μST-SH2
μST-ADIO の使用方法

3.1 版　2023 年 10 月 02 日

目次

1. 概要

本アプリケーションノートはA/D、I/O拡張オプションボード『**μST-ADIO**』をμST-SH2用Linuxで使用する方法について述べます。
μST-ADIOは、A/D変換4chと絶縁型I/O入出力を各3ch搭載したA/D、I/O拡張オプションボードです。
μST-SH2とμST-ADIOを組み合わせることにより、アナログセンサとの接続やリレー制御などの用途でご利用いただけます。
本アプリケーションノートでは、μST-SH2用Linuxを使用して、μST-ADIOのサンプルプログラムを動作させる方法について説明します。

本アプリケーションノートを実行するには、『**μST-SH2　Linux開発キット**』がインストールされている必要があります。

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。

2. μST-ADIO について

2.1　μST-ADIO の概要

μST-ADIO は『**μST-SH2**』に A/D 変換機能と I/O 機能を拡張するオプションボードです。分解能 10 ビット、シングルエンド入力 4ch、差動入力 2ch での A/D 変換が可能です。また、I/O は入出力各 3ch で絶縁されています。
μST-SH2 用 Linux にはμST-ADIO 用 A/D デバイスドライバと I/O デバイスドライバが組み込まれており、Linux のアプリケーションプログラムからμST-ADIO にアクセスすることができます。

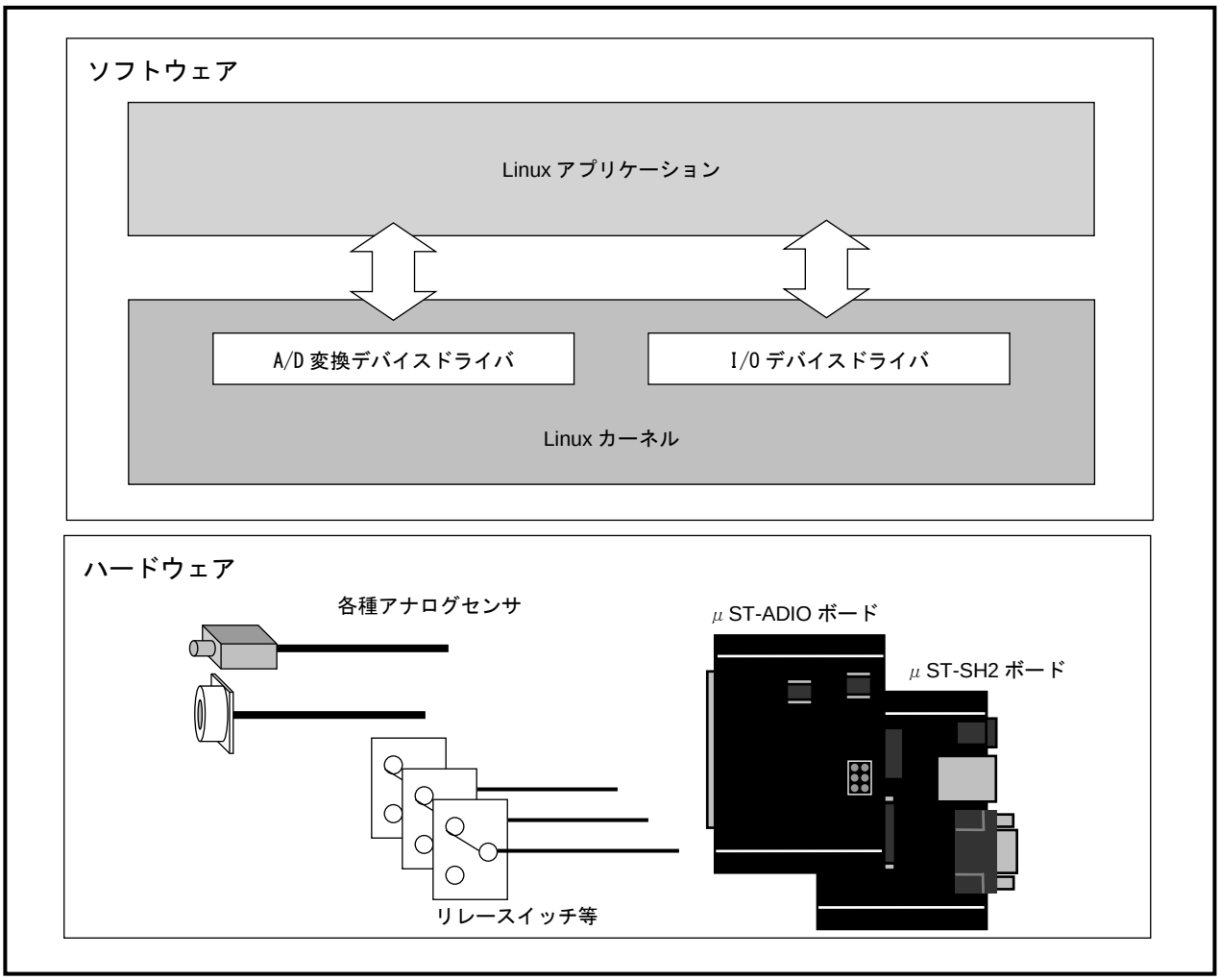

Fig 2.1-1 μST-ADIO の概要

2.2　μST-ADIO の接続

μST-ADIO はμST-SH2 と組み合わせることにより、A/D データの取得や I/O 制御が可能になります。
下図にμST-ADIO およびμST-SH2 を使用したときの接続例を示します。

