MS104-SH4 ソフトウェアマニュアル

Software Manual

1 版

ダイジェスト版

ALPHA PROJECT

http://www.apnet.co.jp

ご使用になる前に

このたびは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頂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。

参考ＵＲＬ

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

目　次

1.　はじめに　1

1.1　Linux について 1
1.2　U-Boot について 1
1.3　VMware Player について 1
1.4　GNU と FSF について 2
1.5　GPL と LGPL について 2
1.6　保証とサポート 2

2.　クロス環境開発　3

2.1　クロス開発環境概要 3
2.2　動作環境 4
2.3　添付 DVD の構成 5
2.4　クロス開発環境のインストール 6

3.　Linux の起動　8

3.1　PC 動作環境 8
3.2　シリアル設定 9
3.3　ネットワーク設定 9
3.4　MS104-SH4 ボードの接続 10
3.5　ターミナルの設定 11

4.　システムの起動と停止　13

4.1　起動 13
4.2　ログイン 16
4.3　停止 16
4.4　U-Boot コマンドラインへの移行 17

5.　ネットワーク　18

5.1　ネットワークの設定 18
5.2　設定の変更 19

6. CF カード 20

6.1 CF カードのマウント 20
6.2 CF カードの活線挿抜 21

7. RTC 22

7.1 時刻の読み出し 22
7.2 時刻の設定 22

8. LED 23

8.1 LED の点灯・消灯 23

9. フラッシュ ROM の書き換え 24

9.1 メモリマップ 24
9.2 Linux の更新 24
9.3 U-Boot の更新 25
9.4 出荷時状態へのリセット 26

10. システムの構築 28

10.1 U-Boot のコンパイル 28
10.2 ルートファイルシステムの作成 29
10.3 Linux カーネルのコンパイル 31

11. RAMFS-Linux システム 32

11.1 RAMFS-Linux システムの概要 32
11.2 プログラム配置イメージ 33
11.3 ramfs ルートファイルシステムの作成 34
11.4 Linux カーネルの作成 36
11.5 RAMFS-Linux システムの起動 42

12. CF-Linux システム 43

12.1 CF-Linux システムの概要 43
12.2 プログラム配置イメージ 44
12.3 cf ルートファイルシステムの作成 45
12.4 Linux カーネルの作成 47
12.5 CF-Linux システムの構築 49
12.6 CF-Linux システムの起動 51

13. フラッシュ ROM ブート 52

13.1 オートブート 52
13.2 マニュアルブート 53
13.3 オートブート設定 54

14. ネットワークブート 58

14.1 TFTP 58

15. ストレージブート 59

15.1 CF カード(ramfs ルートファイルシステム) 59
15.2 CF カード(cf ルートファイルシステム) 61

16. プログラムの作成 63

16.1 プログラムの開発について 63
16.2 汎用デバイスドライバの概要 64
16.3 汎用デバイスドライバのコンパイル 67
16.4 サンプルアプリケーションのコンパイル 73

17. U-Boot コマンドリファレンス 77

17.1 U-Boot のコマンド 77

18. 製品サポートのご案内 89

19. エンジニアリングサービスのご案内 90

1. はじめに

MS104-SH4 は、SH7750R 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-SH4 のソフトウェア開発が可能です。

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。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 (Fedora) 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、改変したりした著作物)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 及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行い、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. クロス環境開発

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。

2.1 クロス開発環境概要

CPU ボード上で動作する 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『Fedora』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。

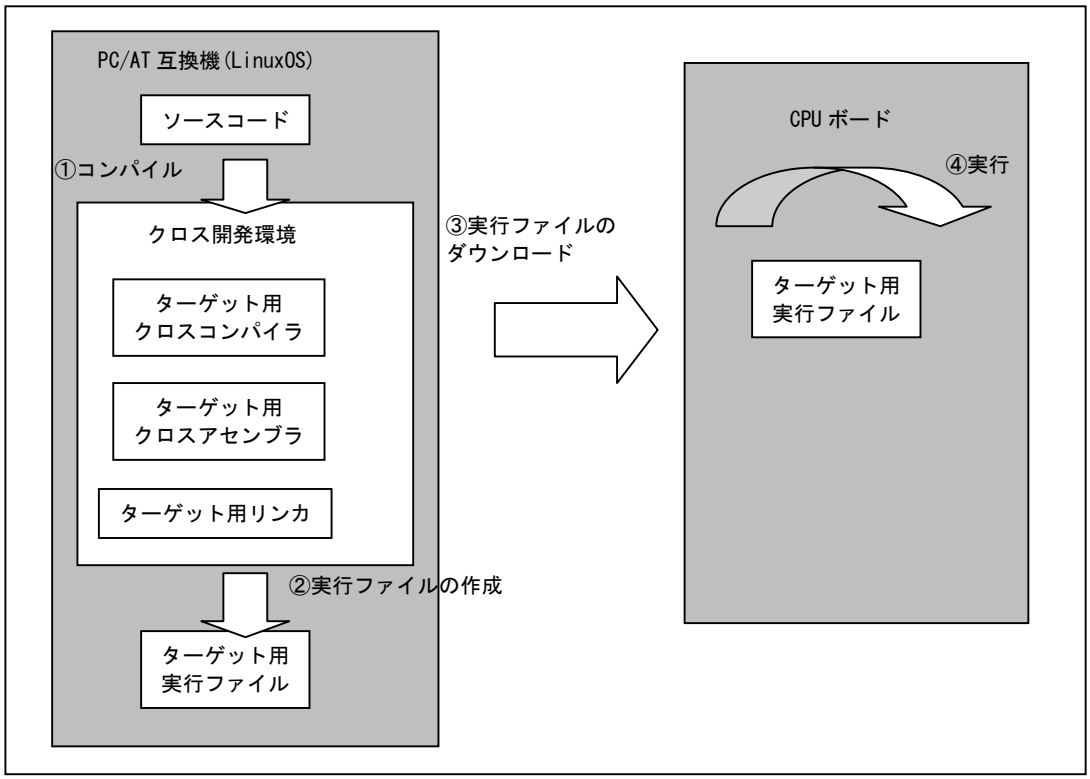


Fig 2.1-1 クロス開発環境

2.2 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、Windows/XP もしくは Windows/Vista 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』をご覧下さい。
また、VMware Player 及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 Windows/XP)
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 2.2-1 クロス開発環境の動作環境

