HT1070
software manual

Version 1.13

2003 年 7 月 28 日

梅沢無線電機株式会社
http://www.umezawa.co.jp/

株式会社アットマークテクノ
http://www.atmark-techno.com/

Armadillo 公式サイト
http://armadillo.atmark-techno.com/

目次

1 Armadillo の使用方法

1.1 オンボード Flash 構成

Armadillo には 4MB の Flash メモリがオンボードで搭載されています。出荷時の Flash 内構成は、以下の通りです。

表 1-1 オンボード Flash 構成

0x00000000	ブートプログラム (Hermit V1.3-armadillo) サイズ: 最大 0x10000 (約 0.06MB)
0x00010000	Linux カーネルイメージ 非圧縮/gzip 圧縮対応 サイズ: 最大 0x170000 (約 1.44MB) 出荷時: 非圧縮カーネル linux-2.4.16-rmk2-armadillo
0x00180000	ユーザランド RAM ディスクイメージ 非圧縮/gzip 圧縮対応 サイズ: 最大 0x280000 (2.5MB)
(0x003fffff)	出荷時: gzip 圧縮 RAM ディスク(6.5MB)イメージ

1.2 起動モード

Armadillo は、JP1/JP2 を設定することにより、起動モードを切り替えることができます。

JP1: オンボード Flash カーネル/Compact Flash カーネル切替
JP2: オンボード Flash 起動/CPU オンチップブート ROM 起動切替

表 1-2 起動モードジャンパ設定

設定			起動モード
JP1	JP2	Compact Flash	
OFF	OFF	―	Linux (オンボード Flash)
ON	OFF	有	Linux (Compact Flash)
		無	オンボード Flash ライタ
―	ON	―	CPU オンチップブート ROM

・JP1:OFF JP2:OFF の場合

オンボード Flash から起動し、オンボード Flash 上の Linux カーネル/RAM ディスクイメージを RAM に展開して、RAM ディスク上のシステムを起動します。

・JP1:ON JP2:OFF の場合

オンボード Flash から起動し、Compact Flash(CF)上の Linux パーティション内に見つかったカーネルを RAM に展開して、カーネルが見つかったパーティション上のシステムを起動します。

CF が挿入されていない場合や、CF 上に Linux パーティションが見つからなかった場合、パーティション内に Linux カーネルが見つからなかった場合は、オンボード Flash ライタ機能などを利用するためのターミナルが動作します。

・JP1:OFF/ON JP2:ON の場合

CPU(CS89712)オンチップのブート ROM から起動します。このブート ROM は、COM1 にシリアル接続された機器からプログラムを送り込んで実行するためのホストとして動作するものです。オンボード Flash から起動できなくなった場合の復旧用などとして使用します。

1.3 テストモード

Armadillo は、JP3 を設定することにより、JTAG の有効/無効を切り替えることができます。

JP3: テストモード切替 (JTAG 有効/無効)

表 1-3 テストモードジャンパ設定

設定 JP3	テストモード
OFF	JTAG 無効
ON	JTAG 有効

・JP3:OFF の場合

JTAG が無効になります。

・JP3:ON の場合

JTAG が有効になります。

1.4 オンボード Flash からの Armadillo の起動

デフォルトの Armadillo の Linux カーネルは、シリアルポート COM1 を標準入出力として使用し、接続したホスト PC 上のシリアル端末アプリケーションをログイン端末とすることができます。

ここでは、ホスト PC と Armadillo をシリアルポート経由で接続し、デフォルトで Armadillo のオンボード Flash に搭載されている“ Armadillo Linux ”を起動して、操作する例を説明します。

※ホスト PC には、シリアル端末アプリケーションがインストールされている必要があります。ここでは、Linux で uucp の cu を使用する例と、Windows で Tera Term Pro を使用する例について取り上げます。

※Linux PC で cu コマンドがインストールされていない場合､ディストリビューションのマニュアルに沿って“ uucp ”パッケージをインストールしてください。

※Windows で“ Tera Term Pro ”を使用する場合、下記の URL をご参照ください。

Tera Term Home Page:

http://hp.vector.co.jp/authors/VA002416/ (2003 年 2 月 8 日現在)

1.4.1 Armadillo の起動

1. Armadillo の電源が Off であることを確認し、Armadillo の COM1 と、ホスト PC のシリアルポートをクロス(リバース)シリアルケーブルで接続する

   ※Armadillo をネットワークに接続する場合、LAN ケーブルも接続してください。

   ※デフォルトの Armadillo Linux は､DHCP サーバから IP アドレスを取得してネットワーク接続するように設定されています。DHCP を使用せず、Armadillo に固定 IP アドレスを割り振る場合、一旦ネットワーク接続せずに起動後、シリアルポートからログインしてネットワークの設定を行う必要があります。

2. ジャンパを JP1:OFF JP2:OFF に設定する。

3. ホスト PC 上で、シリアル端末アプリケーションを、以下のパラメータでシリアル接続するよう設定して起動する

   シリアル接続パラメータ
   転送レート　　: 115200 bps
   データ長　　: 8 bit
   パリティ　　: なし
   ストップビット　: 1 bit
   フロー制御　　: なし

・cu の場合

[pc]# cu -l [使用シリアルポート] -s 115200
※cu を終了するには、「~.」(チルダ/ドット)と入力します。

・Tera Term Pro の場合

Tera Term Pro を起動し、Serial モードで使用シリアルポートを指定して接続
メニューの Setup→Serial port...で上記パラメータを設定

4. Armadillo の電源を On にする
   起動ログが端末アプリケーション上に表示されます。正常に表示されない場合は、Armadillo の電源を Off にし、シリアルケーブルの接続や Armadillo のジャンパ設定、シリアル端末アプリケーションのパラメータ設定を再確認してください。

5. DHCP 接続を「Ctrl+C」でキャンセルする(ネットワーク非接続/DHCP 非使用の場合)
   ※Armadillo をネットワークに接続していない場合や、ネットワークには接続しているが DHCP を使用できない場合、「Starting DHCP for interface eth0:」と表示された後、一定時間起動スクリプトが停止いたします。この場合、シリアル端末アプリケーション上で Ctrl キーを押しながら C を入力することでキャンセルできます。
   ※DHCP が使用可能な場合、「Starting DHCP for interface eth0:」と表示後、数秒程度で DHCP 接続され「done」と表示されます(この停止時間は、お使いの DHCP サーバの応答時間によります)。

6. “ armadillo login: ”と表示される

これで、Armadillo の起動は完了です。
デフォルトで、
ユーザ名: root　　パスワード: root
ユーザ名: guest　　パスワード: なし
といったユーザーが用意されていますので、これらでログインしてください。
※Armadillo Linux には、“ logout ”コマンドは用意されていません。
ログアウトするには、“ exit ”コマンドを使用してください。

1.4.2 DHCP によるネットワーク接続

DHCP 接続の場合、上記手順で自動的に Armadillo に IP アドレスが割り当てられ、ネットワークに接続されます。Armadillo に割り当てられた IP アドレスを知りたい場合は、Armadillo にログイン後、ifconfig コマンドを使用してください。
※下線部が Armadillo の IP アドレスとなります。

```
[armadillo]# ifconfig eth0
eth0        Link encap:Ethernet HWaddr xx:xx:xx:xx:xx:xx
            inet addr:xxx.xxx.xxx.xxx Bcast.xxx.xxx.xxx.xxx Mask:xxx.xxx.xxx.xxx
    (以下省略)
```

1.4.3 固定 IP アドレスによるネットワーク接続

DHCP を使用せず、Armadillo に固定 IP アドレスを割り振る場合は、Armadillo に root ユーザでログイン後、以下の手順を行ってください。

1. /etc/network.d/interface.eth0 を書き換える

   [armadillo]# vi /etc/network.d/interface.eth0

   (/etc/network.d/interface.eth0 の例)

```
INTERFACE="eth0"
DHCP="no"
IPADDRESS="[Armadillo に割り当てる IP アドレス]"
NETMASK="[Armadillo を接続するネットワークのネットマスク]"
BROADCAST="[Armadillo を接続するネットワークのブロードキャスト]"
GATEWAY="[Armadillo を接続するネットワークのゲートウェイアドレス]"
```