※　テストボードは製品に付属していません。用途に適したテストボードを各自ご用意ください。

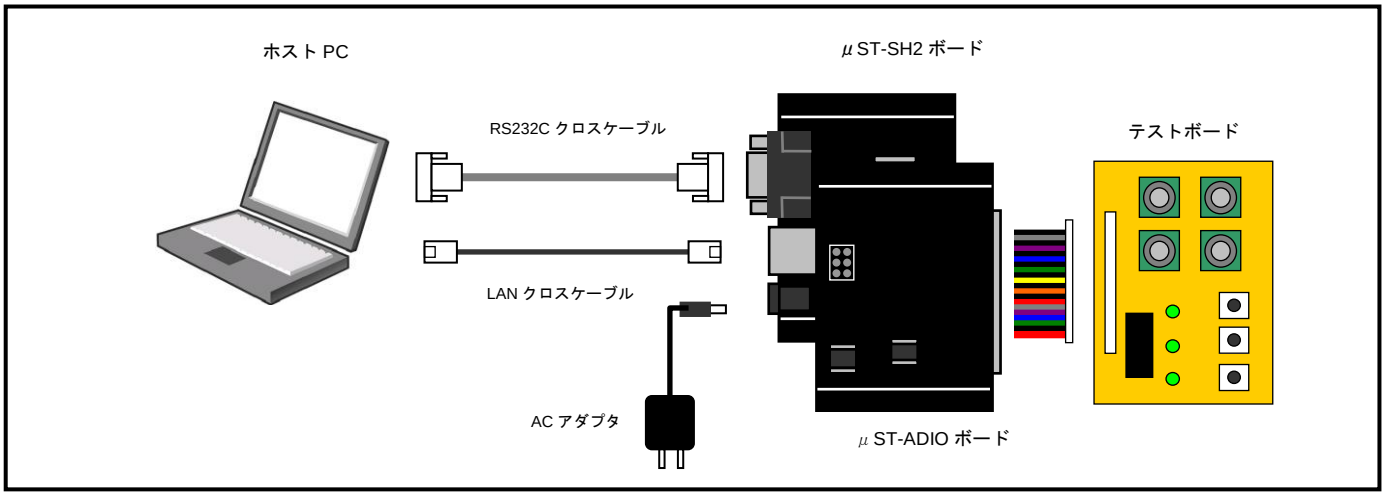

Fig 2.2-1　μST-ADIO ボードの接続（PC に接続する場合）

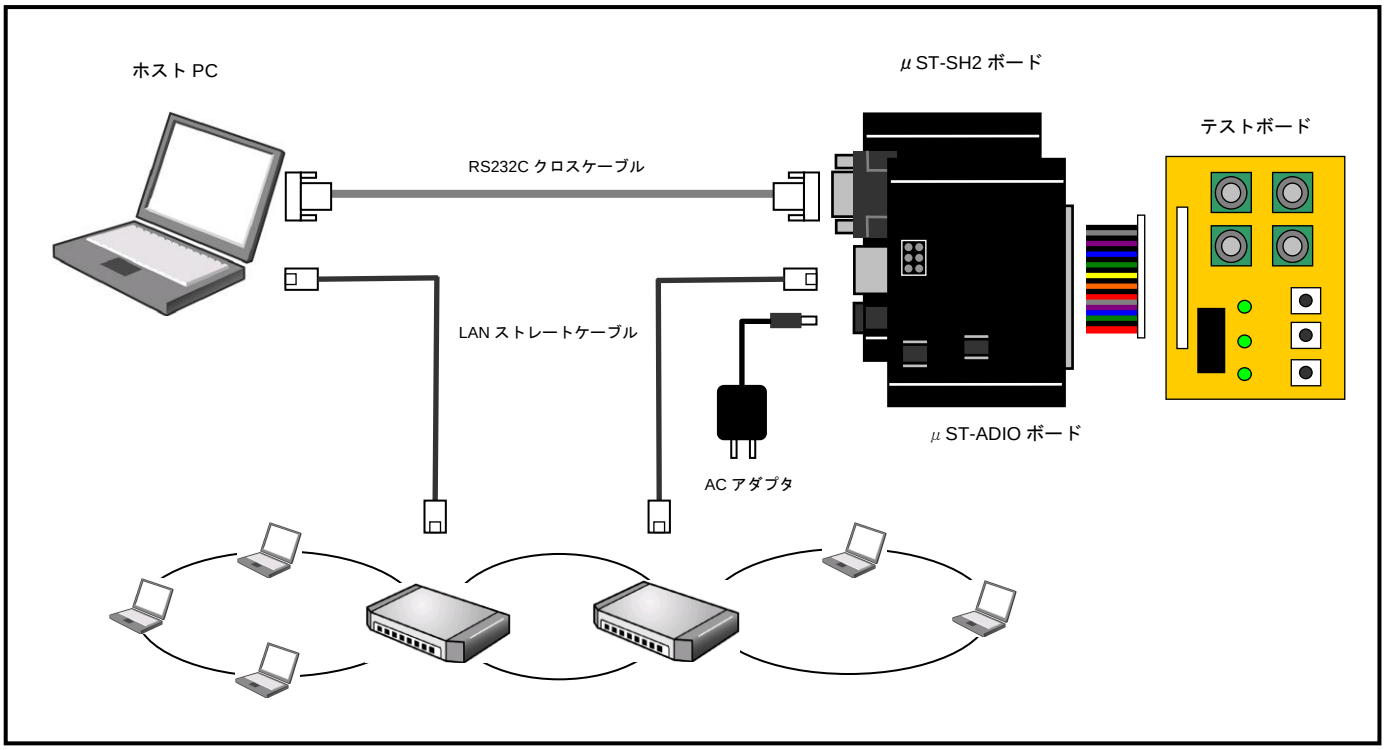

Fig 2.2-2　μST-ADIO ボードの接続（HUB に接続する場合）

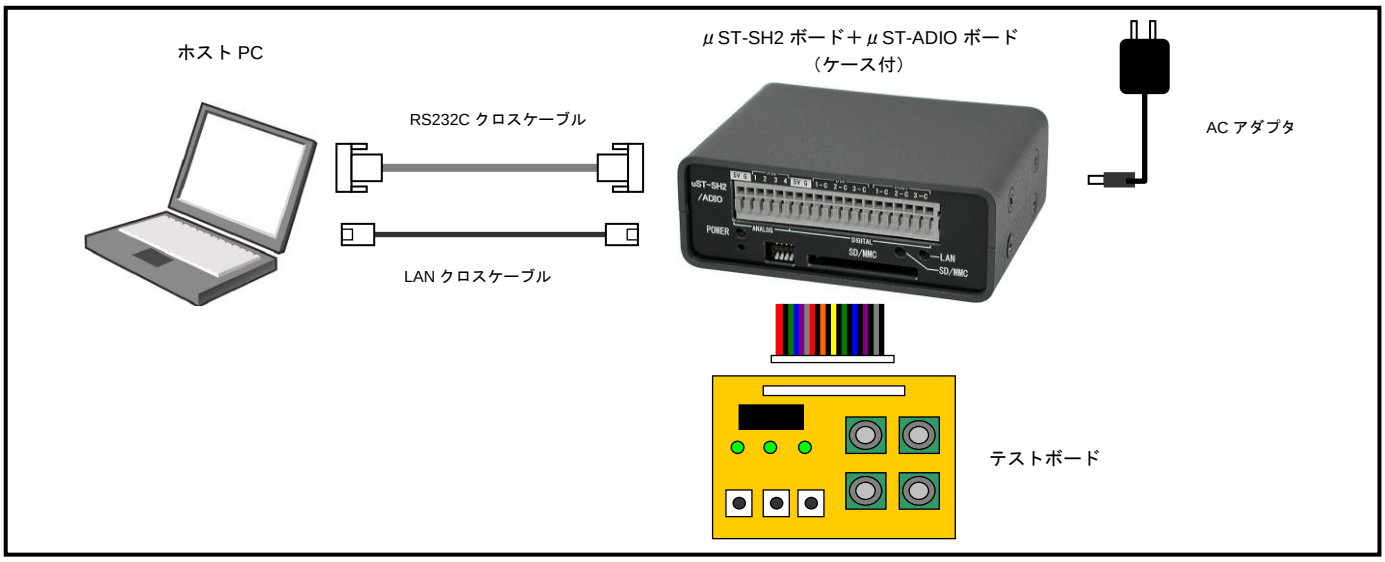

Fig 2.2-3　μST-ADIO ボードの接続（PC に接続する場合：ケース付）

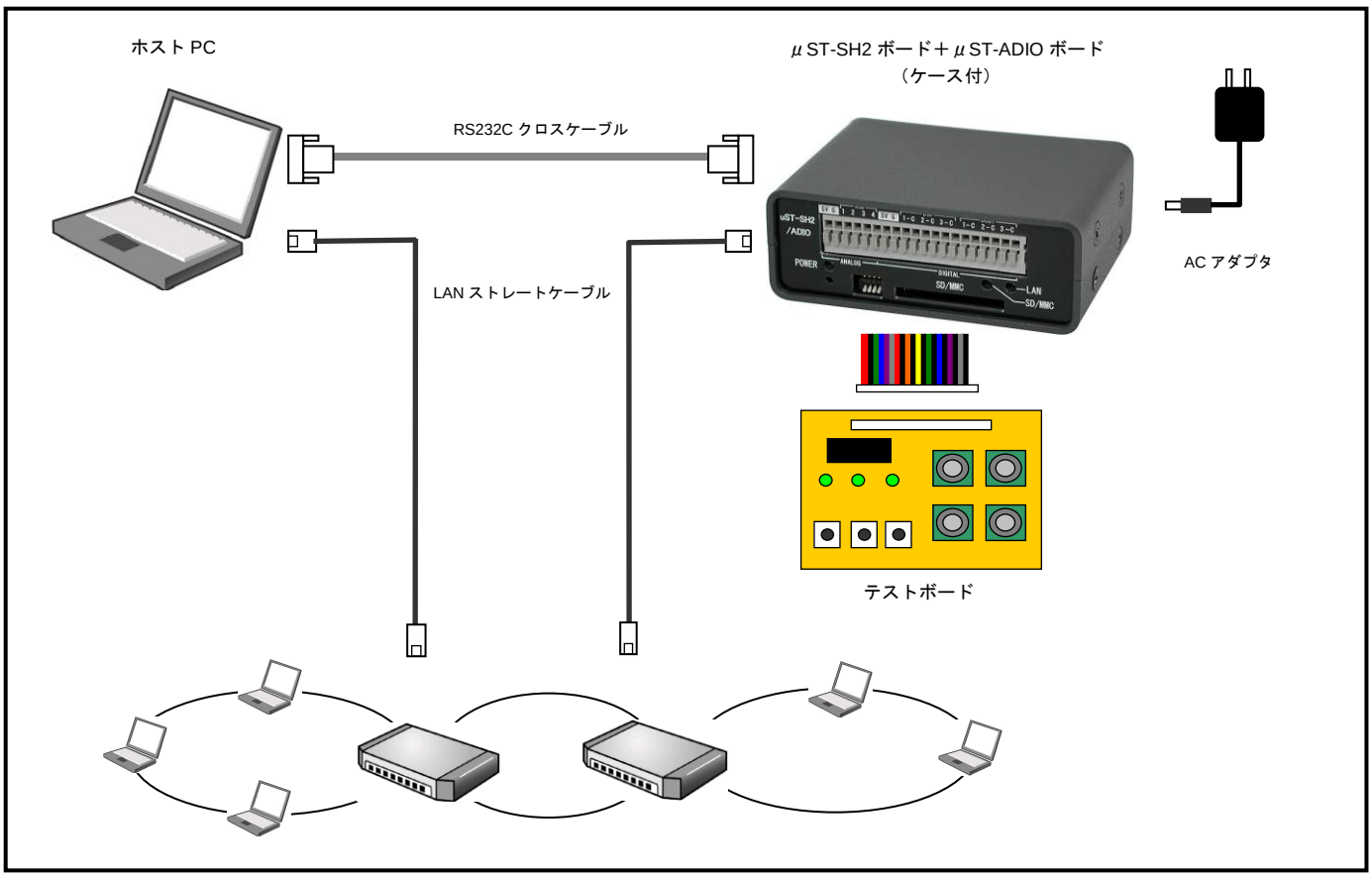

Fig 2.2-4　μST-ADIO ボードの接続（HUB に接続する場合：ケース付）

2.3　デバイスドライバ

μST-ADIO デバイスドライバ概要

