdaudio-sample.tar.gz    : MS104-LCD/AUDIO サンプルプログラム
|   `-- ms104fpga-sample.tar.gz        : MS104-FPGA/CⅢサンプルプログラム
`-- vmware
    |-- image16g.zip                   : VMware 仮想ディスク圧縮ファイル
    |-- disk.vmx                       : VMware 仮想マシン構成ファイル
    `-- VMware-player-2.5.1-126130.exe : VMware Player インストーラ
```

Table 3.3-1　DVD-ROM 構成

※　『VX_X』はバージョン番号を示します。バージョン 1.0 の場合は『V1_0』になります。

4. Linux の起動

本章では Linux の起動方法について説明します。

4.1 動作環境

Linux の起動を確認するためには、CPU ボードと以下の環境が必要です。

●ホスト PC

Linux では PC をコンソール端末として使用します。また、シリアルポートが使用可能な PC が必要となります。
PC では、ハイパーターミナル等のターミナルソフトウェアを動作させます。

●電源

MS104-SH4AG 本体に必要な電源は DC5V±5%です。単体で動作させる場合には AC アダプタを用意してください。

●LAN

MS104-SH4AG をネットワークに接続する場合は、LAN ケーブルを接続してください。直接ホスト PC と接続する際はクロスケーブル、ハブを介してネットワークに接続する際はストレートケーブルをご使用ください。
LAN ケーブルは、10/100BASE-TX 対応（UTP カテゴリ 5）ケーブルをご利用ください。

使用機器等	環　　境
CPU ボード	MS104-SH4AG
HOST PC	PC/AT 互換機
OS	Windows2000/XP
メモリ	使用 OS による
ソフトウェア	ターミナルソフト
シリアルポート	1 ポート
USB ポート	1 ポート
LAN ポート	10/100BASE-TX　1 ポート
RS232C ケーブル	クロスケーブルを使用
D-Sub 変換ケーブル	付属品
LAN ケーブル	ホスト PC と接続時はクロスケーブルを使用 ハブと接続時はストレートケーブルを使用
電源	AC アダプタ（DC5V±5%　1A 以上）

Table 4.1-1　動作環境

4.2 シリアル設定

Linuxはシリアル通信を介した対話型コマンドコンソールを持ちます。シリアルからコマンドを入力することで操作が可能です。
以下にMS104-SH4AG用Linuxのシリアル通信の初期設定を記します。

ポート番号	COM1（J3コネクタ）
通信速度	38400bps
データ長	8bit
ストップビット	1bit
パリティ	なし
フロー制御	なし

Table 4.2-1 シリアル初期設定

4.3 ネットワーク設定

MS104-SH4AG用Linuxはネットワークに対応しています。
Linux、ゲストOS（Fedora9）およびホストOS（Windows）のデフォルトネットワーク設定は以下になります。

ネットワークの設定	
IPアドレス	192.168.128.200
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.128.254
DNSサーバ	なし

Table 4.3-1 Linuxデフォルトネットワーク設定

ネットワークの設定	
IPアドレス	192.168.128.201
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.128.254
DNSサーバ	192.168.128.1