   ※/etc/network.d/sample に固定 IP アドレスを割り振る場合のサンプルがありますので、参考にしてください。

2. DNS サーバを設定する場合、/etc/resolv.conf を書き換える

   vi /etc/resolv.conf

   (/etc/resolv.conf の例)

```
nameserver [DNS サーバの IP アドレス]
```

3. /etc/rc.d/rc.start/rc.40.network を起動する

   [armadillo]# /etc/rc.d/rc.start/rc.40.network

なお、ここで行った変更は RAM ディスク上に書き込まれますので、次回起動時には反映されません。毎起動時の設定を変更してオンボード Flash に保存したい場合は、**“2.8 ユーザランド RAM ディスクイメージの更新”**を参照してください。

1.5 telnet による Armadillo へのログイン

Armadillo がデフォルトのオンボード Flash から正常に起動し、ネットワークに接続されている場合、同じネットワークに接続された他の PC から telnet によりログインして操作することが可能です。

ログイン可能なユーザは、デフォルトでは

ユーザ名: guest　　パスワード: (なし)

のみですので、このユーザでログインしてください。

root 権限が必要な操作を telnet 経由で行う場合、一般ユーザでログイン後、su コマンドで root ユーザ(パスワード:root)に変更してください。

1.6 ftp による Armadillo へのログイン

Armadillo がデフォルトのオンボード Flash から正常に起動し、ネットワークに接続されている場合、同じネットワークに接続された他の PC から ftp によりログインしてファイルを送受信することが可能です。

ログイン可能なユーザは、デフォルトでは

ユーザ名: anonymous　　パスワード: (なし)

ユーザ名: ftp　　パスワード: (なし)

ですので、これらのユーザでログインしてください。

※ftp によるログイン直後のトップディレクトリは、Armadillo 上での/home/ftp になります。

※デフォルトでは、ログイン直後のトップディレクトリからはダウンロードのみ可能、/pub ディレクトリ下はアップロード/ダウンロードともに可能な状態に設定されています。

1.7 WEB ブラウザからの Armadillo の閲覧

Armadillo がデフォルトのオンボード Flash から正常に起動し、ネットワークに接続されている場合、同じネットワークに接続された他の PC から WEB ブラウザによる閲覧が可能です。

PC で WEB ブラウザを起動し、“ http://[Armadillo の IP アドレス] ”を指定してください。

※デフォルトでは、/home/www-data ディレクトリが、WEB サーバのトップディレクトリとなっています。

1.8 Armadillo の終了

Armadillo を終了する場合、halt コマンドを使用します。
halt コマンドを実行後、

System halted.
Kernel panic: Attempted to kill init!

とシリアル端末に表示されるのを待ってから、電源を Off にしてください。

ただし、拡張ボードや外部機器などを接続しておらず、Compact Flash をマウントしていない場合は、halt コマンドを使用せず、いきなり電源を Off にしても問題はありません。
※拡張ボードや外部機器を接続している場合は、その機器の仕様を優先してください。
※Compact Flash をマウント中に Armadillo の電源を Off にした場合、Compact Flash 上のデータが破壊されることがあります。先にアンマウントするか、halt コマンドで終了してください。
※Compact Flash 上のシステムから起動した場合は、アンマウントすることはできませんので、必ず halt コマンドを使用してください。

2 クロス開発環境

2.1 クロス開発環境のインストール

Armadillo で動作するカーネルやアプリケーションは、Linux の動作する PC(AT 互換機)上でクロス開発することができます。

クロス開発環境として、Armadillo には以下のパッケージが用意されています。

binutils	Binary utilities
cpp	The GNU C preprocessor
gcc	The GNU C compiler
g++	The GNU C++ compiler
libstdc++	GNU stdc++ library
libstdc++-dev	GNU stdc++ library (development files)

これらのパッケージは、deb/rpm/tgz の３種類の形式のものが用意されています。クロス開発を行う PC でお使いのディストリビューションに合ったものをインストールしてください。

※各パッケージのインストールは、root ユーザで行う必要があります

・パッケージインストール方法

deb パッケージを使用する場合 (Debian 系)

```
[pc]# dpkg -i [deb パッケージ名]
```

rpm パッケージを使用する場合 (RedHat 系)

```
[pc]# rpm -i [rpm パッケージ名]
```

tgz 圧縮ファイルを使用する場合 (Slackware など)

```
[pc]# cd /
[pc]# tar -zxf [tgz 圧縮ファイル名]
```

※パッケージ管理ツール(dpkg/rpm)についての詳細な情報は、man コマンドや各ディストリビューションに付属するドキュメントなどを参照してください。

2.2 クロス開発環境用ライブラリ群のインストール

クロス開発環境用の標準 C ライブラリ群として、Armadillo には以下のパッケージが用意されています。

libc6-arm-cross	GNU C Library
libc6-dev-arm-cross	GNU C Library (Development)

これらのパッケージは、deb/rpm/tgz の３種類の形式のものが用意されています。クロス開発を行う PC でお使いのディストリビューションに合ったものをインストールしてください。

※パッケージインストール方法については、**“ 2.1 クロス開発環境のインストール ”**を参考にしてください。

2.3 シリアルダウンローダ/ オンボード Flash ライタのインストール

Armadillo は、シリアルポートに接続した Linux の動作する PC からデータを送り込み、オンチップ Flash を書き換えることができます。

このためのアプリケーションとして、Armadillo には以下のパッケージが用意されています。

shoehorn	CPU オンチップブート ROM と協調動作するダウンローダ
hermit	Armadillo ブートプログラムと協調動作するダウンローダ ※Armadillo ブートプログラム自体も含みます

これらのパッケージは、deb/rpm/tgz の３種類の形式のものが用意されています。クロス開発を行う PC でお使いのディストリビューションに合ったものをインストールしてください。

※パッケージインストール方法については、**“ 2.1 クロス開発環境のインストール ”**を参考にしてください。

2.4 クロス開発環境での開発

PC にクロス開発環境パッケージをインストールすると、PC 上で ARM-Linux 上で動作するアプリケーションやライブラリを開発が可能になります。

ARM-Linux をターゲットとしたバイナリファイルを make するには、開発ユーティリティの各コマンドやヘッダ・ライブラリとして、ARM-Linux 用のものを指定する必要があります。通常(ネイティブ開発環境)のコマンドとの対応は、下記のようになります。

表 2-1 クロス開発環境コマンド一覧

binutils の各コマンド	addr2line ar as c++filt gasp ld nm objcopy objdump ranlib readelf size string strip	arm-linux-addr2line arm-linux-ar arm-linux-as arm-linux-c++filt arm-linux-gasp arm-linux-ld arm-linux-nm arm-linux-objcopy arm-linux-objdump arm-linux-ranlib arm-linux-readelf arm-linux-size arm-linux-string arm-linux-strip
gcc の各コマンド	gcc	arm-linux-gcc
g++の各コマンド	g++	arm-linux-g++
標準インクルードパス	/usr/include (省略可能)	/usr/arm-linux/include (省略不可)
標準ライブラリパス	/usr/lib (省略可能)	/usr/arm-linux/lib (省略不可)

例として、C ソースファイル sample1.c をコンパイルして実行ファイル sample1 出力する場合、下記の用に記述します。

(ネイティブ環境用)

```
gcc sample1.c -o sample1
    または
gcc -I/usr/include –L/usr/lib sample1.c -o sample1
```

(ARM-Linux 用)

```
arm-linux-gcc -I/usr/arm-linux/include -L/usr/arm-linux/lib sample1.c -o sample1
```

2.5 カーネルイメージの作成

Armadillo には、カーネルのソースファイルが付属しています。ここでは、このソースをクロス開発環境をインストールした PC 上でコンパイルし、カーネルイメージを作成する手順について説明します。

※クロス開発環境のインストールは、**“ 2.1 クロス開発環境のインストール ”** を参照してください。

1. カーネルソース群を make するドライブに展開し、カーネルソースディレクトリに移動する。

```
[pc]# gzip -cd [カーネルソース圧縮ファイル] | tar -xf -
[pc]# cd [展開されたカーネルソースのディレクトリ]
```

2. コンフィグ設定を行う。

```
[pc]# make menuconfig
```

※設定が完了したら、カーソルキーの左右で<Exit>を選択して“ Enter ”キーを押し、“ Do you wish to save your new kernel configuration? ”と表示されたら<Yes>を選択して、設定を保存してください。