μST-ADIO のデバイスドライバは A/D 変換デバイスドライバと I/O デバイスドライバで構成されています。
A/D 変換デバイスドライバと I/O デバイスドライバはキャラクタ型デバイスドライバです。キャラクタ型デバイスドライバはシリアルポートなどシーケンシャルにデータにアクセスするデバイスになります。
A/D 変換デバイスドライバと I/O デバイスドライバのソースファイルは『**ustadio.c**』になります。『**ustadio.c**』は Linux カーネルの一部としてコンパイルします。

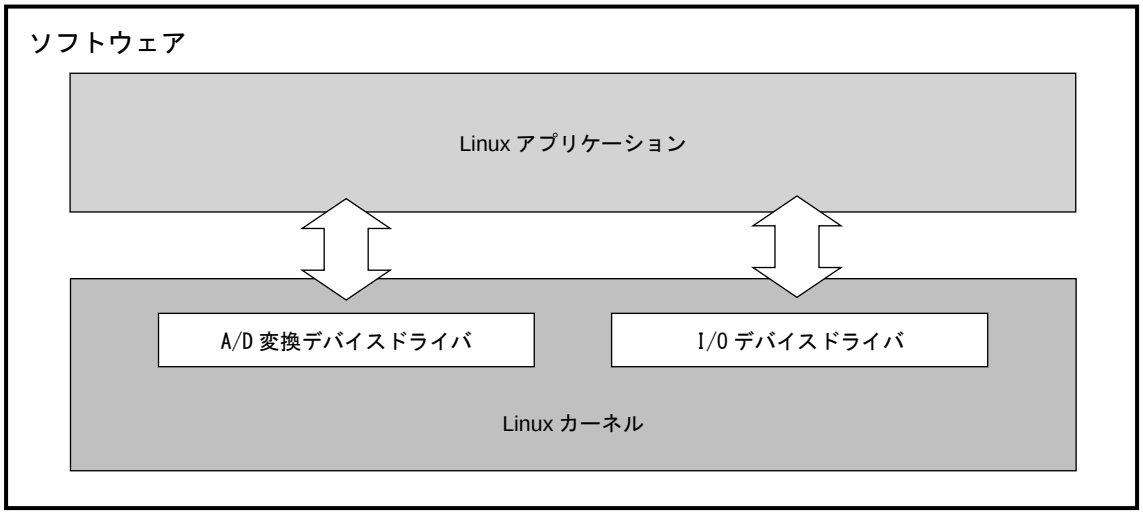

Fig 2.3-1 μST-ADIO デバイスドライバ概要

```
Linux カーネルソースディレクトリ
|--include
| |--linux
|   |--ustadio.h                :   μST-ADIO デバイスドライバヘッダファイル
                                      ・
                                      ・
|--drivers
  |--char
    |--ustadio.c                :   μST-ADIO デバイスドライバソースファイル
                                      ・
                                      ・
```

Fig 2.3-2　μST-ADIO デバイスドライバソースファイル構成

A/D 変換デバイスドライバ概要

A/D 変換デバイスドライバが提供するシステムコール（API）は『**open**』、『**close**』、『**ioctl**』、『**read**』になります。
A/D 変換デバイスを示すデバイスファイルは『**/dev/ad**』、メジャー番号とマイナー番号は『**10**』と『**101**』になります。