Table 4.3-2 ゲストOSデフォルトネットワーク設定

ネットワークの設定	
IPアドレス	192.168.128.202
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.128.254
DNSサーバ	192.168.128.1

Table 4.3-3 ホストOSデフォルトネットワーク設定

※ 本書では上記のネットワーク設定を使用して説明を進めます。
※ ホストOS（WindowsOS）は外部ネットワークと遮断し、上記のネットワーク設定に変更してください。

4.4 MS104-SH4AG ボードの接続

ホスト PC と MS104-SH4AG ボードの接続例を示します。
LAN をネットワークと接続する場合は、ネットワーク管理者と相談し、設定に注意して接続してください。
D-Sub 変換ケーブルは、MS104-SH4AG ボードの COM1（J3）に接続してください。

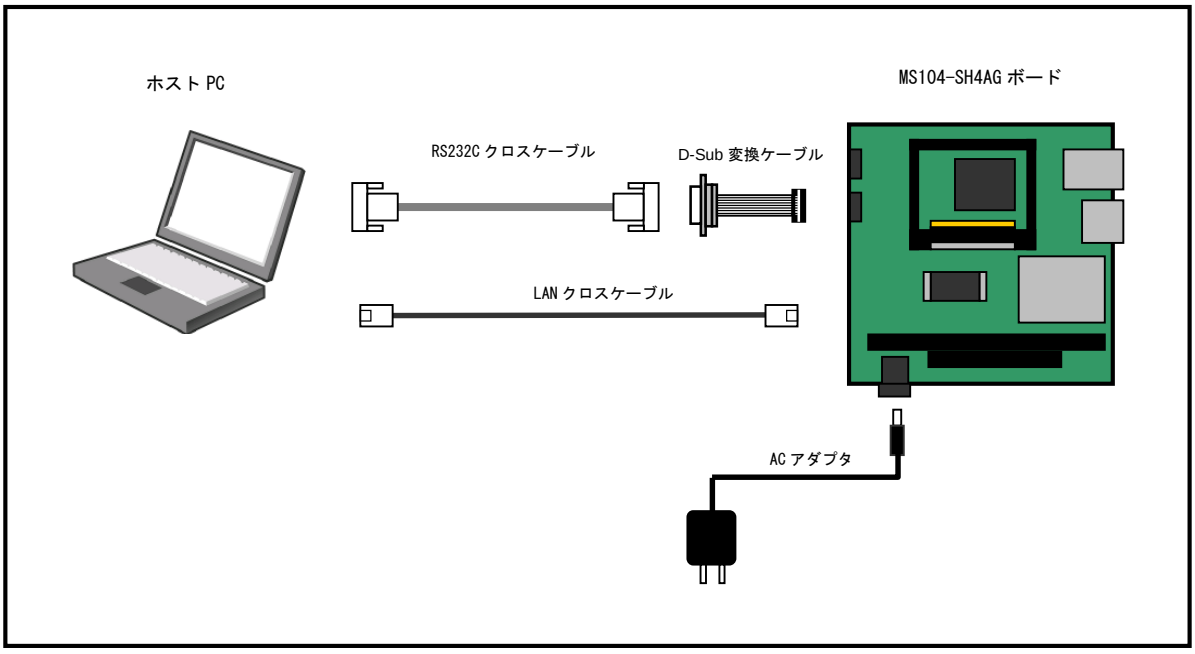


Fig 4.4-1 MS104-SH4AG ボードの接続（PC に接続する場合）

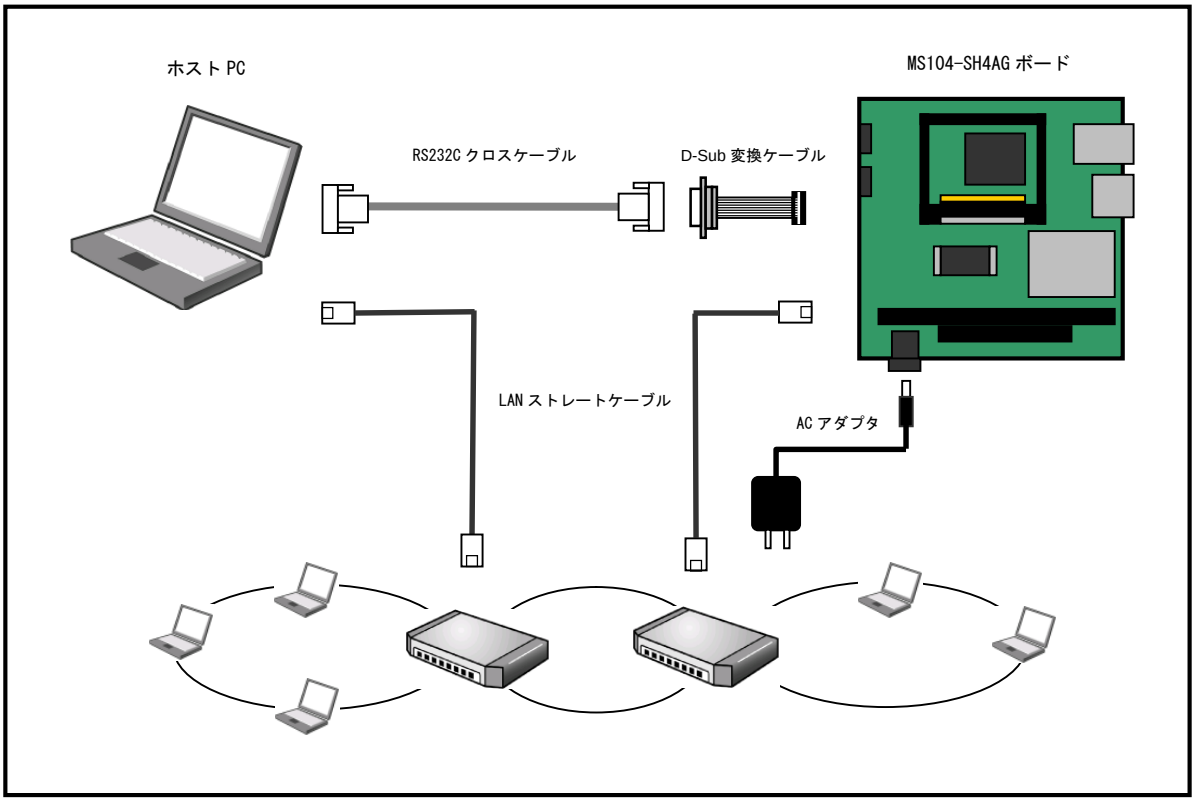


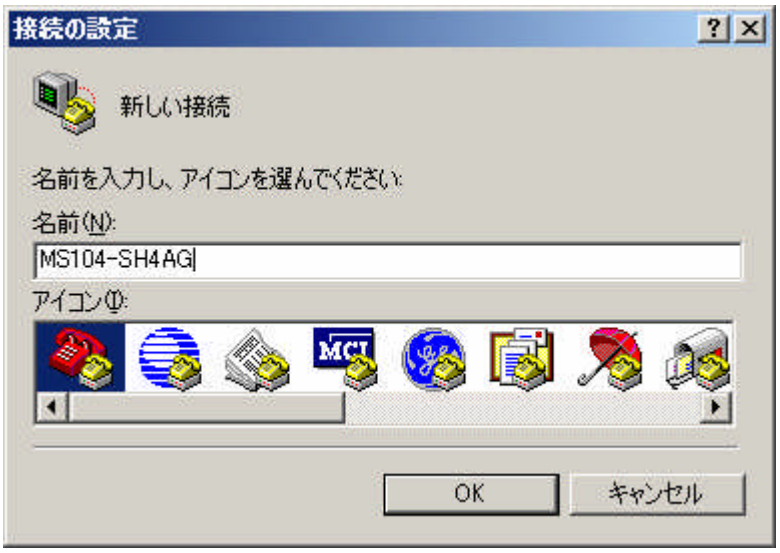
Fig 4.4-2 MS104-SH4AG ボードの接続（HUB に接続する場合）

4.5 ターミナルの設定

MS104-SH4AG からのコンソール出力を受け取るホスト OS（Windows）のターミナルソフトの設定・起動方法について説明します。

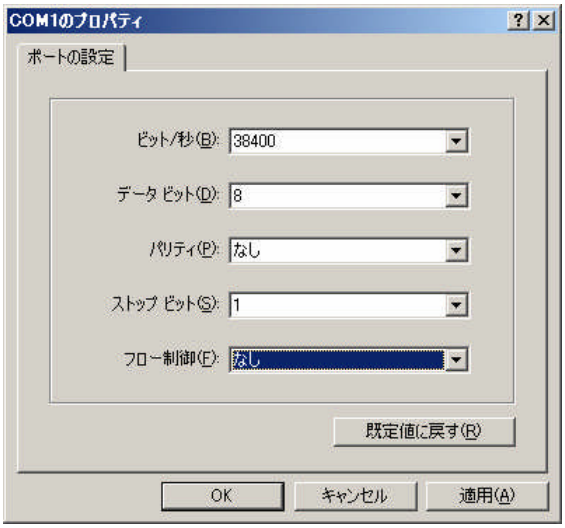
① Windows のスタートメニューから、『**プログラム**』-『**アクセサリ**』-『**通信**』-『**ハイパーターミナル**』を選択し、ハイパーターミナルを起動します。

② 『**名前**』を入力し、『**OK**』ボタンを押します。



③ シリアルケーブルが接続されている『**COM ポート**』を選択し、『**OK**』ボタンを押します。

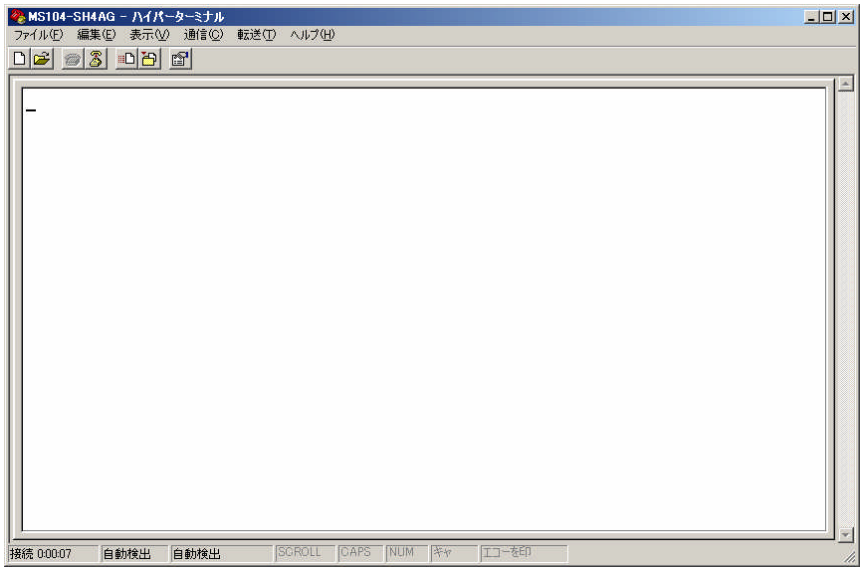
④　『**ポートの設定**』の設定を行い、『**OK**』ボタンを押します。



●ポートの設定

[ビット/秒]	38400
[データビット]	8
[パリティ]	なし
[ストップビット]	1
[フロー制御]	なし

⑤　ハイパーターミナルが起動します。



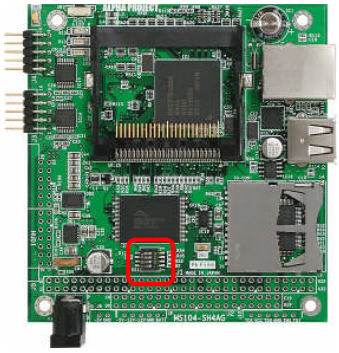
4.6 Linux の起動

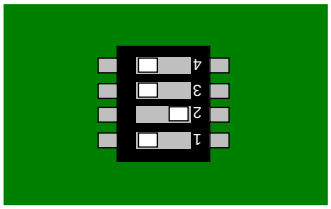
MS104-SH4AG 上で Linux の起動を行います。
MS104-SH4AG は出荷時状態で Linux が自動起動します。

① ホスト OS（Windows）のターミナルソフトを起動します。（設定は『**4.5 ターミナルの設定**』を参照してください）

② 『**4.4 MS104-SH4AG の接続**』にしたがって、ホスト PC と MS104-SH4AG のシリアルポート COM1（J3）とイーサネットポートを接続します。

③ MS104-SH4AG のディップスイッチが以下のようになっていることを確認します。
ディップスイッチの各設定の詳細に関しては、『**MS104-SH4AG ハードウェアマニュアル**』でご確認ください。





④ MS104-SH4AG の電源を投入します。
AC アダプタを接続すると MS104-SH4AG の電源が入ります。電源が投入されると、約 2 秒後に Linux カーネルが自動起動します。全ての起動までにはおよそ 10 秒ほどかかります。