2.3 添付 DVD の構成

MS104-SH4 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_A03_VX_X
|-- FlashWriter
|   `-- setup.exe
|-- applicationnote                    :アプリケーションノート
|   |-- an1401.pdf
|   |-- an333.pdf
|   |-- an334.pdf
|   `-- an335.pdf
|-- binaries
|   |-- giorw-sample.ko                :汎用デバイスドライバ
|   |-- ms104-640.bmp                  :サンプルビットマップ画像 w:640px
|   |-- ms104.bmp                      :サンプルビットマップ画像 w:800px
|   |-- ms104fpga-sample               :MS104-FPGA/CⅢボードサンプルプログラム
|   |-- ms104sh4-sample                :MS104-SH4 LED 制御サンプルプログラム
|   |-- rootfs.sh4-ms104sh4-vga.cpio.gz :ルートファイルシステム MS104-VGA/LCD cpio フォーマット
|   |-- rootfs.sh4-ms104sh4-vga.tar.gz :ルートファイルシステム MS104-VAG/LCD tar フォーマット
|   |-- rootfs.sh4-ms104sh4.cpio.gz    :ルートファイルシステム cpio フォーマット
|   |-- rootfs.sh4-ms104sh4.tar.gz     :ルートファイルシステム tar フォーマット
|   |-- u-boot-ms104sh4-vga.bin        :U-Boot バイナリイメージ(MS104-VGA/LCD 対応版)
|   |-- u-boot-ms104sh4.bin            :U-Boot バイナリイメージ
|   |-- uImage-ms104sh4                :Linux カーネルイメージ
|   |-- uImage-ms104sh4-fpga           :Linux カーネルイメージ (MS104-FPGA/CⅢ対応版)
|   |-- uImage-ms104sh4-ramfs          :Linux カーネルイメージ (ramfs 付き)
|   |-- uImage-ms104sh4-usb            :Linux カーネルイメージ (MS104-USB H/S 対応版)
|   `-- uImage-ms104sh4-vga            :Linux カーネルイメージ (MS104-VGA/LCD 対応版)
|-- 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