3. 依存関係記述ファイルを更新する

```
[pc]# make dep
```

4. カーネルを make し、Image ファイルを出力する

```
[pc]# make r
```

これで、カーネルの作成は完了です。カーネルソースディレクトリに作成される“ Image ”及び“ Image.gz ”がカーネルイメージファイル及びその圧縮ファイルです。オンボード Flash からの起動で使用する場合は後述の **“ 2.10 オンボード Flash への書き込み ”** を、Compact Flash 上で使用する場合は **“ 3.1 Compact Flash への Linux システムの構築 ”** を参照してください。

また、前回の make による中間ファイルをすべて消去する方法は、下記の通りです。
※コンフィグ設定情報と依存関係記述ファイルは消去されず、そのまま残ります

```
[pc]# make clean
```

2.6 Armadillo オリジナルデバイスドライバ仕様

Armadillo に搭載されている各デバイス用のドライバは、デフォルトのカーネルに内蔵されています。各ドライバの仕様は、以下の通りです。

2.6.1 シリアルポート

(ソースファイル: driver/serial/serial_clps711x.c)

Armadillo のシリアルポート UART1(CON3)及び UART2(CON4)に対応するデバイスノードのパラメータは、以下の通りです。

表 2-2 シリアルポートノード一覧

タイプ	メジャー番号	マイナー番号	ノード名 (/dev/???)	デバイス名
キャラクタデバイス	204	16	ttyAM0	UART1 Data Register
		17	ttyAM1	UART2 Data Register

※シリアルポートのドライバは、フロー制御機能を持っていません。
※デフォルトのカーネルは、起動ログを UART1 に出力します。
※デフォルトの Armadillo Linux および Debian GNU/Linux は、UART1 を端末として使用するために占有します。

2.6.2 パラレルポート (ソースファイル: driver/char/cs89712port.c)

パラレルポート(CON5)に対応するデバイスノードのパラメータは、以下の通りです。

表 2-3 パラレルポートノード一覧

タイプ	メジャー番号	マイナー番号	ノード名 (/dev/???)	デバイス名
キャラクタデバイス	210	16	pbdr	Port B Data Register 全 CH (8bit)
		17	pbdr0	Port B Data Register CH0 (Pin.3)
		18	pbdr1	Port B Data Register CH1 (Pin.4)
		19	pbdr2	Port B Data Register CH2 (Pin.5)
		20	pbdr3	Port B Data Register CH3 (Pin.6)
		21	pbdr4	Port B Data Register CH4 (Pin.7)
		22	pbdr5	Port B Data Register CH5 (Pin.8)
		23	pbdr6	Port B Data Register CH6 (Pin.9)
		24	pbdr7	Port B Data Register CH7 (Pin.10)
		144	pbddr	Port B Data Direction Register 全 CH (8bit)
		145	pbddr0	Port B Data Direction Register CH0 (Pin.3)
		146	pbddr1	Port B Data Direction Register CH1 (Pin.4)
		147	pbddr2	Port B Data Direction Register CH2 (Pin.5)
		148	pbddr3	Port B Data Direction Register CH3 (Pin.6)
		149	pbddr4	Port B Data Direction Register CH4 (Pin.7)
		150	pbddr5	Port B Data Direction Register CH5 (Pin.8)
		151	pbddr6	Port B Data Direction Register CH6 (Pin.9)
		152	pbddr7	Port B Data Direction Register CH7 (Pin.10)

- データ型

pbdr/pbddr (全 CH): unsigned char(符号なし 8bit) 0x00～0xff

pbdr0～7/pbddr0～7 (各 CH): unsigned char(符号なし 8bit) 0x00 / 0x01

パラレルポート各ピンを入出力どちらで使用するかを pbddr で設定(0:入力/1:出力)し、データの読み書きを pbdr で行うことができます。

pbdr0～7/pbddr0～7 は各 CH ごとについての読み書きが可能で、pbdr/pbddr は全 CH(8bit)一括の読み書きが可能です。CH0(pbdr0/pbddr0)が最下位ビット、CH7(pbdr7/pbddr7)が最上位ビットとして、pbdr/pbddr(全 CH)の各ビットに対応します。

(パラレルポート操作のサンプル)

```
#include <fcntl.h>
#include <stdio.h>

int main (void)
{
        int fd_ddr, fd_dr;
        unsigned char val;

        //CH0 の Direction を書き込み専用でオープン
        fd_ddr = open (“/dev/pbddr0”, O_WRONLY);
        if (fd_ddr < 0) {
                fprintf (stderr, “Open error.¥n”);
                return –1;
        }
        // CH0 を読み書き可能でオープン
        fd_dr = open (“/dev/pbdr0”, O_RDWR);
        if (fd_dr < 0) {
                fprintf (stderr, “Open error.¥n”);
                return –1;
        }

        val = 1;
        write (fd_ddr, &val, sizeof(unsigned char)); //CH0 を出力に
        val = 1;
        write (fd_dr, &val, sizeof(unsigned char)); //CH0 に High を出力

        val = 0;
        write (fd_ddr, &val, sizeof(unsigned char)); //CH0 を入力に
        read (fd_dr, &val, sizeof(unsigned char)); //CH0 を val に読み込む
        printf (“pbdr0: %d¥n”, val); //val を表示

        close (fd_ddr);
        close (fd_dr);

        return 0;
}
```

2.6.3 A/D コンバータ (ソースファイル: driver/ssi/ssi-max149.c)

A/D コンバータ(CON2)に対応するデバイスノードのパラメータは、以下の通りです。

表 2-4 A/D コンバータノード一覧

タイプ	メジャー番号	マイナー番号	ノード名 (/dev/???)	デバイス名
キャラクタデバイス	211	0	adcs0	シングルエンドモード電圧値 CH0 (Pin.3)
		1	adcs1	シングルエンドモード電圧値 CH1 (Pin.4)
		2	adcs2	シングルエンドモード電圧値 CH2 (Pin.5)
		3	adcs3	シングルエンドモード電圧値 CH3 (Pin.6)
		4	adcs4	シングルエンドモード電圧値 CH4 (Pin.7)
		5	adcs5	シングルエンドモード電圧値 CH5 (Pin.8)
		6	adcs6	シングルエンドモード電圧値 CH6 (Pin.9)
		7	adcs7	シングルエンドモード電圧値 CH7 (Pin.10)
		8	adcd0_1	差動モード電圧値 CH0-CH1 (Pin.3-Pin.4)
		9	adcd1_0	差動モード電圧値 CH1-CH0 (Pin.4-Pin.3)
		10	adcd2_3	差動モード電圧値 CH2-CH3 (Pin.5-Pin.6)
		11	adcd3_2	差動モード電圧値 CH3-CH2 (Pin.6-Pin.5)
		12	adcd4_5	差動モード電圧値 CH4-CH5 (Pin.7-Pin.8)
		13	adcd5_4	差動モード電圧値 CH5-CH4 (Pin.8-Pin.7)
		14	adcd6_7	差動モード電圧値 CH6-CH7 (Pin.9-Pin.10)
		15	adcd7_6	差動モード電圧値 CH7-CH6 (Pin.10-Pin.9)

- データ型

  int (符号付き 32bit)

  バイポーラモード: 0xfffffc00(-1.25v)～0x00000000(0v)～0x000001ff(+1.25v)

  ユニポーラモード: 0x00000000(0v)～0x000002ff(+2.5v)

  ※但し、差動モードの場合でも、双方の入力電圧は 0(GND)～3.3v(Vdd)の範囲を超えられません。詳しくは MAX149 データシートを参照してください。

- モードコントロール

バイポーラモード: ioctl システムコールで MAX149_IOCTL_BIP(=0)を指定

ユニポーラモード: ioctl システムコールで MAX149_IOCTL_UNI(=1)を指定

adcs0～7 で各 CH ごとの電圧値を読み取ることが可能です(シングルエンドモード)。

これに対し、adcd0_1～8_7 では、隣合った CH の電圧値の差分を読み取ることができます(差動モード)。

また、ioctl システムコールで MAX149_IOCTL_BIP(=0)を指定することでバイポーラモード(-1.25v～+1.25v/デフォルト)、MAX149_IOCTL_UNI(=1)を指定することでユニポーラモード(0v～+2.5v)を切り替えることができます。