```
U-Boot 1.3.3-svn (Dec 10 2008 - 12:00:00)

CPU: SH4
BOARD: SH7764 ALPHAPROJECT MS104-SH4AG
DRAM:  64MB
FLASH: 16MB
In:    serial
Out:   serial
Err:   serial
Net:   miiphy_register: added 'DP83848J', read=0x87f069a8, write=0x87f0698a
Hit any key to stop autoboot:  0
## Booting kernel from Legacy Image at a00a0000 ...
   Image Name:   ms104sh4ag-linux-ramfs
   Created:      2008-12-10   12:00:00 UTC
   Image Type:   SuperH Linux Kernel Image (uncompressed)
   Data Size:    2666520 Bytes =  2.5 MB
   Load Address: 84400000
   Entry Point:  84400000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK
Uncompressing Linux... Ok, booting the kernel.

... 中略

Welcome to the Erik's uClibc development environment.
uclibc login:
```

⑤ Linux 起動後、ログインプロンプト『**uclibc login:**』が表示されます。
ログインを実行するため、『**root**』を入力してください。

```
Welcome to the Erik's uClibc development environment.
uclibc login: root    入力
#
```

Table 4.6-1　Linux の起動ログ

```
Uncompressing Linux... Ok, booting the kernel.
Linux version 2.6.25.10 (guest@fedora9) (gcc version 4.2.4) #2 Thu Dec 11 13:34:21 JST 2008
Booting machvec: MS104-SH4AG
Node 0: start_pfn = 0x4000, low = 0x8000
Zone PFN ranges:
  Normal      16384 ->    32768
Movable zone start PFN for each node
early_node_map[1] active PFN ranges
    0:    16384 ->    32768
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 16256
Kernel command line: console=ttySC0,38400
MS104-SH4AG IRQ setup ...
PID hash table entries: 256 (order: 8, 1024 bytes)
Using tmu for system timer
Using 13.500 MHz high precision timer.
Console: colour dummy device 80x25
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Memory: 61080k/65536k available (1914k kernel code, 778k data, 908k init)
PVR=10300800 CVR=73440400 PRR=00001020
I-cache : n_ways=4 n_sets=256 way_incr=8192
I-cache : entry_mask=0x00001fe0 alias_mask=0x00001000 n_aliases=2
D-cache : n_ways=4 n_sets=256 way_incr=8192
D-cache : entry_mask=0x00001fe0 alias_mask=0x00001000 n_aliases=2
SLUB: Genslabs=10, HWalign=32, Order=0-1, MinObjects=4, CPUs=1, Nodes=1
Mount-cache hash table entries: 512
CPU: SH7764
net_namespace: 152 bytes
NET: Registered protocol family 16
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
MS104-SH4AG Setup...
Total HugeTLB memory allocated, 0
JFFS2 version 2.2. (NAND) c 2001-2006 Red Hat, Inc.
io scheduler noop registered (default)
VRAM=87600000
sh7764fb loaded
g2d_thread: >>
SuperH SCI(F) driver initialized
sh-sci: ttySC0 at MMIO 0xffe00000 (irq = 43) is a scif
console [ttySC0] enabled
sh-sci: ttySC1 at MMIO 0xffe10000 (irq = 79) is a scif
sh-sci: ttySC2 at MMIO 0xffe20000 (irq = 107) is a scif
brd: module loaded
loop: module loaded
SH7764 Ethernet driver
eth0: SH7764 Ethernet Controller 00:0c:7b:25:00:02, IRQ 57
Uniform Multi-Platform E-IDE driver
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
    ide0: SG-DMAProbing IDE interface ide0...
hda: , ATA DISK drive
hda: applying conservative PIO "downgrade"
hda: PIO1 selected (peak 5MB/s) 400 ns
hda: no DMA mode selected
hda: applying conservative PIO "downgrade"
hda: PIO1 selected (peak 5MB/s) 400 ns
ide0 at 0x1f0-0x1f7,0x3f6 on irq 80
hda: max request size: 128KiB
hda: 2030112 sectors (1039 MB) w/1KiB Cache, CHS=2014/16/63
 hda: hda1
MS104SH4AG IDE(builtin)
Driver 'sd' needs updating - please use bus_type methods
m66597-hcd m66597-hcd.0: USB Host Controller
m66597-hcd m66597-hcd.0: new USB bus registered, assigned bus number 1
m66597-hcd m66597-hcd.0: irq 83, io base 0xfe400000
usb usb1: configuration #1 chosen from 1 choice
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
USB M66597 Hispeed detect check interval = 3000
Buffer Transfer Mode = BulkOut PIO:BulkIn PIO
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
s35190a s35190a: rtc core: registered s35190a as rtc0
i2c /dev entries driver
```