|   |-- hj-link.pdf                    :HJ-LINK ハードウェアマニュアル
|   |-- linuxkit-ms104sh4.pdf          :Linux 開発キット MS104-SH4 ソフトウェアマニュアル
|   `-- linuxkit-vmware.pdf            :Linux 開発キット共通マニュアル VMware Player 編
|-- sources
|   |-- buildroot-2009.02-alp.tar.gz   :BuildRoot ソースコード
|   |-- dl.tar                         :BuildRoot ダウンロードソース
|   |-- install-ms104sh4.sh            :クロス開発環境インストールシェルスクリプト
|   |-- linux-2.6.28.8-alp.tar.gz      :Linux カーネルソースコード
|   |-- ms104fpga-sample.tar.gz        :MS104-FPGA/CⅢボードサンプルソースコード
|   |-- ms104sh4-sample.tar.gz         :MS104-SH4 LED 制御サンプルソースコード
|   |-- ms104vga-sample.tar.gz         :MS104-VGA/LCD ボードサンプルソースコード
|   |-- staging_dir.tar.gz             :クロスコンパイラ
|   |-- u-boot-2009.03-alp.tar.gz      :U-Boot ソースコード
|   |-- u-boot-tools.tar.gz            :U-Boot 付属ツール
|   `-- uninstall-ms104sh4.sh          :クロス開発環境アンインストールシェルスクリプト
`-- 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-2.5.2-156735.exe :VMware Player インストーラ
```

『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。

3. Linux の起動

3.1 PC 動作環境

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

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-SH4 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-SH4 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
PC	PC/AT 互換機
OS	Windows2000/XP/Vista(推奨 Windows/XP)
空き容量	5GByte 以上(推奨 16GByte)
メモリ	512MByte 以上