(A/D コンバータ操作のサンプル)

```
#include <fcntl.h>
#include <stdio.h>

//カーネルソースから ssi-max149.h を指定
#include “kernel/source/linux/include/linux/ssi-max149.h”

int main (void)
{
        int fd_s0, fd_d0_1;
        int val;

        //CH0 を読み込み専用でオープン
        fd_s0 = open (“/dev/adcs0”, O_RDONLY);
        if (fd_s0 < 0) {
                fprintf (stderr, “Open error.¥n”);
                return –1;
        }
        //CH0-CH1 の差分を読み込み専用でオープン
        fd_d0_1 = open (“/dev/adcd0_1”, O_RDONLY);
        if (fd_d0_1 < 0) {
                fprintf (stderr, “Open error.¥n”);
                return –1;
        }

        ioctl (fd_s0, MAX149_IOCTL_UNI); //ユニポーラモードに設定
        read (fd_s0, &val, sizeof(int)); //CH0 の値を val に読み込む
        printf (“CH0: %lfV¥n”, (double)val * 2.5 / 1023.0); //val を V 単位表示
        ioctl (fd_d0_1, MAX149_IOCTL_UNI); //ユニポーラモードに設定
        read (fd_d0_1, &val, sizeof(int)); //CH0-1 の値を val に読み込む
        printf (“CH0-1: %lfV¥n”, (double)val * 2.5 / 1023.0); //val を V 単位表示

        close (fd_d0_1);
        close (fd_s0);
        return 0;
}
```

2.6.4 リアルタイムクロック

(ソースファイル: driver/i2c/i2c-s3531a.c)

リアルタイムクロック(RTC)S-3531A のドライバは、OS 標準 RTC として動作します。デバイスノードのパラメータは、以下の通りです。

表 2-5 リアルタイムクロックノード一覧

タイプ	メジャー番号	マイナー番号	ノード名 (/dev/???)	デバイス名
キャラクタデバイス	10	135	rtc	リアルタイムクロック S-3531A

2.6.5 CPU オンチップ SRAM/ブート ROM

(ソースファイル: driver/mtd/maps/mtd-armadillo.c)

Armadillo の CPU(cs89712)には、SRAM とブート ROM が内蔵されています。この 2 つのデバイスについては、デフォルトのカーネルでは Memory Technology Device(MTD) としてマップします。各デバイスノードのパラメータは、以下の通りです。

表 2-6 MTD デバイスノード一覧

タイプ	メジャー番号	マイナー番号	ノード名 (/dev/???)	デバイス名
キャラクタデバイス	90	0	mtd0	CPU オンチップ SRAM
		1	mtd1	CPU オンチップ ブート ROM
ブロックデバイス	31	0	mtdblock0	CPU オンチップ SRAM
		1	mtdblock1	CPU オンチップ ブート ROM

2.7 割り込み(IRQ)と Linux 上での扱いについて

Armadillo 搭載 CPU CS89712 の割り込みは、複数の CPU レジスタで管理されています。

表 2-7 CPU 割り込み一覧

CPU 割込レジスタ	ビット	名称	説明
INTMR1/INTSR1	4	CSINT	Codec sound interrupt
INTMR1/INTSR1	5	EINT1	External interrupt input 1
INTMR1/INTSR1	6	EINT2	External interrupt input 2
INTMR1/INTSR1	7	EINT3	External interrupt input 3 (Ethernet)
INTMR1/INTSR1	8	TC1OI	TC1 underflow interrupt
INTMR1/INTSR1	9	TC2OI	TC2 underflow interrupt
INTMR1/INTSR1	10	RTCMI	RTC Compare match interrupt
INTMR1/INTSR1	11	TINT	64 Hz tick interrupt
INTMR1/INTSR1	12	UTXINT1	Internal UART1 transmit FIFO empty interrupt
INTMR1/INTSR1	13	URXINT1	Internal UART1 receive FIFO full interrupt
INTMR1/INTSR1	14	UMSINT	Internal UART1 modem status changed interrupt
INTMR1/INTSR1	15	SSEOT1	Synchronous serial interface 1 end of transfer interrupt
INTMR2/INTSR2	0	KBDINT	Key press interrupt
INTMR2/INTSR2	1	SS2RX	Master / slave SSI 16 bytes received
INTMR2/INTSR2	2	SS2TX	Master / slave SSI 16 bytes transmited
INTMR2/INTSR2	12	UTXINT2	UART2 transmit FIFO empty interrupt
INTMR2/INTSR2	13	URXINT2	UART2 receive FIFO full interrupt

また、PC/104 準拠バス及び IDE の割り込みが PLD 内の割り込みコントローラで管理されており、このコントローラは上記 CPU 割り込みの EINT1 に接続されています。
※この点の詳細については、ハードウェアマニュアルの 6.1 節をご参照ください。

表 2-8 PC/104 割り込み一覧

PLD 割込レジスタ	ビット	名称	説明
MISCREG_INTMR/SR/CR3	0	ISA3	PC/104 #3 interrupt
MISCREG_INTMR/SR/CR3	1	ISA4	PC/104 #4 interrupt
MISCREG_INTMR/SR/CR3	2	ISA5	PC/104 #5 interrupt
MISCREG_INTMR/SR/CR3	3	ISA6	PC/104 #6 interrupt
MISCREG_INTMR/SR/CR2	0	ISA7	PC/104 #7 interrupt
MISCREG_INTMR/SR/CR2	1	ISA9	PC/104 #9 interrupt
MISCREG_INTMR/SR/CR2	2	ISA10	PC/104 #10 interrupt
MISCREG_INTMR/SR/CR2	3	ISA11	PC/104 #11 interrupt
MISCREG_INTMR/SR/CR1	0	ISA12	PC/104 #12 interrupt
MISCREG_INTMR/SR/CR1	1	IDE	IDE interrupt
MISCREG_INTMR/SR/CR1	2	ISA15	PC/104 #15 interrupt

Armadillo の Linux 上では、これらすべての割り込みを一連の数値で管理します。このため、PC/104 の IRQ 番号は Linux 上での IRQ 番号と一致しないことに注意してください。

Linux 上での IRQ 番号は、カーネルソースの include/asm-arm/arch-clps711x/irqs.h に定義されています。

表 2-9 Linux 上での IRQ 一覧

Linux 上での IRQ	名称	説明
4	IRQ_CSINT	Codec sound interrupt
5	IRQ_EINT1	External interrupt input 1
6	IRQ_EINT2	External interrupt input 2
7	IRQ_EINT3	External interrupt input 3 (Ethernet)
8	IRQ_TC1OI	TC1 underflow interrupt
9	IRQ_TC2OI	TC2 underflow interrupt
10	IRQ_RTCMI	RTC Compare match interrupt
11	IRQ_TINT	64 Hz tick interrupt
12	IRQ_UTXIN	Internal UART1 transmit FIFO empty interrupt
13	IRQ_URXIN	Internal UART1 receive FIFO full interrupt
14	IRQ_UMSIN	Internal UART1 modem status changed interrup5
15	IRQ_SSEOT	Synchronous serial interface 1 end of transfer interrupt
16	IRQ_KBDIN	Key press interrupt
17	IRQ_SS2RX	Master / slave SSI 16 bytes received
18	IRQ_SS2TX	Master / slave SSI 16 bytes transmited
28	IRQ_UTXIN	UART2 transmit FIFO empty interrupt
29	IRQ_URXIN	UART2 receive FIFO full interrupt
30	IRQ_ISA3	PC/104 #3 interrupt
31	IRQ_ISA4	PC/104 #4 interrupt
32	IRQ_ISA5	PC/104 #5 interrupt
33	IRQ_ISA6	PC/104 #6 interrupt
34	IRQ_ISA7	PC/104 #7 interrupt
35	IRQ_ISA9	PC/104 #9 interrupt
36	IRQ_ISA10	PC/104 #10 interrupt
37	IRQ_ISA11	PC/104 #11 interrupt
38	IRQ_ISA12	PC/104 #12 interrupt
39	IRQ_IDE	IDE interrupt
40	IRQ_ISA15	PC/104 #15 interrupt

また、PC/104 の IRQ 番号を扱いやすくするため、前記ヘッダファイル”include/asm-arm/arch-clps711x/irqs.h”に IRQ 変換用インライン関数を用意しています。

```
//ISA IRQ へ(to)の変換
static __inline__ unsigned int convirq_to_isa (unsigned int irq);
```