```
tsc2007_i2c_probe: client[name= flags=0000 addr=0048]
set_type_ms104sh4ag_irq: irq=129 flow_type=00000008
input: TSC2007 Touchscreen as /devices/platform/sh7764-i2c.0/i2c-adapter/i2c-0/0-0048/input/input0
tsc2007 0-0048: touchscreen, irq 129
sh7764-i2c sh7764-i2c.0: 400 kHz ICCCR=0e mmio ffe70000-ffe70024 irq 53
mmc_spi spi0.0: ASSUMING 3.2-3.4 V slot power
SD Card detect thread started
mmc_spi spi0.0: SD/MMC host mmc0, no DMA
Registered led device: led0
Registered led device: led1
Advanced Linux Sound Architecture Driver Version 1.0.16rc2 (Thu Jan 31 16:40:16 2008 UTC).
ALSA device list:
  #0: MS104 SSI-0
  #1: MS104 SSI-1
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
s35190a s35190a: setting system clock to 2008-12-16 15:52:51 UTC (1229442771)
Freeing unused kernel memory: 908 ｋnitializing random number generator... done.
Starting network...



Welcome to the Erik's uClibc development environment.
uclibc login:
```

※　環境やバージョンによって若干異なる場合があります。

4.7 Linux の動作確認

MS104-SH4AG 上での Linux の動作確認を行います。

ログイン

Linux 起動後、ログインプロンプト『**uclibc login:**』が表示されます。
ログインを実行するにはユーザ『**root**』を入力してください。

ログイン設定	
ユーザ	root
パスワード	なし

Table 4.7-1 ログイン設定

```
Welcome to the Erik's uClibc development environment.
uclibc login: root    入力
#
```

時刻設定

MS104-SH4AG 上で時刻の設定をします。MS104-SH4AG には RTC(リアルタイムクロックが)搭載されており、電源を OFF にして状態でも時刻を保持することができます。Linux は起動時に RTC から時刻を読み出し、以後は RTC にアクセスすることなく、CPU 内のタイマーモジュールによって時刻を管理しています。Linux のコマンドライン上から RTC にアクセスするには『**hwclock**』コマンドを使用します。

① RTC に設定されている時刻を読み出すには『**hwclock**』コマンドを引数無しで入力します。

```
# hwclock    入力
Tue Dec  10 12:00:00 2008  0.000000 seconds
#
```

② RTC に設定されている時刻を変更する際には『**date**』コマンドを使用し、システムの時刻を設定し、その更新されたシステムの時刻を『hwclock』コマンドで RTC に書き込みます。
例として時刻を 2008 年 12 月 3 日 15 時 30 分に設定します。
『**date -s '2008-12-03 15:30'**』実行後、『**hwclock -w**』を実行してください。

```
# date -s '2008-12-03 15:30'    入力
Wed Dec  3 15:30:00 UTC 2008
# hwclock -w    入力
#
```

SD メモリカード

SD メモリカードをファイルシステム上の任意のディレクトリにマウントすることにより、他のファイルと同様にアクセスすることができます。活線挿抜に対応しているため、電源を ON にした状態でもカードの抜き差しが可能です。

① SD メモリカードをスロットに差し込むと以下のようなメッセージがコマンドライン上に出力されます。

```
# mmc0: new SD card on SPI
mmcblk0: mmc0:0000 SQ01G 967680KiB
 mmcblk0: p1
```

② FAT ファイルシステムでフォーマットされている SD メモリカードを『/mnt/sd』ディレクトリにマウントします。
『mount -t vfat /dev/mmcblk0p1 /mnt/sd』コマンドを実行してください。

```
# mount -t vfat /dev/mmcblk0p1 /mnt/sd     入力
#
```

③ ls コマンドで内容を確認します。

```
# ls /mnt/sd     入力
a.txt
#
```

④ 『umount』コマンドで SD カードをアンマウント(マウント解除)することができます。カードを抜き出す際には必ず SD カードをアンマウントしてください。
『umount /mnt/sd』を実行してください。

```
# umount /mnt/sd     入力
#
```

⑤ カードを抜き出すと以下のようなメッセージがコマンドライン上に出力されます。

```
# mmc0: SPI card removed
```

CF カード

CF カードをファイルシステム上の任意のディレクトリにマウントすることにより、他のファイルと同様にアクセスすることができます。活線挿抜には対応していないため、カードの抜き差しは電源を OFF にした状態で行う必要があります。

① ext2 ファイルシステムでフォーマットされている CF カードを『/mnt/cf』ディレクトリにマウントします。
『mount /dev/hda1 /mnt/cf』コマンドを実行してください。

```
# mount /dev/hda1 /mnt/cf     入力
#
```

② ls コマンドで内容を確認します。

```
# ls /mnt/cf     入力
a.txt
#
```

③ 『umount』コマンドで CF カードをアンマウント(マウント解除)することができます。
『umount /mnt/cf』を実行してください。

```
# umount /mnt/cf     入力
#
```

USB メモリ

USB メモリをファイルシステム上の任意のディレクトリにマウントすることにより、他のファイルと同様にアクセスすることができます。活線挿抜に対応しているため、電源を ON にした状態でもカードの抜き差しが可能です。

① USB メモリを USB コネクタに差し込むと以下のようなメッセージがコマンドライン上に出力されます。Linux では、USB メモリは SCSI デバイスとして認識されます。出力されるメッセージは環境により異なります。

```
# usb 1-1: new high speed USB device using m66597-hcd and address 2
usb 1-1: configuration #1 chosen from 1 choice
scsi0 : SCSI emulation for USB Mass Storage devices
scsi 0:0:0:0: Direct-Access     XXXXXXXX USB Flash Disk   A4   PQ: 0 ANSI: 2
sd 0:0:0:0: [sda] 128000 512-byte hardware sectors (66 MB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] 128000 512-byte hardware sectors (66 MB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
 sda: unknown partition table
sd 0:0:0:0: [sda] Attached SCSI removable disk
sd 0:0:0:0: Attached scsi generic sg0 type 0
```

② FAT ファイルシステムでフォーマットされている USB メモリを『/mnt/usb』ディレクトリにマウントします。
『mount -t vfat /dev/sda1 /mnt/usb』コマンドを実行してください。