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
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 3.1-1 動作環境

3.2 シリアル設定

以下に MS104-SH4 のシリアル通信の初期設定を記します。

項目	値
通信速度	115200
データ長	8
ストップビット	1
パリティ	なし
フロー制御	なし

Table 3.2-1 シリアル通信設定例

3.3 ネットワーク設定

MS104-SH4 及びホスト PC のデフォルトネットワーク設定は以下になります。

項目	値
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	設定なし

Table 3.3-1 MS104-SH4 デフォルトネットワーク設定

項目	値
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 3.3-2 ホスト PC デフォルトネットワーク設定

3.4 MS104-SH4 ボードの接続

ホスト PC と MS104-SH4 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-SH4 の J4 コネクタに接続してください。

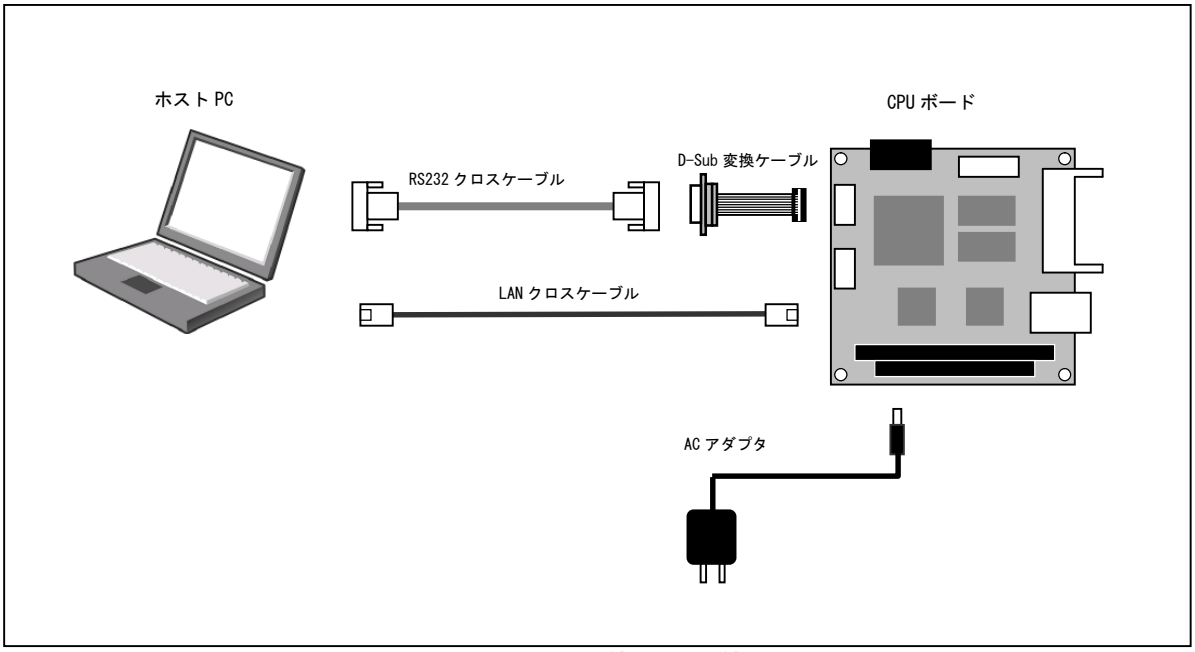


Fig 3.4-1 MS104-SH4 の接続(PC に接続する場合)

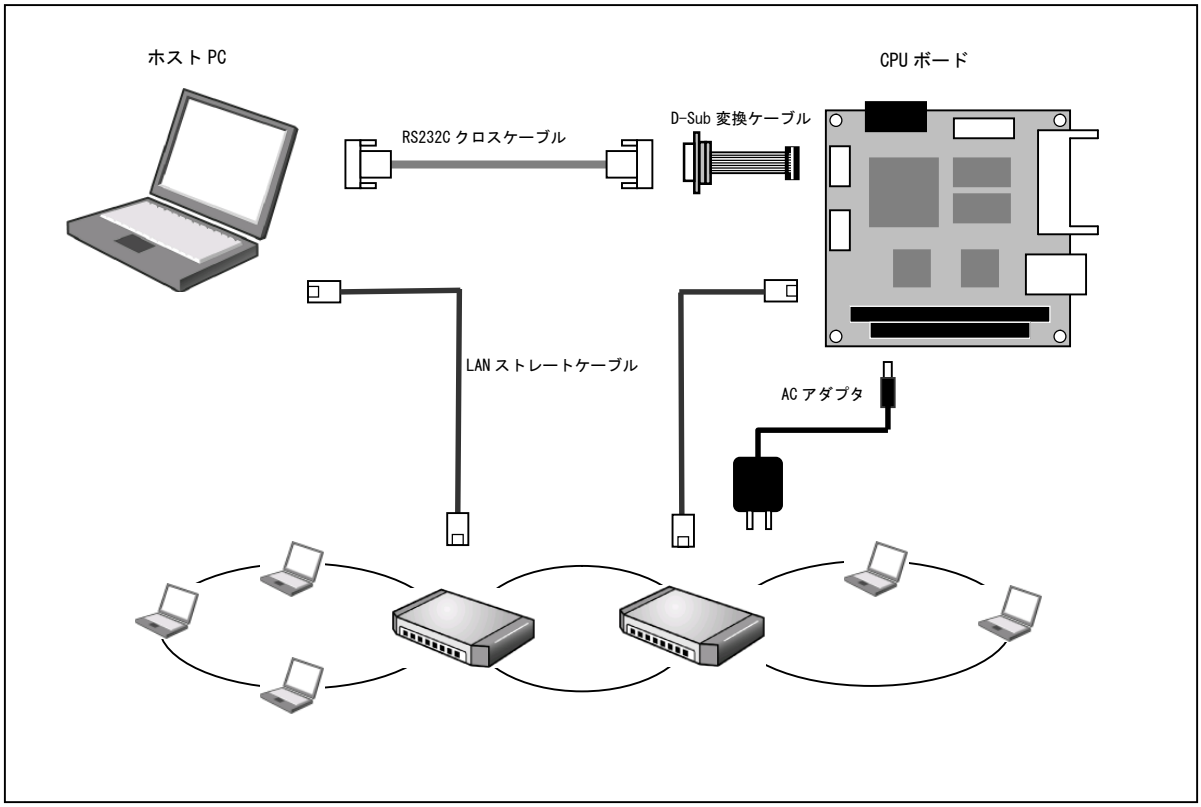


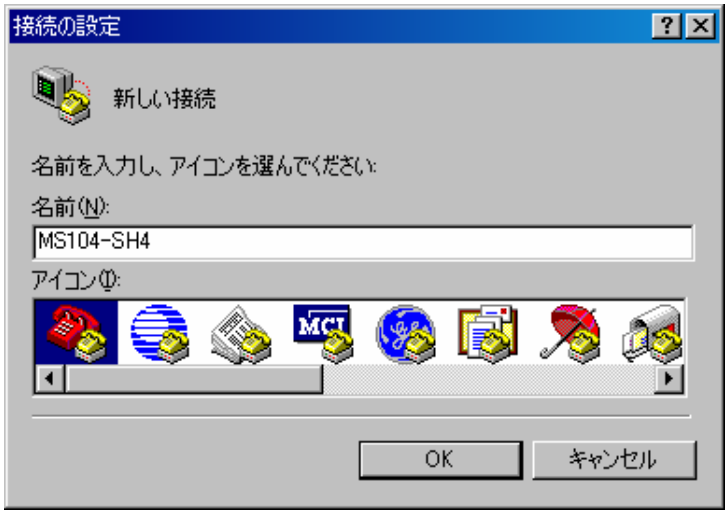
Fig 3.4-2 MS104-SH4 の接続（ハブに接続する場合）

3.5　ターミナルの設定

MS104-SH4 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。

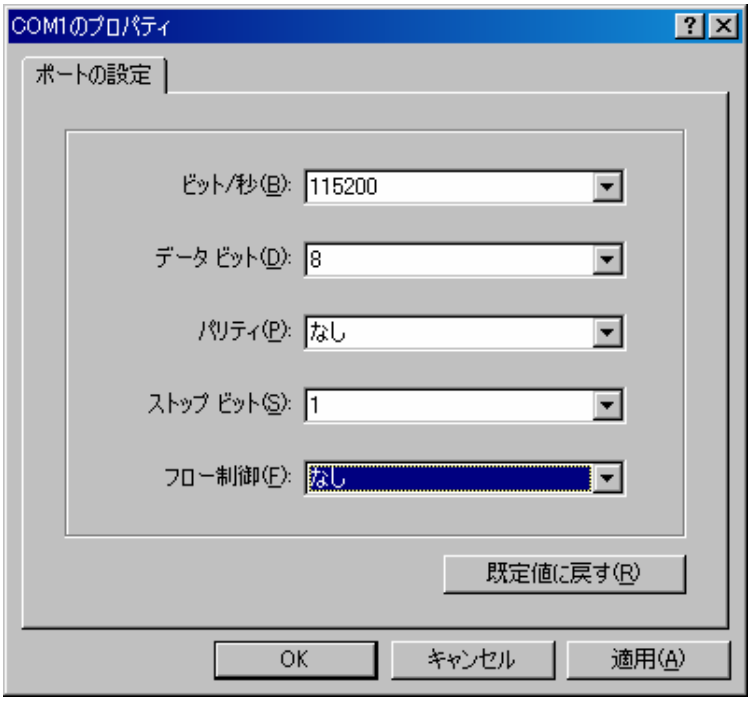
① Windows のスタートメニューから、『プログラム』-『アクセサリ』-『通信』-『ハイパーターミナル』を選択し、ハイパーターミナルを起動します。

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。

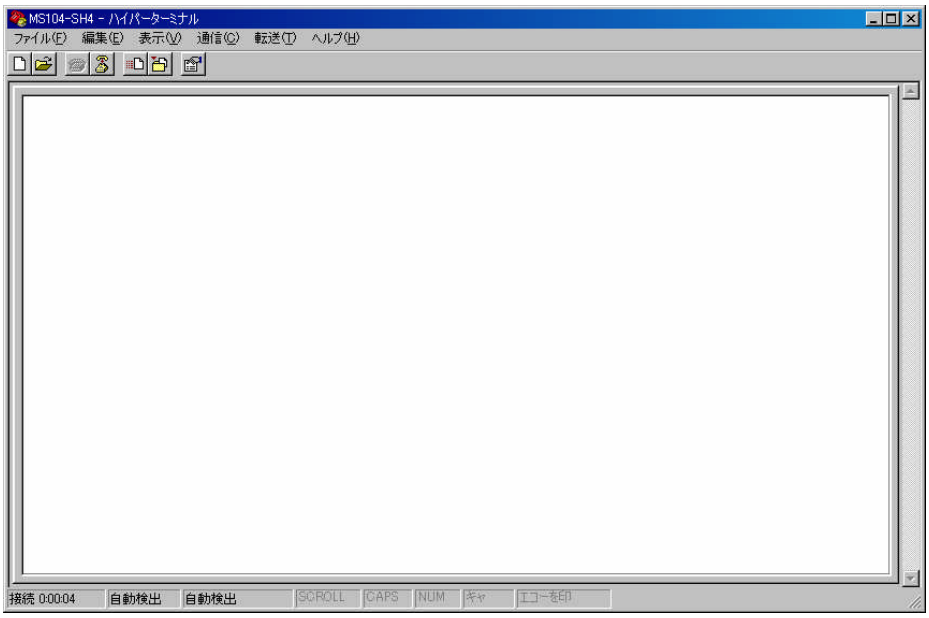
④　『ポートの設定』の設定を行い、『OK』ボタンを押します。



ポートの設定	値
ビット/秒	115200
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

Table 3.5-1　ポートの設定

⑤　ハイパーターミナルが起動します。



4. システムの起動と停止

4.1 起動

MS104-SH4 の電源を投入します。U-Boot 及び Linux が起動しシリアルコンソールに以下のようなメッセージが出力されます。ログインプロンプトが表示されるまでに 15 秒ほどかかります。