・Linux 上での IRQ 番号から、PC/104 の IRQ 番号へ変換します。

例)
```
const unsigned int linux_irq = 3;
unsigned int isa_irq;
//(数値の)3 が ISA_IRQ3 (=30)に変換される
isa_irq = convirq_to_isa (linux_irq);
```

```
//ISA IRQ から(from)の変換
static __inline__ unsigned int convirq_from_isa (unsigned int irq);
```

・PC/104 の IRQ 番号から、Linux 上での IRQ 番号へ変換します。

例)
```
const unsigned int isa_irq = ISA_IRQ3;
unsigned int linux_irq;
//IRQ_ISA3(=30)が(数値の)3 に変換される
linux_irq = convirq_from_isa (isa_irq);
```

2.8 デバイスドライバモジュールの作成

デバイスドライバモジュールを新規作成するためのサンプルとして、梅沢無線電機株式会社製 HT2020 ボード用デバイスドライバモジュールのソースが用意されています。ここでは、このモジュールを make して、組み込むまでの手順を説明します。

1. モジュールソースを解凍する

```
[pc]# tar zxf ht2020.tgz
```

2. カーネルソースのある場所を書き換える
※あらかじめ、PC に Armadillo 用カーネルソースを展開しておいてください。

```
[pc]# cd ht2020
[pc]# vi Makefile
```

※Makefile の 1 行目の INCLUDEDIR に、Armadillo 用カーネルソースのあるパスの include ディレクトリパスを指定してください。

```
INCLUDEDIR = ../../../kernel/source/linux/include

(以下省略)
```

3. make する

```
[pc]# make
```

make が完了して出力される ht2020.o が、HT2020 用デバイスドライバモジュールです。モジュールを Armadillo 上に転送し、以下のコマンドで組み込むことができます。

```
[armadillo]# insmod ht2020.o io=0x100
```

※io には、HT2020 をマップする IO アドレスを指定してください。

これで、ドライバモジュールの組み込みは完了です。
※「lsmod」コマンドで、組み込まれているモジュールの一覧を見ることができます。
※HT2020 を使用するサンプルアプリケーションとしては、ht2020_sample.tgz が用意されていますので、参考にしてください。

2.9 ユーザランド RAM ディスクイメージの更新

Linux を使用するには、カーネルとともに、その上で動作するアプリケーションやデバイスノード(デバイスドライバの入口)などが必要です。これらはすべて、ディスクドライブ内に構築されたファイルシステム上に置かれ、全体で“ユーザランド”と呼ばれます。

通常の PC 用 Linux や、Armadillo で Compact Flash を使用する場合は、物理的なディスクドライブ上にユーザランドが構築されます。これに対し、物理的なディスクドライブを使用しない場合は、RAM 上に仮想的に作られたディスクドライブ(RAM ディスク)上にユーザランドを構築します。

Armadillo をオンボード Flash から起動する場合、RAM ディスク上に展開されるユーザランドのイメージを、オンボード Flash 上に書き込んでおく必要があります。ここでは、Armadillo Linux のユーザランドを RAM ディスクイメージとして作成したファイル“initrd.img.gz”の中身を更新する手順について説明します。

1. gzip 圧縮を解凍する
   ※解凍した“initrd.img”の状態が、RAM ディスクイメージとなります。

   ```
   [pc]# gunzip initrd.img.gz
   ```

2. RAM ディスクイメージファイルをマウントするための空のディレクトリを作成する

   ```
   [pc]# mkdir [マウントディレクトリ名]
   ```

3. RAM ディスクイメージファイルを手順 2 で作成したディレクトリにマウントする
   ※お使いのホスト PC の Linux で、loop デバイスをサポートしている必要があります。loop デバイスをサポートしていない場合は、お使いのディストリビューションのマニュアル等を参照して、loop デバイスのためのモジュールを追加するなどしてください。
   ※手順 3～5 は、root ユーザで行う必要があります。

   ```
   [pc]# mount -o loop initrd.img [マウントディレクトリ名]
   ```

4. RAM ディスクイメージをマウントしたディレクトリの中を更新する
   ※マウントディレクトリの中が、ArmadilloLinux のディレクトリツリーとなっています。ファイルの追加・削除や設定ファイルの書き換えなど、直接操作することが可能です。

   ・ホスト名を変更する例

   [pc]# cd [マウントディレクトリ名]/etc
   [pc]# vi HOSTNAME
   (HOSTNAME の例)

   ```
   [新ホスト名]
   ```

   [pc]# cd ../..

5. RAM ディスクイメージのディレクトリマウントを解除する

   [pc]# umount [マウントディレクトリ名]

6. gzip 圧縮する

   [pc]# gzip –9 initrd.img

これで、ユーザランド RAM ディスクイメージの更新は完了です。“ initrd.img.gz ”が、更新後のものとなっています。これをオンボード Flash に書き込む場合は、**“ 2.11 オンボード Flash への書き込み ”** を参照してください。

2.10 ユーザランド RAM ディスクイメージの新規作成

ユーザランド RAM ディスクイメージは、Linux の動作している PC 上で新規に作成・構築することが可能です。ここではその手順について説明します。

1. ユーザランド用のディレクトリを作成する

   [pc]# mkdir [ユーザランドディレクトリ名]

2. 手順 1 で作成したディレクトリ内を操作し、ディレクトリ作成、アプリケーションや共有ライブラリのコピー、デバイスノードの作成などを行い、ユーザランドディレクトリツリーを構築する

3. RAM ディスクイメージファイルをマウントするための空のディレクトリを作成する

   [pc]# mkdir [マウントディレクトリ名]

4. RAM ディスクイメージ用の空のファイルを作成する

   [pc]# dd if=/dev/zero of=[イメージファイル名] bs=1024
   count=[イメージファイルサイズ(KB 単位)]
   ※すべて 1 行で入力します。

5. RAM ディスクイメージファイルを ext2 ファイルシステムとして初期化する
   ※手順 5～8 の操作は、root ユーザで行う必要があります。

   [pc]# mke2fs [イメージファイル名]

6. RAM ディスクイメージファイルを手順 3 で作成したディレクトリにマウントする
   ※お使いのホスト PC の Linux で、loop デバイスをサポートしている必要があります。loop デバイスをサポートしていない場合は、お使いのディストリビューションのマニュアル等を参照して、loop デバイスのためのモジュールを追加するなどしてください。

   [pc]# mount -o loop [イメージファイル名] [マウントディレクトリ名]

7. 手順 1 で作成したディレクトリの中身を、RAM ディスクイメージをマウントしたディレクトリに tar を使用してコピーする

   [pc]# (cd [ユーザランドディレクトリ名]; tar cf - *) |
   (cd [マウントディレクトリ名]; tar xf -)
   ※すべて 1 行で入力します。

8. RAM ディスクイメージのディレクトリマウントを解除する

   [pc]# umount [マウントディレクトリ名]

これで、ユーザランド RAM ディスクイメージファイルの作成は完了です。

ただし、通常 Armadillo で使用できるユーザランド RAM ディスクイメージのサイズは最大 2.5MB(2,621,440bytes)までです。イメージファイルサイズがこれを超える場合は、以下のように圧縮してください。
※圧縮した場合、Armadillo の起動に要する時間が長くなります。

[pc]# gzip -9 [イメージファイル名]

圧縮後もイメージファイルが最大サイズより大きい場合、手順 3 で指定するサイズを小さくしたり、ユーザランドに含めるファイルを減らすなどして再作成してください。

作成したイメージファイルをオンボード Flash に書き込む場合は、**“2.11 オンボード Flash への書き込み”**を参照してください。

2.11 オンボード Flash への書き込み

カーネルイメージ/ユーザランド RAM ディスクイメージは、シリアルダウンローダをインストールしたホスト Linux PC から、シリアルポート経由で Armadillo にダウンロードして、オンボード Flash に書き込むことができます。ここでは、このダウンロード・Flash 書き込み手順について説明します。

※オンボード Flash 上のブートプログラム領域(0x00000000～0x0000ffff)に標準のブートプログラム以外を書き込んでいる場合は、以下の手順では書き込みできません。**“ 2.11 CPU オンチップ ROM 起動によるオンボード Flash への書き込み ”**を参照して書き込みを行ってください。