```
# mount -t vfat /dev/sda1 /mnt/usb     入力
#
```

③ ls コマンドで内容を確認します。

```
# ls /mnt/usb     入力
a.txt
#
```

④ 『umount』コマンドで CF カードをアンマウント(マウント解除)することができます。USB メモリをコネクタから引き抜くときは必ずアンマウントを実行してください。
『umount /mnt/sd』を実行してください。

```
# umount /mnt/usb     入力
#
```

4.8 ネットワークの設定

Linux のネットワーク設定を変更する方法および Web サーバへのアクセス、NFS の使用方法について説明します。

ネットワーク設定の確認

ネットワーク設定を確認する方法について説明します。

① Linux の IP アドレス・サブネットマスクを確認するため、『**ifconfig**』と入力してください。
表示された『**eth0**』の項目内の『**inet addr**』が IP アドレス、『**Mask**』がサブネットマスクとなります。

```
# ifconfig        入力
eth0      Link encap:Ethernet  HWaddr 00:0C:7B:25:00:02
          inet addr:192.168.128.200  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4105 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:415293 (405.5 KiB)  TX bytes:0 (0.0 B)
          Interrupt:57

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

#
```

サブネットマスク

IPアドレス

② 『**route**』コマンドでゲートウェイの設定を確認することができます。
default の行が、デフォルトゲートウェイの設定です。

```
# route        入力
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.128.0   *               255.255.255.0   U     0      0        0 eth0
default         192.168.128.254 0.0.0.0         UG    0      0        0 eth0
#
```

ゲートウェイ

③ DNS サーバのアドレスを確認するには『**/etc/resolv.conf**』ファイルを開きます。
『**vi /etc/resolv.conf**』を実行し、エディタを起動します。
デフォルトで DNS は設定されていないので、『**/etc/resolv.conf**』ファイルは空になります。

```
# vi /etc/resolv.conf
```

『**/etc/resolv.conf**』ファイル

ネットワーク設定の変更

ネットワーク設定を変更する方法について説明します。ネットワーク設定を変更する場合には、ネットワーク設定ファイル『vi /etc/network/interfaces』を書き換える必要があります。

※ ramfs ルートファイルシステムは RAM 上で動作するため、電源を落とすと変更した設定が破棄されます。

① IP アドレス、サブネットマスク、ゲートウェイを変更するには『/etc/network/interfaces』ファイルを編集します。
『vi /etc/network/interfaces』を実行し、エディタを起動します。

```
# vi /etc/network/interfaces
```

『/etc/network/interfaces』ファイル

```
# Configure Loopback
auto lo eth0

iface lo inet loopback

iface eth0 inet static
address 192.168.128.200          : IP アドレス
netmask 255.255.255.0            : サブネットマスク
gateway 192.168.128.254          : ゲートウェイ
```

●IP アドレスの設定
IP アドレスの設定は『address IP アドレス』で設定します。

●サブネットマスクの設定
サブネットマスクの設定は『netmask サブネットマスク』で設定します。

●ゲートウェイの設定
ゲートウェイのアドレスの設定は『gateway ゲートウェイアドレス』で設定します。

② DNS サーバのアドレスを変更するには『/etc/resolv.conf』ファイルを開きます。
『vi /etc/resolv.conf』を実行し、エディタを起動します。
『192.168.128.1』の DNS サーバを追加するため、『/etc/resolv.conf』ファイルに『nameserver 192.168.128.1』を追加します。

```
# vi /etc/resolv.conf        入力
```

『/etc/network/interfaces』ファイル

```
nameserver 192.168.128.1         : DNS
```

●DNS の設定
DNS のアドレスの設定は『nameserver DNS アドレス』で設定します。

③ ネットワーク設定を反映させます。
『ifdown eth0』コマンドでネットワークを停止し、『ifup eth0』コマンドでネットワークを再開します。

```
# ifdown eth0      入力
# ifup eth0        入力
#
```

Web サーバ

Web サーバは Linux 起動時に自動的に立ち上がります。Web ブラウザで Linux のデフォルト IP アドレス『192.168.128.200』を参照すれば、デモページを参照することができます。
Web サーバのデモページは『/var/www/html』ディレクトリに保存されています。

※　デモページが開けない場合は『ping』コマンドを実行し、正しくネットワークの設定が行われているかご確認ください。

Fig 4.8-1　MS104-SH4AG デモページ

NFS

Linux カーネルの機能により NFS（Network File System：ネットワークを介した分散ファイルシステム）を利用することができます。NFS は利用すれば Linux 上にある共有ディレクトリ内のファイルを共有することができます。
ゲスト OS(Fedora9)上の NFS 共有ディレクトリ『/nfs』を MS104-SH4AG からマウントします。

※　NFS を使用するには、ゲスト OS が起動し、NFS サーバを有効にする必要があります。

①　NFS 共有ディレクトリをマウントするには『mount -t nfs -o nolock NFS サーバ IP アドレス:共有ディレクトリ名 マウント先ディレクトリ』と入力します。
『mount -t nfs -o nolock 192.168.128.201:/nfs /mnt/nfs』を実行してください。

```
# mount -t nfs -o nolock 192.168.128.201:/nfs /mnt/nfs
#
```
入力

②　『umount』コマンドで NFS をアンマウント(マウント解除)することができます。
『umount /mnt/nfs』を実行してください。