```
U-Boot 2009.03 ( 4 月 02 2009 - 15:42:32)

CPU: SH4
BOARD: SH7750R ALPHAPROJECT MS104-SH4
DRAM:  32MB
FLASH: 16MB
In:    serial
Out:   serial
Err:   serial
Net:   Hit any key to stop autoboot:  0
Instruction Cache is ON
## Booting kernel from Legacy Image at a0100000 ...
   Image Name:   Linux-2.6.28.8
   Created:      2009-04-03   2:38:55 UTC
   Image Type:   SuperH Linux Kernel Image (gzip compressed)
   Data Size:    2213760 Bytes =  2.1 MB
   Load Address: 8c001000
   Entry Point:  8c002000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
Linux version 2.6.28.8 (guest@fedora10) (gcc version 4.3.2 (GCC) ) #35 Fri Apr 3 11:38:54 JST 2009
Boot params:
... MOUNT_ROOT_RDONLY - 00000000
... RAMDISK_FLAGS     - 00000000
... ORIG_ROOT_DEV     - 00000000
... LOADER_TYPE       - 00000000
... INITRD_START      - 00000000
... INITRD_SIZE       - 00000000
Booting machvec: MS104-SH4
ms104sh4_setup: cmdline=console=ttySC1,115200
Node 0: start_pfn = 0xc000, low = 0xe000
Zone PFN ranges:
  Normal   0x0000c000 -> 0x0000e000
Movable zone start PFN for each node
early_node_map[1] active PFN ranges
    0: 0x0000c000 -> 0x0000e000
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 8128
Kernel command line: console=ttySC1,115200
MS104-SH4 IRQ setup
PID hash table entries: 128 (order: 7, 512 bytes)
Using tmu for system timer
```

```
Using 14.746 MHz high precision timer.
Console: colour dummy device 80x25
Dentry cache hash table entries: 4096 (order: 2, 16384 bytes)
Inode-cache hash table entries: 2048 (order: 1, 8192 bytes)
Memory: 28580k/32768k available (1583k kernel code, 469k data, 1004k init)
PVR=04050045 CVR=20480000 PRR=00000103
I-cache : n_ways=2 n_sets=256 way_incr=8192
I-cache : entry_mask=0x00001fe0 alias_mask=0x00001000 n_aliases=2
D-cache : n_ways=2 n_sets=512 way_incr=16384
D-cache : entry_mask=0x00003fe0 alias_mask=0x00003000 n_aliases=4
SLUB: Genslabs=10, HWalign=32, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
Calibrating delay loop (skipped)... 235.92 BogoMIPS PRESET (lpj=471859)
Mount-cache hash table entries: 512
CPU: SH7750R
net_namespace: 288 bytes
NET: Registered protocol family 16
SCSI subsystem initialized
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 1024 (order: 1, 8192 bytes)
TCP bind hash table entries: 1024 (order: 0, 4096 bytes)
TCP: Hash tables configured (established 1024 bind 1024)
TCP reno registered
NET: Registered protocol family 1
io scheduler noop registered (default)
SuperH SCI(F) driver initialized
sh-sci: ttySC0 at MMIO 0xffe00000 (irq = 25) is a sci
sh-sci: ttySC1 at MMIO 0xffe80000 (irq = 43) is a scif
console [ttySC1] enabled
smc91x.c: v1.1, sep 22 2004 by Nicolas Pitre <nico@cam.org>
eth0: SMC91C11xFD (rev 1) at a8000300 IRQ 8
eth0: Ethernet addr: 00:0c:7b:20:00:11
Driver 'sd' needs updating - please use bus_type methods
physmap platform flash device: 01000000 at 00000000
physmap-flash: Found 1 x16 devices at 0x0 in 16-bit bank
 Intel/Sharp Extended Query Table at 0x0031
Using buffer write method
cfi_cmdset_0001: Erase suspend on write enabled
cmdlinepart partition parsing not available
RedBoot partition parsing not available
Using physmap partition information
Creating 4 MTD partitions on "physmap-flash":
0x00000000-0x000e0000 : "U-Boot"
mtd: Giving out device 0 to U-Boot
0x000e0000-0x00100000 : "U-Boot env"
mtd: Giving out device 1 to U-Boot env
0x00100000-0x00400000 : "Kernel"
mtd: Giving out device 2 to Kernel
0x00400000-0x01000000 : "Userland"
mtd: Giving out device 3 to Userland
ms104sh4_cf ms104sh4_cf: MS104-SH4 CompactFlash Socket Driver
```

```
rs5c316 rs5c316: rtc core: registered rs5c316 as rtc0
Registered led device: led3
Registered led device: led2
Registered led device: led1
TCP cubic registered
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
rs5c316 rs5c316: setting system clock to 2009-04-09 09:08:07 UTC (1239268087)
Freeing unused kernel memory: 1004k fre?pcmcia_socket pcmcia_socket0: pccard: PCMCIA card inserted into
slot 0
pcmcia 0.0: pcmcia: registering new device pcmcia0.0
scsi0 : pata_pcmcia
ata1: PATA max PIO0 cmd 0xc0000000 ctl 0xc000000e irq 12
ata1.00: CFA: , 20051220, max MWDMA4
ata1.00: 4061232 sectors, multi 0: LBA
ata1.00: configured for PIO0
ata1.00: configured for PIO0
ata1: EH complete
scsi 0:0:0:0: Direct-Access     ATA                          2005 PQ: 0 ANSI: 5
sd 0:0:0:0: [sda] 4061232 512-byte hardware sectors: (2.07 GB/1.93 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
sd 0:0:0:0: [sda] 4061232 512-byte hardware sectors: (2.07 GB/1.93 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
 sda: sda1
sd 0:0:0:0: [sda] Attached SCSI removable disk
sd 0:0:0:0: Attached scsi generic sg0 type 0
Initializing random number generator... done.
Starting network...
Starting vsftpd: OK



Welcome to Buildroot
uclibc login:
```

4.2 ログイン

Linux の起動が完了するとログインプロンプトが表示されます。出荷時状態ではパスワードは設定されていないため、以下に示すユーザ名を入力することによりログインすることができます。

ユーザ名	権限
root	特権ユーザ
default	一般ユーザ

Table 4.2-1 ユーザ名と権限

4.3 停止

システムを停止するときは halt コマンドを実行します。『System halted.』のメッセージが表示されたことを確認し電源を落とします。

CF カード等の外部デバイス上のファイルシステムを使用しているときは、停止用のコマンドを使用することにより安全にシステムを停止させることができます。

```
# halt
The system is going down NOW!
Sending SIGTERM to all processes
Sending SIGKILL to all processes
Requesting system halt
sd 0:0:0:0: [sda] Stopping disk
System halted.
```

再起動をするときは reboot コマンドを実行します。

```
# reboot
The system is going down NOW!
Sending SIGTERM to all processes
Sending SIGKILL to all processes
Requesting system reboot
Restarting?