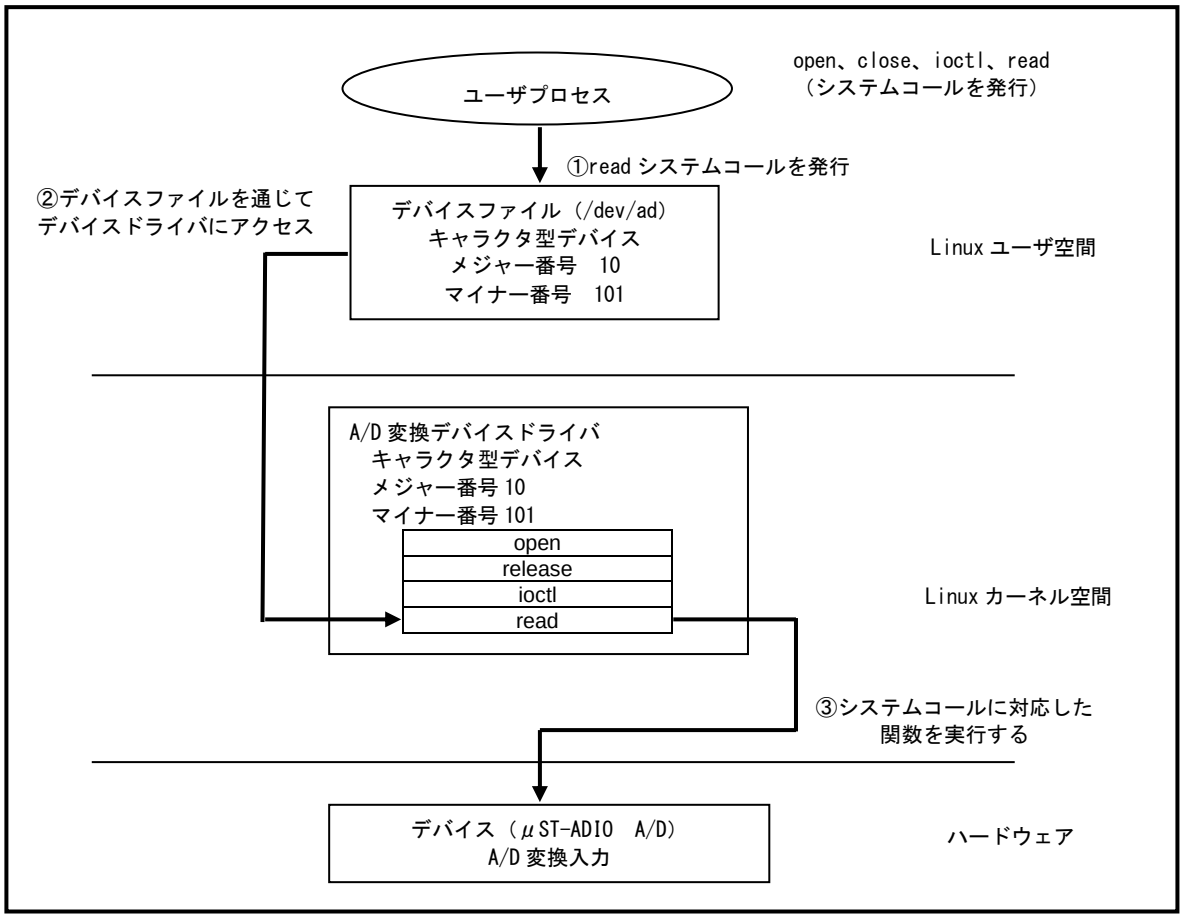

Fig 2.3-3　A/D 変換デバイスドライバの概要

A/D 変換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。

●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	ファイルディスクリプタを返す エラー時は-1 を返す
備考	論理デバイス名/dev/ad を使用 フラグは O_RDONLY を使用

●close システムコール

機能	デバイスをリリースする
書式	int close(int fd)
引数	fd　　：ファイルディスクリプタ
戻り値	リリース成功時には 0、エラー時は-1 を返す
備考	

●ioctl システムコール（AD_CONTROL）

機能	A/D 変換入力モードの設定
書式	int ioctl(int fd, AD_CONTROL, unsinged long arg)
引数	fd　　：ファイルディスクリプタ arg　　：A/D 変換入力モード設定値
戻り値	リリース成功時には 0、エラー時は-1 を返す
備考	第二引き数はコマンド A/D 変換入力モード設定については『**Table 2.3-4　A/D 変換入力モード設定**』を参照

●read システムコール

機能	A/D 変換の値を読込む
書式	ssize_t read(int fd, void *buf, size_t count)
引数	fd　　：ファイルディスクリプタ buf　　：バッファ count　：データ長
戻り値	読み込んだバイト数
備考	データ長は 2 バイト（1 ワード）を指定 データ形式は A/D 変換入力モードに依存