※シリアルダウンローダのインストールは、**“ 2.3 シリアルダウンローダ/オンボード Flash ライタのインストール ”**を参照してください。

1. Armadillo の電源が Off であることを確認し、Armadillo の COM1 と、ホスト PC のシリアルポートをクロス(リバース)シリアルケーブルで接続する

2. ジャンパを JP1:ON/JP2:OFF に設定し、Compact Flash ソケットには何も挿入されていない状態にする

3. Armadillo の電源を On にする
   ※このとき、LED(D9)が数秒間だけ点灯しますので、消灯するまで待って手順 4 に移ってください。

4. hermit でイメージをダウンロード/オンボード Flash 書き込みする

※以下は、ホスト PC 側のシリアルポート“ /dev/ttyS0 ”に Armadillo を接続した場合の例です。他のシリアルポートに接続した場合、hermit のオプションに

--port [シリアルポート名]

を追加してください。

・Linux カーネルイメージの場合

[pc]# hermit download -i [カーネルイメージファイル] -a 0x10000

※指定するカーネルイメージファイルは、非圧縮の“ Image ”、または圧縮済の“ Image.gz ”です。使用可能なイメージファイルのサイズは、最大 1,507,328 バイトまでです。作成したカーネルの“ Image ”が最大サイズを超える場合、“ Image.gz ”を使用してください。

・ユーザランド RAM ディスクイメージの書き込み

[pc]# hermit download -i [ユーザランドイメージファイル] -a 0x180000

※使用可能なイメージファイルのサイズは、最大 2,621,440 バイトまでです。イメージファイルが最大サイズを超える場合は書き込むことができませんので、RAM ディスクイメージサイズを小さくするなど再作成してください。

5. Armadillo の電源を Off にする

これで、オンボード Flash への書き込みは完了です。

オンボード Flash に書き込んだ Linux を起動する場合は、**“ 1.4 オンボード Flash からの Armadillo の起動 ”**を参照してください。

2.12 CPU オンチップ ROM 起動による オンボード Flash への書き込み

オンボード Flash 上のブートプログラム領域(0x00000000-0x00010000)に標準のブートプログラム以外を書き込んでいる場合でも、Armadillo を CPU オンチップ ROM から起動することで、シリアルダウンローダをインストールしたホスト Linux PC から、シリアルポート経由で Armadillo にイメージをダウンロードして、オンボード Flash へ書き込むことができます。

ここでは、この CPU オンチップ ROM 起動によるダウンロード・Flash 書き込み手順について説明します。

※シリアルダウンローダのインストールは、**“2.3 シリアルダウンローダ/オンボード Flash ライタのインストール”**を参照してください。

1. Armadillo の電源が Off であることを確認し、Armadillo の COM1 と、ホスト PC のシリアルポートをクロス(リバース)シリアルケーブルで接続する

2. ジャンパを JP2:ON に設定する

3. shoehorn を起動する
   ※以下は、ホスト PC 側のシリアルポート“/dev/ttyS0”に Armadillo を接続した場合の例です。他のシリアルポートに接続した場合、shoehorn のオプションに
   --port [シリアルポート名]
   を追加してください。

   ```
   [pc]# shoehorn --armadillo --boot --terminal --loader /usr/lib/shoehorn/loader.bin
         --kernel /usr/lib/hermit/loader-armadillo-boot.bin --initrd /dev/null
   ```
   ※すべて 1 行で入力します。

4. Armadillo の電源を On にする
   ※すぐにメッセージ表示が開始されます。正常に表示されない場合は、Armadillo の電源を Off にし、シリアルケーブルの接続や Armadillo のジャンパ設定を確認してください。

5.“hermit> ”と表示されたら、Ctrl+C をキー入力する

ここまでで、ホスト PC から hermit を使用して Armadillo へシリアルダウンロードを行うための準備が整います。

オンボード Flash へのイメージの書き込み方は、**“ 2.11 オンボード Flash への書き込み ”**の手順 4 以降と同様です。ブートプログラムをデフォルトのものに戻す場合、下記の通り行います。

```
[pc]# hermit download -i /usr/lib/hermit/loader-armadillo.bin -a 0x0 --force-locked
```

※すべて 1 行で入力します。

※オンボード Flash 上のブートプログラム領域(0x00000000～0x0000ffff)を hermit で書き換える場合、「--force-locked」オプションが必要となります。

2.13 Win32 版 Hermit ホストについて

Hermit V1.3-armadillo-4 では、Windows 環境からシリアル経由でオンボード Flash を書き換えるためのユーティリティ(Hermit ホスト)を提供しています。Linux 版 Hermit と同様、Armadillo をオンボード Flash からブートさせて Flash を書き換えることができます。また、同時に Shoehorn 相当の機能もサポートしておりますので、オンチップ ROM からのブートによる Armadillo の Flash 書き換えが必要な場合も対応できます。

[インストール]

”hermit-1.3-armadillo-4_win32.zip”が Win32 版 Hermit(Shoehorn 相当 DLL を含む)のパッケージです。WindowsPC 上で zip 解凍の可能なユーティリティを使用して、インストールしたい任意のディレクトリに解凍してください。

[オンボード Flash への書き込み]

1. Armadillo と WindowsPC の接続
   Armadillo の電源が Off であることを確認し、Armadillo の COM1 と、ホスト PC のシリアルポートをクロス(リバース)シリアルケーブルで接続してください。
2. Armadillo のジャンパ設定
   ジャンパを JP1:ON/JP2:OFF に設定し、Compact Flash ソケットには何も挿入されていない状態にしてください。
3. Armadillo の起動
   Armadillo の電源を入れてください。
4. Hermit の起動
   Armadillo を接続したシリアルポートを使用して Terminal アプリケーションを起動している場合、これを終了してください。シリアルポートを使用しているアプリケーションがないことを確認したら、hermit.exe をダブルクリックし、実行してください。Hermit host for win32 ウィンドウが表示されます。
5. Flash 書き込みの設定
   ウィンドウ上で、下記の各条件が設定できます。基本的に Linux 版のオプションと同一の名称になっていますので、それぞれ適切に設定を行ってください。
   - ・Port　Armadillo を接続したシリアルポート名称(COM1、COM2 など)
   - ・Input file　Flash に書き込むデータファイル名称
     (ブートローダを書き込む場合は Hermit と同一ディレクトリにある”loader-armadillo.bin”、その他の場合は任意のファイル)
   - ・Address　書き込み先頭アドレス
   - ・Verbose　ON にすると、書き込みログの出力を詳細表示する
   - ・Force locked　ON にすると、ブートローダ自身への上書きを許可する
     (アドレス 0x0 から始まるブートローダ領域を書き換える場合は、この設定を ON にする必要があります)
   - ・Shoehorn　Shoehorn 相当の機能を使用する(次項目で説明します)
6. Flash 書き込みの開始
   「Download」ボタンを押してください。Flash の書き込みを開始します。
7. Hermit for Win32 の終了
   「serial: completed 0x00XXXXXX (NNNNNN) bytes.」と表示され Flash の書き込みが完了したら、「Exit」ボタンを押しアプリケーションを終了してください。

[CPU オンチップ ROM 起動によるオンボード Flash への書き込み]

1. Armadillo と WindowsPC の接続
   Armadillo の電源が Off であることを確認し、Armadillo の COM1 と、ホスト PC のシリアルポートをクロス(リバース)シリアルケーブルで接続してください。
2. Armadillo のジャンパ設定
   ジャンパを JP1:OFF/JP2:ON に設定し、Compact Flash ソケットには何も挿入されていない状態にしてください。
3. Hermit の起動
   Armadillo を接続したシリアルポートを使用して Terminal アプリケーションを起動している場合、これを終了してください。シリアルポートを使用しているアプリケーションがないことを確認したら、hermit.exe をダブルクリックし、実行してください。Hermit host for win32 ウィンドウが表示されます。
4. Flash 書き込みの設定
   ウィンドウ上の「Shoehorn」ボタンを押下してください。
5. Armadillo の起動
   ターミナルウィンドウ上に

   ```
   loader.bin: 1816 bytes (2048 bytes buffer)
   loader-armadillo-boot.bin: 23856 bytes
   shoehorn: warning: loader stack might clobber code
   Waiting for target - press Wakeup now.
   ```

   のように表示されることを確認したら、Armadillo の電源を入れてください。
6. Hermit 起動
   Armadillo 上にローダがロードされ、Hermit ホストが使用できる状態になります。以降は前ページ[CPU オンチップ ROM 起動によるオンボード Flash への書き込み]の 5.以降とまったく同様に操作できますので、そちらを参照してください。