```
# umount /mnt/nfs
#
```
入力

5. RAMFS-Linux システム

本章では RAMFS-Linux システムの作成・使用方法について説明します。

5.1 RAMFS-Linux システムの概要

RAMFS-Linux システムは RAM 上にルートファイルシステムを展開し、動作します。RAMFS-Linux システムのルートファイルシステムは cpio フォーマットでアーカイブされたファイルを gzip で圧縮したファイル形式になります。
cpio フォーマットの展開コードは Linux カーネルに含まれているため、ramfs ルートファイルシステムを Linux カーネル内に組み込むことができ、ファイルシステムを解析するためのモジュールを必要としません。ramfs ルートファイルシステムは必要なサイズに応じて RAM の容量が動的に変化するため効率的に RAM を使用することができます。
RAMFS-Linux システムは RAM 上で動作するため、電源を落とすとルートファイルシステムの内容は消えてしまい、再度 Linux を起動するときには初期値に戻ります。
ramfs ルートファイルシステムは Linux カーネルのコンパイル時にカーネル Linux カーネルイメージにリンクされているため、起動時に別途ルートファイルシステムを用意する必要がありません。

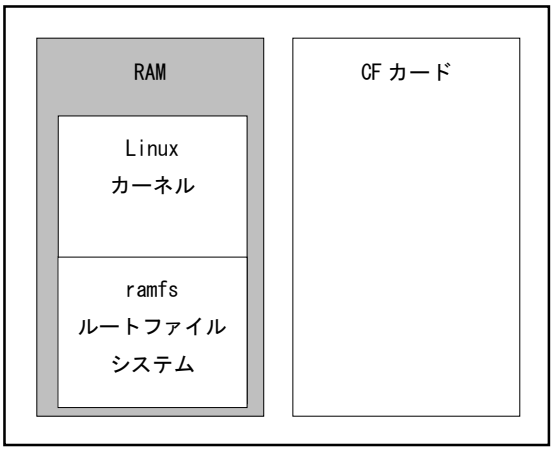


Fig 5.1-1 RAMFS-Linux システム

パッケージ名	機能
BusyBox	コマンドユーティリティ

RAMFS-Linux システムを作成するには大きく分けて

① ramfs ルートファイルシステムの作成
② Linux カーネルイメージの作成

の 2 つの手順があります。
①の ramfs ルートファイルシステムは Buildroot を使用して作成します。ramfs ルートファイルシステムの内容を変更する場合も Buildroot を使用します。
②の Linux カーネルイメージは①で作成された ramfs ルートファイルシステムを Linux カーネルのコンパイル時にリンクして作成します。作成した Linux カーネルイメージが RAMFS-Linux システムとなるため、別途 RAMFS-Linux システムを構築する必要はありません。

5.2 プログラム配置イメージ

RAMFS-Linux システムは、RAM 上に Linux カーネル領域を確保し、Linux カーネル自身がカーネルとルートファイルシステムを展開して動作します。

※1 アドレスは P1 領域アドレスで示します。
※2 RAM に展開される RAMFS-Linux システムのサイズは Linux カーネル・ルートファイルシステムのサイズにより変更します。

FlashROM 16Mbyte

開始アドレス※1	領域名		領域サイズ
0x80000000	U-Boot		512KByte
0x80080000	U-Boot 環境変数領域		128KByte
0x800A0000	RAMFS-Linux システム	Linux カーネル(U-Boot 用イメージ)	3MByte
		ramfs ルートファイルシステム	
		ユーザプログラム	
0x803A0000 0x80FFFFFF	ユーザ使用領域		約 12MByte

RAM に展開

SDRAM 64Mbyte

開始アドレス※1	領域名		領域サイズ
0x84000000	未使用領域		4KByte
0x84001000	RAMFS-Linux システム	Linux カーネル	約 30MByte ※2
		rsmfs ルートファイルシステム	
		ユーザプログラム	
0x86000000 0x87FFFFFF	Linux プロセス使用領域		約 32MByte

Fig 5.2-1 RAMFS-Linux システム配置イメージ

6. CF-Linux システム

本章では CF-Linux システムの作成・使用方法について説明します。

6.1 CF-Linux システムの概要

CF-Linux システムは CF カード上にルートファイルシステムを展開し、動作します。CF-Linux システムはファイルシステムに ext2 を使用しており、Linux カーネルも cf ルートファイルシステムに格納されています。起動時にはブートローダが CF カード上に構築された cf ルートファイルシステムから Linux カーネールを読み出し実行します。
CF-Linux システムはルートファイルシステムが CF カード上に展開されるため、電源を落してもルートファイルシステムの内容は消えず保持されます。ただし、シャットダウン処理を行わなかった場合、ルートファイルシステムが破壊される場合があります。

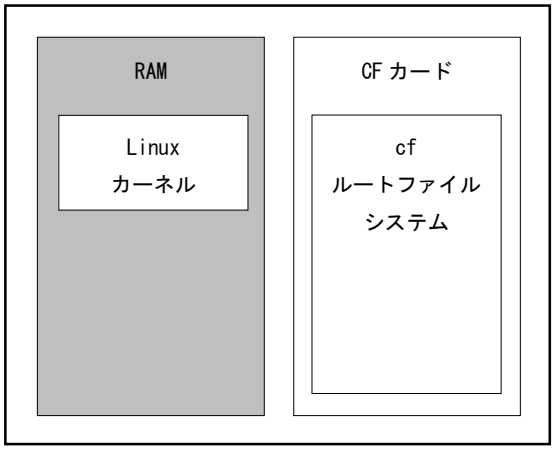


Fig 6.1-1 CF-Linux システム

パッケージ名	機能
BusyBox	コマンドユーティリティ
alsa-lib	ALSA ライブラリ
alsa-utils	ALSA ユーティリティ
DirectFB	グラフィックスライブラリ

CF-Linux システムの作成は大きく分けて

① cf ルートファイルシステムの作成
② Linux カーネルイメージの作成
③ CF-Linux システムの構築

の 3 つの手順があります。
①の cf ルートファイルシステムは Buildroot を使用して作成します。②では Linux カーネルイメージを作成します。③では①、②で作成した cf ルートファイルシステムと Linux カーネルイメージをゲスト OS から NFS 経由でダウンロードし、CF カードに書き込み、CF-Linux システムを構築します。