Table 2.3-4　A/D 変換入力モード設定

設定値	符号	分解能	チャンネル
AD_U10_IN1	なし	10 ビット	AIN1
AD_U10_IN2	なし	10 ビット	AIN2
AD_U10_IN3	なし	10 ビット	AIN3
AD_U10_IN4	なし	10 ビット	AIN4
AD_U10_IN12	なし	10 ビット	AIN1-AIN2 差動
AD_U10_IN21	なし	10 ビット	AIN2-AIN1 差動
AD_U10_IN34	なし	10 ビット	AIN3-AIN4 差動
AD_U10_IN43	なし	10 ビット	AIN4-AIN3 差動
AD_U10_IN14	なし	10 ビット	AIN1-AIN4 差動
AD_U10_IN24	なし	10 ビット	AIN2-AIN4 差動
AD_U8_IN1	なし	8 ビット	AIN1
AD_U8_IN2	なし	8 ビット	AIN2
AD_U8_IN3	なし	8 ビット	AIN3
AD_U8_IN4	なし	8 ビット	AIN4
AD_U8_IN12	なし	8 ビット	AIN1-AIN2 差動
AD_U8_IN21	なし	8 ビット	AIN2-AIN1 差動
AD_U8_IN34	なし	8 ビット	AIN3-AIN4 差動
AD_U8_IN43	なし	8 ビット	AIN4-AIN3 差動
AD_U8_IN14	なし	8 ビット	AIN1-AIN4 差動
AD_U8_IN24	なし	8 ビット	AIN2-AIN4 差動

設定値	符号	分解能	チャンネル
AD_S10_IN1	あり	10 ビット	AIN1
AD_S10_IN2	あり	10 ビット	AIN2
AD_S10_IN3	あり	10 ビット	AIN3
AD_S10_IN4	あり	10 ビット	AIN4
AD_S10_IN12	あり	10 ビット	AIN1-AIN2 差動
AD_S10_IN21	あり	10 ビット	AIN2-AIN1 差動
AD_S10_IN34	あり	10 ビット	AIN3-AIN4 差動
AD_S10_IN43	あり	10 ビット	AIN4-AIN3 差動
AD_S10_IN14	あり	10 ビット	AIN1-AIN4 差動
AD_S10_IN24	あり	10 ビット	AIN2-AIN4 差動
AD_S8_IN1	あり	8 ビット	AIN1
AD_S8_IN2	あり	8 ビット	AIN2
AD_S8_IN3	あり	8 ビット	AIN3
AD_S8_IN4	あり	8 ビット	AIN4
AD_S8_IN12	あり	8 ビット	AIN1-AIN2 差動
AD_S8_IN21	あり	8 ビット	AIN2-AIN1 差動
AD_S8_IN34	あり	8 ビット	AIN3-AIN4 差動
AD_S8_IN43	あり	8 ビット	AIN4-AIN3 差動
AD_S8_IN14	あり	8 ビット	AIN1-AIN4 差動
AD_S8_IN24	あり	8 ビット	AIN2-AIN4 差動

I/O デバイスドライバ概要

I/O デバイスドライバが提供するシステムコール（API）は『**open**』、『**close**』、『**ioctl**』になります。
I/O デバイス入力 I/O ポートにデータ（1 が入力）が届くと割込みが発生します。I/O デバイスドライバは割込みが発生すると I/O シグナルを発行します。
I/O デバイスを示すデバイスファイルは『**/dev/dio**』、メジャー番号とマイナー番号は『**10**』と『**100**』になります。