2.14 ターミナルとして使用するシリアルポートの変更について

Armadillo は、デフォルトでシリアルポート COM1 をターミナルとして使用するよう設定されていますが、代替として COM2 をターミナルとして使用するためのイメージも用意されています。

COM2 をターミナルとして使用する場合、以下を参照してイメージファイルを選択し、オンボード Flash を書き換えて使用してください。

※オンボード Flash の書き換え方法は、**“2.11 オンボード Flash への書き込み”**、**“2.12 CPU オンチップ ROM からの起動によるオンボード Flash への書き込み”**、**“2.13 Win32 版 Hermit ホストについて”**を参照してください。

表 2-10 オンボード Flash 書き込みイメージファイル対応表

ターミナルとして使用するシリアルポート	COM1	COM2
ブートローダ (先頭アドレス:0x00000000)	loader-armadillo.bin	loader-armadillo-ttyAM1.bin
カーネル (先頭アドレス:0x00010000)	Image または Image.gz	Image または Image.gz (COM1 の場合と同一)
ユーザランド (先頭アドレス:0x00180000)	initrd.img.gz	initrd-ttyAM1.img.gz

※ブートローダイメージ”loader-armadillo.bin”および”loader-armadillo-ttyAM1.bin”は、Linux 用 Hermit パッケージ、Hermit for win32 パッケージ両方に同梱されています。

書き換え後の操作については、**“1.4 オンボード Flash からの Armadillo の起動”**を、「COM1」を「COM2」と読み替えながら参照してください。

3 Compact Flash システム構築

3.1 Compact Flash への Linux システムの構築

Armadillo は、Compact Flash に搭載した Linux システムから起動することが可能です。このための、CompactFlash 用 Armadillo Linux イメージと、Debian GNU/Linux 2.2 のイメージが用意されています。

ここでは、これらの Linux システムイメージを、PC からネットワーク経由で Armadillo に転送し、Comapact Flash 上に構築する手順について説明します。

※それぞれの Linux システムのインストール直後のディスク使用容量は、以下の通りです。使用用途によってある程度のディスク空き容量も必要になりますので、この点を考慮して、十分な容量の Compact Flash をご用意ください。

Armadillo Linux:	約 6MB
Debian GNU/Linux 2.2 (標準インストール版):	約 70MB
Debian GNU/Linux 2.2 (開発環境インストール版):	約 100MB

1. Armadillo の電源が Off であることを確認し、Armadillo の COM1 と、ホスト PC のシリアルポートをクロス(リバース)シリアルケーブルで接続する

2. ジャンパを JP1:OFF/JP2:OFF に設定し、空の Compact Flash を挿入する。

3. **“1.4 オンボード Flash からの Armadillo の起動”**を参考にしてオンボード Flash から Armadillo を起動し、ネットワークに接続する

4. Armadillo 上で、Compact Flash のパーティションを設定する
   ※Armadillo 上で Compact Flash から起動する場合、起動するパーティションのタイプに 0x83(Linux)を設定する必要があります。
   ※以降の Armadillo 側の操作は、すべて root ユーザで行う必要があります。

   [armadillo]# fdisk /dev/hda
   ※fdisk コマンドの例
   d コマンドで既存のパーティションを削除
   n コマンドでパーティションを作成
   t コマンドでパーティションタイプを 83(Linux)に設定
   w コマンドで設定を書き込み、fdisk を終了

5. 作成したパーティションを、EXT2 ファイルシステムとして初期化する
※Armadillo のブートプログラムから Compact Flash 上のシステムを起動する場合、mke2fs のオプションに必ず“ -O none ”をつけてください。

   [armadillo]# mke2fs -O none [パーティションデバイス(/dev/hda1 など)]

6. Compact Flash を/mnt にマウントする

   [armadillo]# mount [パーティションデバイス(/dev/hda1 など)] /mnt

7. Armadillo の/home/ftp/pub に、RAM ファイルシステムをマウントし、一般ユーザーに書き込み権限を与える

   [armadillo]# mount -t ramfs ramfs /home/ftp/pub
   [armadillo]# chmod 777 /home/ftp/pub

8. ホスト PC から Armadillo に ftp で接続し、システムイメージファイルを転送する

```
[pc]# ftp [Armadillo の IP アドレス]
Name: ftp
Password: (なし)
ftp> cd pub
ftp> binary
ftp> put rootimage.tgz          [注: Armadillo Linux の場合]
ftp> bye
```

※Debian GNU/Linux 2.2 の場合、put するファイル名が debian1～2.tgz(標準インストール版の場合)、または debian_devel1～3.tgz(開発環境インストール版)となります。以降の“ rootimage.tgz ”ファイルも当該ファイル名に読み替えてください。

※Debian GNU/Linux 2.2 の場合、システムイメージファイルは、標準インストール版で 2 ファイル、開発環境インストール版 3 ファイルに分割されていますので、手順 8～9 を繰り返して、すべてのファイルを Compact Flash 上に展開してください。

9. Armadillo 上で、システムイメージファイルを Compact Flash に展開する

```
[armadillo]# (cd /mnt; tar zxf /home/ftp/pub/rootimage.tgz)
```

※Debian GNU/Linux 2.2 の場合、

```
[armadillo]# rm /home/ftp/pub/[イメージファイル名]
```

としてイメージファイルを削除し、すべての分割ファイルを展開するまで手順 8～9 を繰り返してください。

10. Compact Flash をアンマウントし、Armadillo を終了する

```
[armadillo]# umount /mnt
[armadillo]# halt
```

これで、Compact Flash 上に Armadillo で起動可能な Linux システムが構築されます。Compact Flash から起動したい場合は、次の“**3.2 Compact Flash からの Armadillo の起動**”を参考にしてください。

3.2 Compact Flash からの Armadillo の起動

Armadillo を Compact Flash 上のシステムから起動する場合、ジャンパを JP1:ON/JP2:OFF と設定し、起動するシステムの入った Compact Flash を挿入してください。

他はオンボード Flash からの起動と変わりありません。**“1.4 オンボード Flash からの Armadillo の起動”**を参考にしてください。

4 各システム収録アプリケーション

4.1 Armadillo Linux 収録アプリケーションについて

オンボード Flash にデフォルトで搭載されている Armadillo Linux に収録されているアプリケーションは、以下のソースファイルパッケージを使用しています。

表 4-1 Armadillo Linux 収録アプリケーション一覧

busybox	Tiny utilities for small and emdebbed systems. (組込機器向けの小さな基本ユーティリティ集)
cron	management of regular backgroud processing (バックグラウンドプロセス管理サービス)
e2fsprogs	The EXT2 file system utilities and libraries (EXT2 ファイルシステムユーティリティとライブラリ)
iptables	IP packet filter administration for 2.4.X kernels (IP パケットフィルター管理ユーティリティ)
linux-ftpd	FTP server (FTP サーバ)
netbase	Basic TCP/IP networking binarys (基本 TCP/IP ネットワーキング) →うち、inetd(inetd サーバ)のみ収録
netkit-telnet	The telnet client. / The telnet server. (TELNET クライアント/TELNET サーバ)
ntp	Daemon and utilities for full NTP v4 timekeeping participation. (NTP v4 デーモンとユーティリティ) →うち、ntpdate(NTP クライアントユーティリティ)のみ収録
pump	Simple DHCP/BOOTP client for 2.2.x kernels (DHCP/BOOTP クライアント)
thttpd	tiny/turbo/throttling HTTP server (tiny/turbo/throttling HTTP サーバ)
tiny-login	a multi-call binary for login and user account administration (ログイン/ユーザアカウント管理)
util-linux	Miscellaneous system utilities. (様々なシステムユーティリティ) →うち、hwclock(ハードウェアクロックユーティリティ)、 及び fdisk(Linux パーティションテーブルエディタ)のみ収録

4.2 Armadillo 用 Debian GNU/Linux 2.2 について

Compact Flash 向け Debian GNU/Linux 2.2 として、標準インストール版(potato_std)と開発環境インストール版(potato_devel)が用意されています。これらの収録パッケージは、以下の通りです。