U-Boot 2009.03 ( 3 月 04 2009 - 12:14:37)
... 中略 ...
```

4.4 U-Boot コマンドラインへの移行

電源投入直後、『Hit any key to stop autoboot』の文字が表示され、2 秒以内にキー入力を行うと U-Boot のコマンドコンソールが起動します。コマンドコンソールが起動すると、『=>』が表示されコマンドライン上での作業が可能となります。

```
U-Boot 2009.03 ( 3 月 16 2009 - 12:02:06)

CPU: SH4
BOARD: SH7750R ALPHAPROJECT MS104-SH4
DRAM:  32MB
FLASH: 16MB
In:    serial
Out:   serial
Err:   serial
Net:   Hit any key to stop autoboot:  0  <-- 何かキーを入力する
=>
```

U-Boot 上での作業を行うときには電源投入直後に U-Boot のコマンドライン環境へ移行する必要があります。出荷時の設定では自動的に Linux が起動するため上記の方法で停止してください。

5. ネットワーク

Linux のネットワーク設定を変更する方法について説明します。

5.1 ネットワークの設定

5.1.1 IP アドレスの設定

IP アドレス、サブネットマスク及びゲートウェイを **Table 5.1-1** のように変更するには、/etc/network/interfaces を以下のように編集します。

項目	値
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 1	192.168.128.1
DNS サーバ 2	192.168.128.2

Table 5.1-1 IP アドレスの設定

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

```
# Configure Loopback
auto lo eth0
iface lo inet loopback
iface eth0 inet static
address 192.168.128.200
netmask 255.255.255.0
gateway 192.168.128.254
```

5.1.2 DSN サーバの設定

DNS サーバの設定を変更するには/etc/resolv.conf を以下のように編集します。

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

```
nameserver 192.168.128.1
nameserver 192.168.128.2
```

5.2 設定の変更

設定を反映させます。

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

ifconfig コマンドで設定を確認します。

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:7B:20:xx:xx
          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:14223 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1564062 (1.4 MiB)  TX bytes:1356 (1.3 KiB)
          Interrupt:8 Base address:0x300 DMA chan:ff

#
```

接続されているネットワーク機器と ping コマンドで通信を確認します。

```
# ping -c 3 192.168.128.253
PING 192.168.128.253 (192.168.128.253): 56 data bytes
64 bytes from 192.168.128.253: seq=0 ttl=64 time=1.780 ms
64 bytes from 192.168.128.253: seq=1 ttl=64 time=0.910 ms
64 bytes from 192.168.128.253: seq=2 ttl=64 time=0.902 ms

--- 192.168.128.253 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.902/1.197/1.780 ms
#
```

6. CF カード

6.1 CF カードのマウント

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可能です。

活線挿抜はマウントを解除した状態で行ってください。

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/sda1 /mnt/cf/
#
```

ls コマンドで内容を確認します。

```
# ls /mnt/cf/
a.txt
#
```

umount コマンドで CF カードをアンマウント(マウント解除)します。

```
# umount /mnt/cf/
```

6.2 CF カードの活線挿抜

CF カードをマウントしていないときは電源が ON の状態でもカードの抜き差しが可能です。コンソール 64 は CF カードを挿入したときの表示です。コンソール 65 は CF カードを抜き出したときの表示です。

```
# pcmcia_socket pcmcia_socket0: pccard: PCMCIA card inserted into slot 0
pcmcia 0.0: pcmcia: registering new device pcmcia0.0
scsi0 : pata_pcmcia
ata1: PATA max PIO0 cmd 0xc0000000 ctl 0xc000000e irq 12
ata1.00: CFA: FLASH CARD, 20070418, max MWDMA4
ata1.00: 1989792 sectors, multi 0: LBA
ata1.00: configured for PIO0
scsi 0:0:0:0: Direct-Access     ATA      FLASH CARD       2007 PQ: 0 ANSI: 5
sd 0:0:0:0: [sda] 1989792 512-byte hardware sectors: (1.01 GB/971 MiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
sd 0:0:0:0: [sda] 1989792 512-byte hardware sectors: (1.01 GB/971 MiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
 sda: sda1
sd 0:0:0:0: [sda] Attached SCSI removable disk
sd 0:0:0:0: Attached scsi generic sg0 type 0
```

```
# pcmcia_socket pcmcia_socket0: pccard: card ejected from slot 0
ata1.00: disabled
sd 0:0:0:0: [sda] Stopping disk
sd 0:0:0:0: [sda] START_STOP FAILED
sd 0:0:0:0: [sda] Result: hostbyte=0x04 driverbyte=0x00
```

7. RTC

7.1 時刻の読み出し

RTC に設定されている時刻の読み出しには hwclock コマンドを使用します。

```
# hwclock
Mon Mar  2 15:36:10 2009  0.000000 seconds
#
```

7.2 時刻の設定

RTC に設定されている時刻を変更するには date コマンドを使用してシステムの時刻を設定し、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。

```
# date -s '2009-03-02 15:30'
Mon Mar  2 15:30:00 MST 2009
# hwclock -w
#
```

8. LED

8.1 LED の点灯・消灯

Sysfs ファイルシステムを通じて 3 つの LED を制御することができます。

出荷時状態では起動時に LD1 が点灯されています。

LD1 を点灯します。LD2 及び LD3 も同様です。