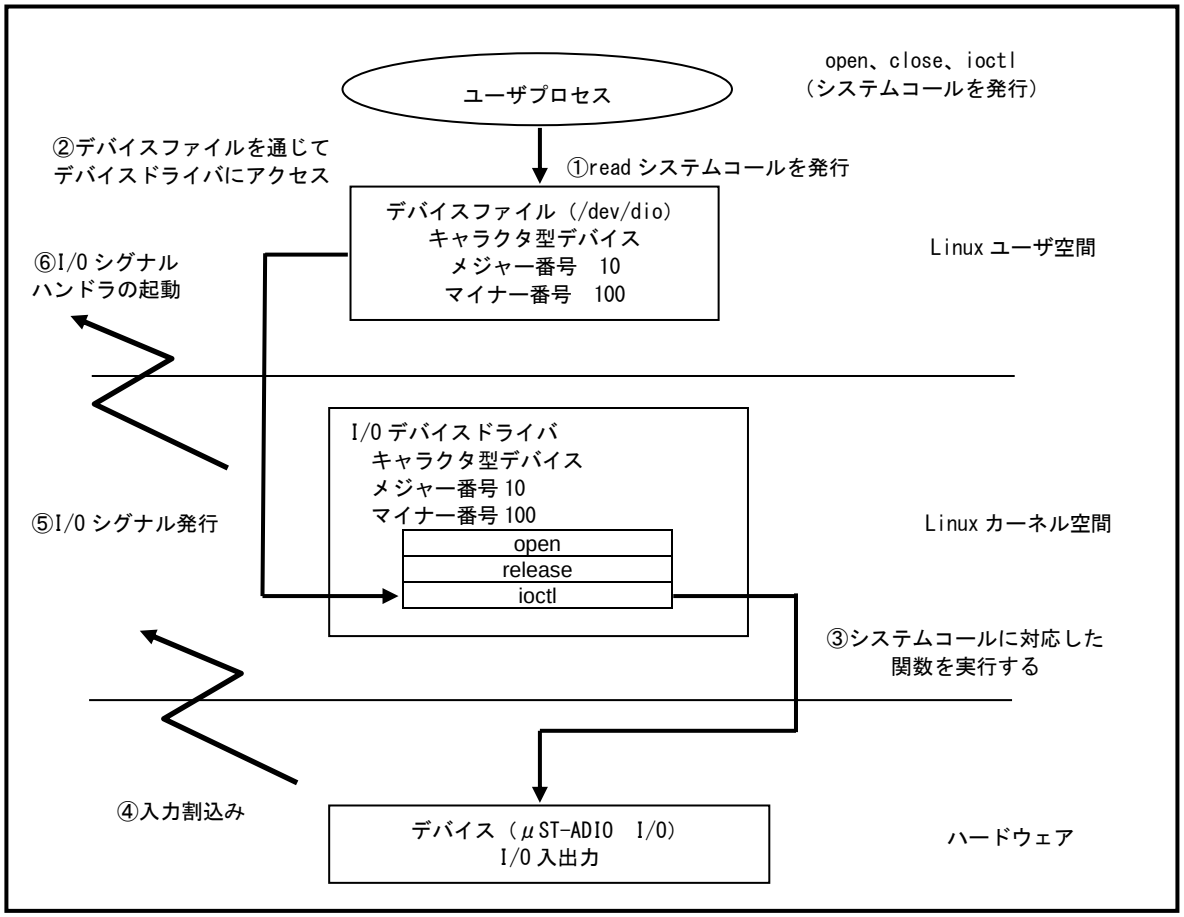

Fig 2.3-5　I/O デバイスドライバの概要

I/O 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。

●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	ファイルディスクリプタを返す エラー時は-1 を返す
備考	論理デバイス名/dev/dio を使用 フラグは O_RDWR を使用

●close システムコール

機能	デバイスをリリースする
書式	int close(int fd)
引数	fd　　：ファイルディスクリプタ
戻り値	リリース成功時には 0、エラー時は-1 を返す
備考	

●ioctl システムコール（DIO_INPUT）

機能	I/O ポートの値を取得する
書式	int ioctl(int fd, DIO_INPUT, int *arg)
引数	fd　　：ファイルディスクリプタ arg　　：I/O ポートの値
戻り値	成功時には 0、エラー時は-EINVAL を返す
備考	第二引き数はコマンド データ長は 1 バイトを指定

ビット	7	6	5	4	3	2	1	0
データ	－	－	－	－	－	DIN3	DIN2	DIN1

●ioctl システムコール（DIO_OUTPUT）

機能	I/O ポートに値を出力する
書式	int ioctl(int fd, DIO_OUTPUT, int *arg)
引数	fd　　：ファイルディスクリプタ arg　　：I/O ポートの値
戻り値	成功時には 0、エラー時は-EINVAL を返す
備考	第二引き数はコマンド データ長は 1 バイトを指定

ビット	7	6	5	4	3	2	1	0
データ	－	－	－	－	－	DOUT3	DOUT2	DOUT1

3. サンプルプログラム

μST-ADIO とμST-SH2 上で動作するサンプルプログラムを作成する手順について説明します。

3.1　サンプルプログラム概要

μST-SH2 用 Linux 上で動作するμST-ADIO 用サンプルプログラムを作成します。
μST-ADIO 用サンプルプログラムはμST-ADIO デバイスドライバ『**ustadio.c**』を使用します。A/D 変換の入力値を表示する『**sample_ad.c**』と I/O の入出力を行う『**sample_dio.c**』があります。
『**sample_ad.c**』は A/D 変換のシングルエンド入力にて１秒毎に各チャンネルの値を取得し、コンソールに出力します。
『**sample_dio.c**』は１秒毎に出力 I/O ポートのオン・オフを行います。また、入力 I/O ポートに“1”が入力されると入力値をコンソールに出力します。
μST-ADIO デバイスドライバは『**/home/guest/linuxkit-ustsh2/linux-2.6.33.1-ustsh2/drivers/char**』ディレクトリ、
サンプルプログラムは『**/home/guest/linuxkit-ustsh2/samples/ustadio-sample**』ディレクトリにあります。
サンプルプログラムのコンパイルを『**make**』で行うことができますが、ここではコンパイル方法を示すためコマンドライン上からコンパイルを行います。

3.2　サンプルプログラムのコンパイル

A/D 変換

A/D 変換サンプルプログラム『**sample_ad.c**』をゲスト OS 上でコンパイルします。
『**sample_ad.c**』は A/D 変換のシングルエンド入力にて 1 秒毎に各チャンネルの値を取得し、コンソールに出力します。
取得する値は、符号なしの分解能 10 ビットになります。

※　ゲスト OS につきましては『Linux 開発キットソフトウェアマニュアル Linux 編』及び『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。

●sample_ad.c ファイル

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/ustadio.h>

int main(int argc, char *argv[])
{
	int fd;
	int count;
	int ret;
	short data1, data2, data3, data4;
	unsigned char control;

	/* デバイスファイルのオープン */
	fd = open( "/dev/ad", O_RDONLY );
	if( fd < 0 ) {
		printf("Error open A/D device¥n");
		exit( 0 );
	}

	while(1) {
		/* A/D 変換入力モードの設定 */
		/* 符号なし、分解能 10 ビット、チャンネル AIN1 */
		control = AD_U10_IN1;
		ret = ioctl( fd, AD_CONTROL, control );
		if( ret < 0 ) {
			printf("Error ioctl A/D control¥n");
			break;
		}