6.2 プログラム配置イメージ

CF-Linux システムは、CF カード内の Linux カーネルを RAM に展開し、CF カード上のルートファイルシステムにアクセスしながら動作します。

※1 アドレスは P1 領域アドレスで示します。

CF カード 1GByte

領域名		領域サイズ
CF-Linux システム	Linux カーネル(U-Boot 用イメージ)	約 8.5MByte
	cf ルートファイルシステム	
	ユーザプログラム	
	ユーザ使用領域	約 1GByte

RAM に展開

SDRAM 64Mbyte

開始アドレス※1	領域名	領域サイズ
0x84000000	未使用領域	4KByte
0x84001000	Linux カーネル	約 4MByte
0x84400000 0x87FFFFFF	Linux プロセス使用領域	約 60MByte

CF カードにアクセス

CF カード 1Gbyte

領域名		領域サイズ
CF-Linux システム	Linux カーネル	約 8.5MByte
	cf ルートファイルシステム	
	ユーザプログラム	
	ユーザ使用領域	約 1GByte

Fig 6.2-1 CF-Linux システム配置イメージ

7. プログラムの作成

本章では、MS104-SH4AG 上の任意のアドレスにアクセス可能な汎用デバイスドライバの作成方法とそのデバイスドライバを使用して LED と I/O ポートの制御を行えるアプリケーションの作成方法について説明します。

7.1 プログラムの開発について

ソースファイルのコンパイルから動作までの一連の流れを示します。

① ゲスト OS 上でソースファイルを作成。
② ゲスト OS 上でソースファイルをクロスコンパイルし、実行ファイルを作成。
③ MS104-SH4AG ボード上でゲスト OS を nfs でマウントし、実行ファイルをダウンロード。
④ MS104-SH4AG ボード上で動作を確認。

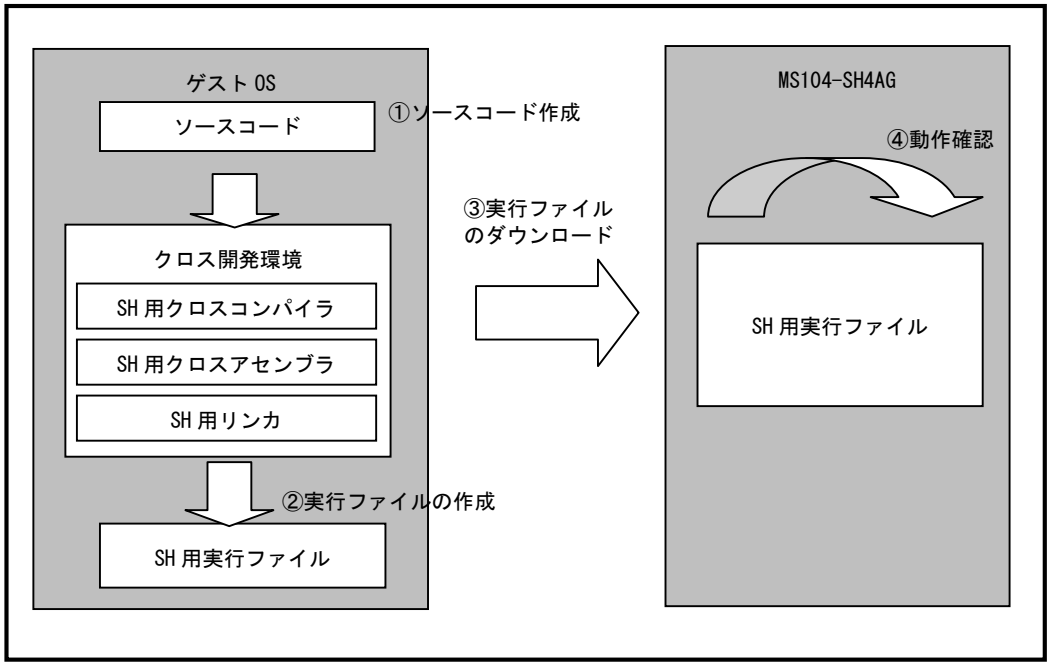


Fig 7.1-1 プログラムの開発手順

7.2 汎用デバイスドライバの概要

汎用デバイスドライバはデバイスへのアクセス関数を提供します。

汎用デバイスドライバの概要

ユーザプログラム上からデバイスにアクセスする際、通常はデバイスファイルを通じてシステムコールを発行し、デバイスドライバに処理を依頼します。汎用デバイスドライバはデバイスへのアクセス関数を提供することにより、ユーザプログラム上からデバイスにアクセスする手段を提供します。
汎用デバイスドライバはキャラクタ型デバイスドライバになり、モジュールとしてコンパイルします。
ユーザプログラム上からデバイスに『**iorw-ms104sh4ag.c**』が提供するシステムコール（API）は『**open**』、『**close**』、『**ioctl**』、になります。汎用デバイスドライバを示すデバイスファイルは『**/dev/iorw0**』になります。

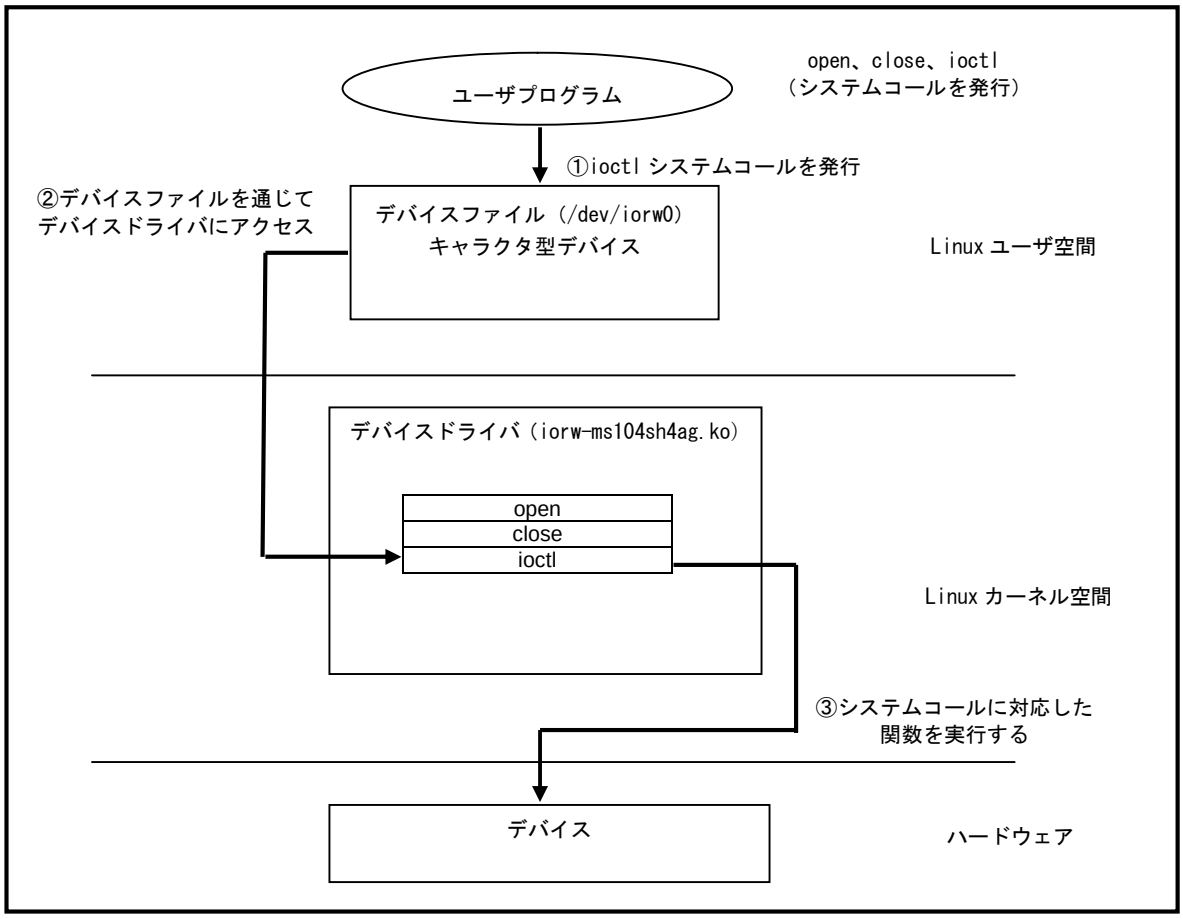


Fig 7.2-1 汎用デバイスドライバの概要

システムコール

汎用デバイスドライバは『**ioctl**』システムコールを介して渡されたアドレスを元に、デバイスへのリードおよびライトを実行します。アクセス幅の指定は各 ioctl コマンド毎に割り当てられています。
汎用デバイスドライバの各システムコールについて下記に示します。各システムコールの書式は Linux の標準 API に従います。

● 汎用デバイスドライバ用構造体

構造体名	ms104sh4ag_iorw

書式	struct ms104sh4ag_iorw {
	unsigned long addr;
	unsigned long data;
	};

メンバ	addr : アドレス data : データ
備考	汎用デバイスドライバ用構造体はアドレス『**addr**』とデータ『**data**』をメンバとして持ち、リード時は指定したアドレス『**addr**』の値をデータ『**data**』として取得します。 ライト時は指定したアドレス『**addr**』にデータ『**data**』を書き込みを行います。

● open システムコール

機能	デバイスをオープンする
書式	int open(char* devicename, int flags)
引数	devicename : 論理デバイス名 flags : フラグ
戻り値	ファイルディスクプリタを返す エラー時は-1 を返す
備考	論理デバイス名『**/dev/iorw0**』を使用 フラグは O_RDWR を使用

● close システムコール

機能	デバイスをクローズする
書式	int close(int fd)
引数	fd : ファイルディスクプリタ
戻り値	クローズ成功時には 0、エラー時は-1 を返す
備考	

● ioctl システムコール（MS104SH4AG_IOC_IOR8）

機能	8 ビットリードアクセスを実行する。
書式	int ioctl(int fd, MS104SH4AG_IOC_IOR8, int *arg)
引数	fd : ファイルディスクプリタ arg : 汎用デバイスドライバ用構造体ポインタ
戻り値	成功時には 0、エラー時は-1 を返す
備考	汎用デバイスドライバ用構造体はアドレス『**addr**』を指定します。

● ioctl システムコール（MS104SH4AG_IOC_IOW8）

機能	8 ビットライトアクセスを実行する。