```
# echo 1 > /sys/class/leds/led1/brightness
#
```

LD1 を消灯します。LD2 及び LD3 も同様です。

```
# echo 0 > /sys/class/leds/led1/brightness
#
```

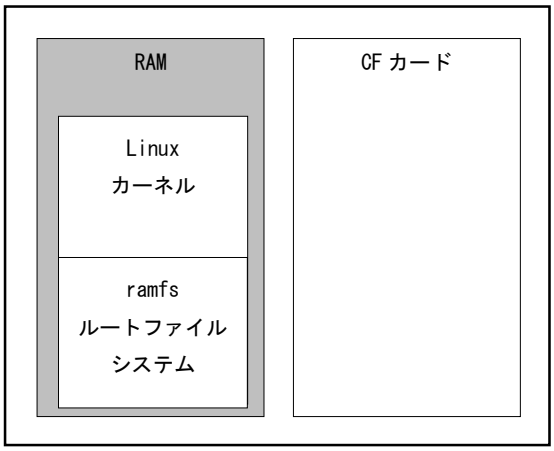
Sysfs は Linux カーネル 2.6 によって提供される仮想ファイルシステムです。Sysfs はデバイスやドライバについての情報をカーネルモデルからユーザ空間へエクスポートし、設定のためにも使われています。BSD システムにある sysctl 機構に似ていますが、分離した機構の代わりにファイルシステムとして実装されています。

11. 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。

11.1 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。



パッケージ名	機能
BusyBOX	コマンドユーティリティ

Fig 11.1-1 RAMFS-Linux システム

RAMFS-Linux システムを作成するには大きく分けて
①ramfs ルートファイルシステムの作成
②Linux 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。

11.2 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
0x8c000000	U-Boot		512KByte
0x8c0e0000	U-Boot 環境変数領域		128KByte
0x8c100000	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	
0x8c400000 0x8cFFFFFF	ユーザ使用領域		12MByte

RAM に展開

SDRAM 32Mbyte

開始アドレス※1	領域名		領域サイズ
0x8c000000	未使用領域		4KByte
0x8c001000	RAMFS-Linux システム	Linux カーネル	約 3MByte ※2
		rsmfs ルートファイルシステム	
		ユーザプログラム	
0x8c400000 0x8dffffff	Linux プロセス使用領域		約 28MByte

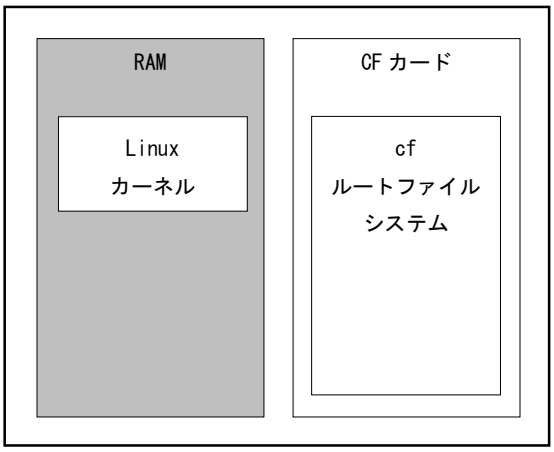
Fig 11.2-1 RAMFS-Linux システム配置イメージ

12. 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。

12.1 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。



パッケージ名	機能
BusyBOX	コマンドユーティリティ

Fig 12.1-1 CF-Linux システム

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。

12.2　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		

RAM に展開

SDRAM　32Mbyte

開始アドレス※1	領域名	領域サイズ
0x8c000000	未使用領域	4KByte
0x8c001000	Linux カーネル	約 4MByte
0x8c400000 0x8dFFFFFF	Linux プロセス使用領域	約 28MByte

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		

Fig 12.2-1　CF-Linux システム配置イメージ

16. プログラムの作成

本章では、MS104-SH4 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。