※標準インストール版・開発環境インストール版それぞれについて、“○”がインストール済みパッケージ、“－”が未インストールパッケージを表しています。

表 4-2 Armadillo 向け Debian GNU/Linux 2.2 収録アプリケーション一覧

標準	開発	パッケージ名	パッケージ説明
○	○	adduser	Add users and groups to the system.
○	○	ae	Anthony's Editor -- a tiny full-screen editor
○	○	apt	Advanced front-end for dpkg
○	○	at	Delayed job execution and batch processing
○	○	base-config	Debian base configuration package
○	○	base-files	Debian base system miscellaneous files
○	○	base-passwd	Debian Base System Password/Group Files
○	○	bash	The GNU Bourne Again SHell
○	○	bc	The GNU bc arbitrary precision calculator language
－	○	binutils	The GNU assembler, linker and binary utilities.
○	○	bsdmainutils	More utilities from 4.4BSD-Lite.
○	○	bsdutils	Basic utilities from 4.4BSD-Lite.
－	○	bzip2	A high-quality block-sorting file compressor - utilities
○	○	console-data	Keymaps, fonts, charset maps, fallback tables for console-tools
○	○	console-tools	Linux console and font utilities.
○	○	console-tools-libs	Shared libraries for Linux console and font manipulation.
○	○	cpio	GNU cpio -- a program to manage archives of files.
－	○	cpp	The GNU C preprocessor.
○	○	cron	management of regular background processing
○	○	dc	The GNU dc arbitrary precision reverse-polish calculator
○	○	debconf-tiny	Tiny subset of debconf for the base system

—	○	debhelper	helper programs for debian/rules
○	○	debianutils	Miscellaneous utilities specific to Debian.
—	○	devscripts	Scripts to make the life of a Debian Package maintainer easier
—	○	dh-make	Debianizing Tool for debhelper
○	○	diff	File comparison utilities
○	○	dpkg	Package maintenance system for Debian
—	○	dpkg-dev	Package building tools for Debian
○	○	e2fsprogs	The EXT2 file system utilities and libraries.
○	○	elvis-tiny	Tiny vi compatible editor for the base system.
○	○	exim	Exim Mailer
—	○	fakeroot	Gives a fake root environment.
○	○	fbset	Framebuffer device maintenance program.
○	○	fdutils	Linux floppy utilities
—	○	file	Determines file type using "magic" numbers
○	○	fileutils	GNU file management utilities.
○	○	findutils	utilities for finding files--find, xargs, and locate
○	○	ftp	The FTP client.
—	○	g++	The GNU C++ compiler.
—	○	gcc	The GNU C compiler.
—	○	gdb-arm	The GNU Debugger - ARM processor only
○	○	gettext-base	GNU Internationalization utilities for the base system
○	○	grep	GNU grep, egrep and fgrep.
○	○	groff	GNU troff text-formatting system.
○	○	gzip	The GNU compression utility.
○	○	hostname	A utility to set/show the host name or domain name
○	○	info	Standalone GNU Info documentation browser
—	○	jove	This is Jonathan's Own Version of Emacs (jove), a small and powerful editor.
○	○	ldso	The Linux dynamic linker, library and utilities.
—	○	less	A file pager program, similar to more(1)
—	○	libbz2	A high-quality block-sorting file compressor library - runtime
○	○	libc6	GNU C Library: Shared libraries and Timezone data

—	○	libc6-dbg	GNU C Library: Libraries with debugging symbols
—	○	libc6-dev	GNU C Library: Development Libraries and Header Files.
○	○	libdb2	The Berkeley database routines (run-time files).
○	○	libgdbmg1	GNU dbm database routines (runtime version). [libc6 version]
○	○	libgpmg1	General Purpose Mouse Library [libc6]
○	○	libident	simple RFC1413 client library - runtime
○	○	liblockfile1	Shared library with NFS-safe locking functions.
○	○	libncurses4	Shared libraries for terminal handling
○	○	libncurses5	Shared libraries for terminal handling
—	○	libncurses5-dev	Developer's libraries and docs for ncurses
○	○	libnewt0	Not Erik's Windowing Toolkit - text mode windowing with slang
○	○	libopenldap-runtime	OpenLDAP runtime files for libopenldap
○	○	libopenldap1	OpenLDAP libraries.
○	○	libpam-modules	Pluggable Authentication Modules for PAM
○	○	libpam-runtime	Runtime support for the PAM library
○	○	libpam0g	Pluggable Authentication Modules library
○	○	libpcre2	Philip Hazel's Perl Compatible Regular Expression library
—	○	libpopt-dev	lib for parsing cmdline parameters - development files
○	○	libpopt0	lib for parsing cmdline parameters
○	○	libreadline4	GNU readline and history libraries, run-time libraries.
○	○	libstdc++2.10	The GNU stdc++ library
—	○	libstdc++2.10-dbg	The GNU stdc++ library (debugging files)
—	○	libstdc++2.10-dev	The GNU stdc++ library (development files)
○	○	libwrap0	Wietse Venema's TCP wrappers library
○	○	locales	GNU C Library: National Language (locale) data [binary]
○	○	lockfile-progs	Programs for locking and unlocking files and mailboxes.
○	○	login	System login tools
○	○	logrotate	Log rotation utility
—	○	lrzsz	Tools for zmodem/xmodem/ymodem file transfer
○	○	mailx	A simple mail user agent.
—	○	make	The GNU version of the "make" utility.
○	○	makedev	Creates special device files in /dev.

○	○	man-db	Display the on-line manual.
○	○	manpages	Man pages about using a Linux system.
○	○	mawk	a pattern scanning and text processing language
○	○	modconf	Device Driver Configuration
○	○	modutils	Linux module utilities.
○	○	mount	Tools for mounting and manipulating filesystems.
○	○	ncurses-base	Descriptions of common terminal types
○	○	ncurses-bin	Terminal-related programs and man pages
○	○	netbase	Basic TCP/IP networking binaries
○	○	nvi	4.4BSD re-implementation of vi.
○	○	nvi	Change and administer password and group data.
—	○	patch	Apply a diff file to an original
○	○	perl-5.005	Larry Wall's Practical Extracting and Report Language.
○	○	perl-5.005-base	The Pathologically Eclectic Rubbish Lister
○	○	perl-base	Fake package assuring that one of the -base package is installed
○	○	ppp	Point-to-Point Protocol (PPP) daemon.
○	○	pppconfig	A text menu based utility for configuring ppp.
○	○	procps	The /proc file system utilities.
○	○	pump	Simple DHCP/BOOTP client for 2.2.x kernels
○	○	sed	The GNU sed stream editor.
○	○	setserialsed	Controls configuration of serial ports.
○	○	shellutils	The GNU shell programming utilities.
○	○	slang1	The S-Lang programming library - runtime version.
—	○	slang1-dev	The S-Lang programming library, development version.
○	○	sysklogd	Kernel and system logging daemons
○	○	sysvinit	System-V like init.
○	○	tar	GNU tar
○	○	tasksel	New task packages selector
○	○	tcpd	Wietse Venema's TCP wrapper utilities
○	○	telnet	The telnet client.
○	○	textutils	The GNU text file processing utilities.
○	○	update	daemon to periodically flush filesystem buffers.
○	○	util-linux	Miscellaneous system utilities.
○	○	whiptail	Displays user-friendly dialog boxes from shell scripts.
○	○	whois	whois client

5 注意事項

5.1 ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェア(付属のドキュメントも含みます)は、現状のまま(AS IS)提供されるものであり、特定の目的に適合することや、その信頼性、正確性を保証するものではありません。また、本製品の使用による結果についてもなんら保証するものではありません。

Armadillo[HT-1070] software manual　　2003 年 7 月 28 日　　rev.1.13

梅澤無線電機株式会社

東京営業部
101-0044 東京都千代田区鍛冶町 2-8-12 吉川ビル 2F　TEL03-3256-4491 FAX03-3256-4494
仙台営業所
982-0012 仙台市太白区長町南４丁目 25-5　TEL022-304-3880 FAX022-304-3882
札幌営業所
060-0062 札幌市中央区南 2 条西 7 丁目　TEL011-251-2992 FAX011-281-2515

本製品・資料についての技術的なお問い合わせは技術推進部直通ダイヤル(TEL/FAX)へ
０１２０－０２４７６８

株式会社アットマークテクノ

004-0062 札幌市厚別区厚別西２条２丁目 3-14 SD ビル 2F TEL011-890-6551 FAX011-890-6552