書式	int ioctl(int fd, MS104SH4AG_IOC_IOW8, int *arg)
引数	fd : ファイルディスクプリタ arg : 汎用デバイスドライバ用構造体ポインタ
戻り値	成功時には 0、エラー時は-1 を返す
備考	汎用デバイスドライバ用構造体はアドレス『**addr**』、データ『**data**』を指定します。

● ioctl システムコール（MS104SH4AG_IOC_IOR16）

機能	16 ビットリードアクセスを実行する。
書式	int ioctl(int fd, MS104SH4AG_IOC_IOR16, int *arg)
引数	fd : ファイルディスクプリタ arg : 汎用デバイスドライバ用構造体ポインタ
戻り値	成功時には 0、エラー時は-1 を返す
備考	汎用デバイスドライバ用構造体はアドレス『**addr**』を指定します。

● ioctl システムコール（MS104SH4AG_IOC_IOW16）

機能	16 ビットライトアクセスを実行する。
書式	int ioctl(int fd, MS104SH4AG_IOC_IOW16, int *arg)
引数	fd : ファイルディスクプリタ arg : 汎用デバイスドライバ用構造体ポインタ
戻り値	成功時には 0、エラー時は-1 を返す
備考	汎用デバイスドライバ用構造体はアドレス『**addr**』、データ『**data**』を指定します。

● ioctl システムコール（MS104SH4AG_IOC_IOR32）

機能	32 ビットリードアクセスを実行する。
書式	int ioctl(int fd, MS104SH4AG_IOC_IOR16, int *arg)
引数	fd : ファイルディスクプリタ arg : 汎用デバイスドライバ用構造体ポインタ
戻り値	成功時には 0、エラー時は-1 を返す
備考	汎用デバイスドライバ用構造体はアドレス『**addr**』を指定します。

● ioctl システムコール（MS104SH4AG_IOC_IOW32）

機能	32 ビットライトアクセスを実行する。
書式	int ioctl(int fd, MS104SH4AG_IOC_IOW32, int *arg)
引数	fd : ファイルディスクプリタ arg : 汎用デバイスドライバ用構造体ポインタ
戻り値	成功時には 0、エラー時は-1 を返す
備考	汎用デバイスドライバ用構造体はアドレス『**addr**』、データ『**data**』を指定します。

8. 製品サポートのご案内

●ユーザ登録

ユーザ登録は弊社ホームページにて受け付けております。ユーザ登録をしていただきますと、ユーザ専用ページにアクセスすることができます。ユーザ専用ページでは、最新版のマニュアルやソフトウェア、またアプリケーションノート等、お客様にお役立ていただける情報を掲載しておりますので是非ご利用ください。

弊社ホームページアドレス　http://www.apnet.co.jp

●ソフトウェアのサポート

ソフトウェアに関する技術的な質問は、受け付けておりませんのでご了承ください。
サポートをご希望されるお客様には、別途有償サポートプログラムをご用意しておりますので、弊社営業までご連絡ください。

●バージョンアップ

本製品に付属するソフトウェアは、不定期で更新されます。それらは全て弊社ホームページよりダウンロードできます。
CD-ROM、DVD-ROM などの物理媒体での提供をご希望される場合には、実費にて承りますので弊社営業までご連絡ください。

●修理の依頼

修理をご依頼いただく場合には、お名前、製品名、シリアル番号、詳しい故障状況を弊社製品サポートへご連絡ください。
弊社にて故障状況を確認のうえ、修理の可否、修理費用等をご連絡いたします。ただし、過電圧印加や高熱等により製品全体がダメージを受けていると判断される場合には、修理をお断りする場合もございますのでご了承ください。
なお、弊社までの送料はお客様ご負担となります。

製品修理窓口

■ＦＡＸ	０５３－４０１－００３５
■Ｅ－ＭＡＩＬ	repair@apnet.co.jp

●製品サポートの方法

製品サポートについては、FAX もしくは E-MAIL でのみ受け付けております。お電話でのお問い合わせは受け付けておりませんのでご了承ください。なお、お問い合わせの際には、製品名、使用環境、使用方法等、問題点などを詳細に記載してください。

製品サポート窓口

■ＦＡＸ	０５３－４０１－００３５
■Ｅ－ＭＡＩＬ	query@apnet.co.jp

エンジニアリングサービスのご案内

弊社製品をベースとしたカスタム品やシステム開発を承っております。
お客様の仕様に合わせて、設計から OEM 供給まで一貫したサービスを提供いたします。
詳しくは、弊社営業窓口までお問い合わせください。

営業案内窓口

■ＴＥＬ　　０５３－４０１－００３３（代表）
■Ｅ－ＭＡＩＬ　　sales@apnet.co.jp

改定履歴

版数	日付	改定内容
1 版	2008/12/10	新規作成

参考文献

SH-Linux については以下の URL を参考にしてください。

・ SuperH Linux Open site

http://www.superh-linux.org/index.html

謝辞

Linux、SH-Linux、U-Boot の開発に関わった多くの貢献者に深い敬意と感謝の意を示します。

著作権について

商標について

ALPHAPROJECT Co.,LTD.
株式会社アルファプロジェクト
〒433-3114
静岡県浜松市東区積志町８３４
http://www.apnet.co.jp
E-MAIL : sales@apnet.co.jp