16.1 プログラムの開発について

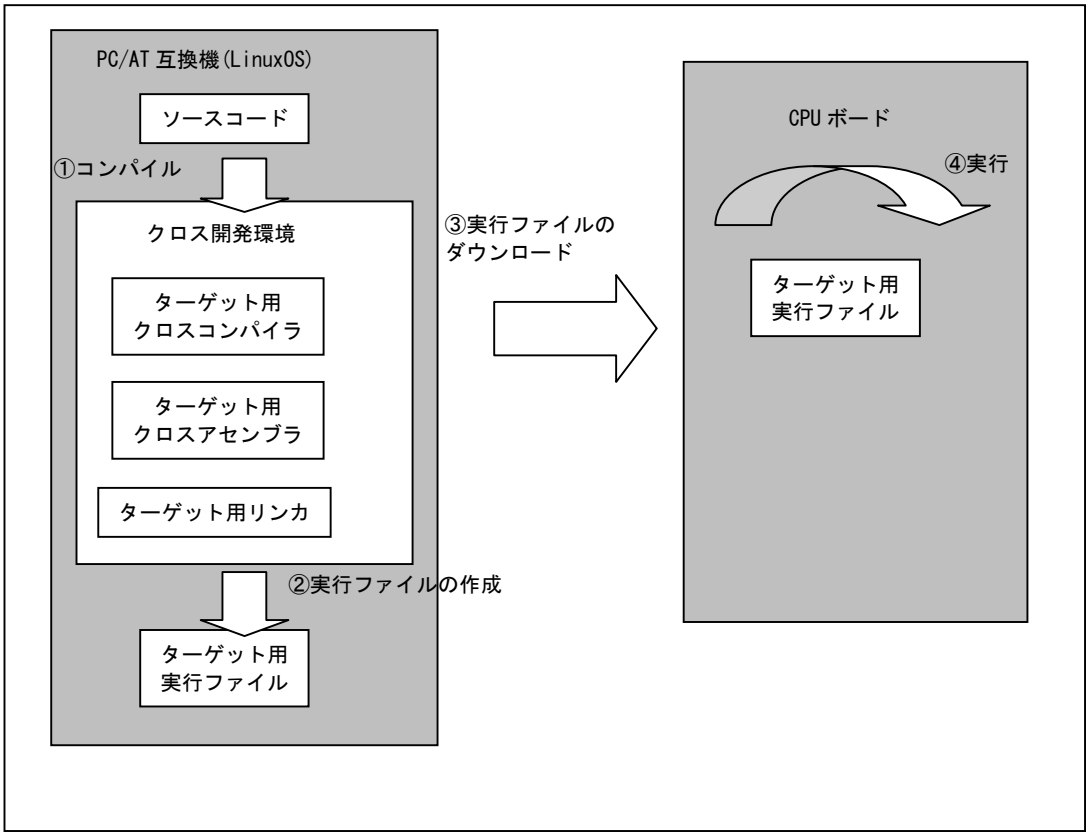
ソースファイルのコンパイルから動作までの一連の流れを示します。

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-SH4 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-SH4 ボード上で動作を確認。

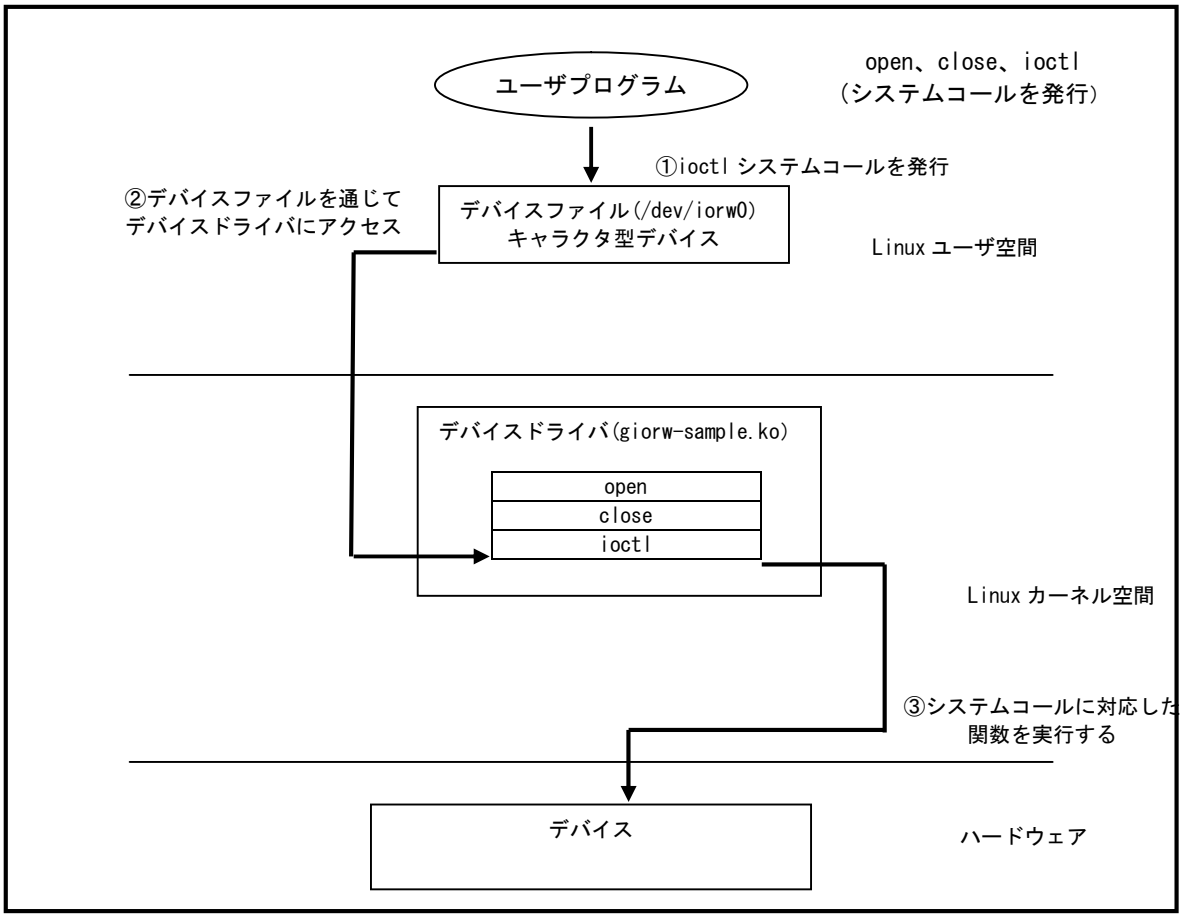


16.2　汎用デバイスドライバの概要

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。

16.2.1　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『**giorw-sample.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。



16.2.2 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及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	giorw

書式	struct giorw {
	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(GIORW_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, GIORW_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(GIORW_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, GIORW_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(GIORW_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, GIORW_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(GIORW_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, GIORW_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。

18. 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、バージョンアップや最新の情報等を E-mail でご案内させていただきまので、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、弊社営業までご相談ください。

●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。

修理・故障に関するお問い合わせ

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。

19. 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）
ＦＡＸ	053-401-0035
Ｅ－ＭＡＩＬ	sales@apnet.co.jp

改定履歴

版数	日付	改定内容
1版	2009/03/18	新規作成

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

ALPHA PROJECT Co.,LTD.
株式会社アルファプロジェクト
〒431-3114
静岡県浜松市東区積志町８３４
http://www.apnet.co.jp
E-MAIL：sales@apnet.co.jp