		/* A/D 変換値の取得 */
		count = read( fd, &data1, sizeof(data1) );
		if( count != sizeof(short) ) {
			printf("Error read A/D¥n");
			break;
		}

		/* A/D 変換入力モードの設定 */
		/* 符号なし、分解能 10 ビット、チャンネル AIN2 */
		control = AD_U10_IN2;
		ret = ioctl( fd, AD_CONTROL, control );
		if( ret < 0 ) {
			printf("Error ioctl A/D control¥n");
			break;
		}

		/* A/D 変換値の取得 */
		count = read( fd, &data2, sizeof(data2) );
		if( count != sizeof(short) ) {
```

```
                                printf("Error read A/D¥n");
                                break;
                        }
                        /* A/D 変換入力モードの設定 */
                        /* 符号なし、分解能 10 ビット、チャンネル AIN3 */
                        control = AD_U10_IN3;
                        ret = ioctl( fd, AD_CONTROL, control );
                        if( ret < 0 ) {
                                printf("Error ioctl A/D control¥n");
                                break;
                        }

                        /* A/D 変換値の取得 */
                        count = read( fd, &data3, sizeof(data3) );
                        if( count != sizeof(short) ) {
                                printf("Error read A/D¥n");
                                break;
                        }

                        /* A/D 変換入力モードの設定 */
                        /* 符号なし、分解能 10 ビット、チャンネル AIN4 */
                        control = AD_U10_IN4;
                        ret = ioctl( fd, AD_CONTROL, control );
                        if( ret < 0 ) {
                                printf("Error ioctl A/D control¥n");
                                break;
                        }

                        /* A/D 変換値の取得 */
                        count = read( fd, &data4, sizeof(data4) );
                        if( count != sizeof(short) ) {
                                printf("Error read A/D¥n");
                                break;
                        }

                        printf("A/D Data¥n");
                        printf(" AIN1 : %d¥n", data1);
                        printf(" AIN2 : %d¥n", data2);
                        printf(" AIN3 : %d¥n", data3);
                        printf(" AIN4 : %d¥n", data4);
                        printf("¥n");

                        sleep( 1 );

                }

                close( fd );

                return 0;
}
```

① ゲスト OS の端末を起動し、guest ユーザでログインします。

```
LinuxKit login: guest  入力
Password:******  入力      パスワード「guest」を入力します。
```

② アプリケーションプログラムのディレクトリに移動します。

『cd linuxkit-ustsh2/samples/ustadio-sample/ad』を実行してください。

```
[guest@LinuxKit ~]$ cd linuxkit-ustsh2/samples/ustadio-sample/ad  入力
```

③ ゲスト OS 上でアプリケーションプログラム『sample_ad.c』をコンパイルします。

『sh2eb-linux-gcc -elf2flt="-ar" -I/home/guest/linuxkit-ustsh2/linux-2.6.33.1-ustsh2/include -o sample_ad sample_ad.c』を改行を入れず 1 行で入力し、出来上がったプログラムに実行権を与えてください。

```
[guest@LinuxKit ad]$ sh2eb-linux-gcc -elf2flt="-ar"
-I/home/guest/linuxkit-ustsh2/linux-2.6.33.1-ustsh2/include -o sample_ad sample_ad.c  入力
[guest@LinuxKit ad]$ chmod a+x sample_ad  入力
```

※　『-I/home/guest/linux...』の『-I』はアルファベット大文字の『アイ』です。

④ ゲスト OS 上でアプリケーションプログラムを NFS 共有ディレクトリ『/nfs』にコピーします。

```
[guest@LinuxKit ad]$ cp -a sample_ad /nfs  入力
```

I/O

I/O サンプルプログラム『**sample_dio.c**』をゲスト OS 上でコンパイルします。
『**sample_dio.c**』は１秒毎に出力 I/O ポートのオン・オフを行います。また、入力 I/O ポートに“1”が入力されるとシグナルハンドラが起動し、入力値をコンソールに出力します。

●sample_dio.c ファイル

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/ioctl.h>
#include <linux/ustadio.h>

int fd;    /* ファイルディスクプリタ */

/* シグナルハンドラ */
static void input_handler(int arg)
{
          int ret;
          unsigned char data;

          /* I/O ポート入力値の取得 */
          ret = ioctl( fd, DIO_INPUT, &data );
          if( ret < 0 ) {
                    printf("Error ioctl DIO input¥n");
                    exit( 0 );
          }

          printf("DIO input 0x%x¥n", data);
}

int main(int argc, char *argv[])
{
          int ret;
          int flags;
          unsigned char data;

          /* デバイスファイルのオープン */
          fd = open("/dev/dio", O_NONBLOCK);
          if( fd < 0 ) {
                    printf("Error open DIO device¥n");
                    exit( 0 );
          }

          /* シグナルハンドラの設定 */
          signal(SIGIO, &input_handler);
          fcntl(fd, F_SETOWN, getpid());
          flags = fcntl(fd, F_GETFL);
          fcntl(fd, F_SETFL, flags | FASYNC);

          data = 0x01;
          while(1) {
                    /* I/O ポートへの出力 */
                    ret = ioctl( fd, DIO_OUTPUT, data );
                    if( ret < 0 ) {
                              printf("Error ioctl DIO output¥n");
                              break;
                    }
                    /* 出力データの変更 */
                    data <<= 1;
                    if( data == 0x08 ) {
                              data = 0x01;
                    }
```

```
                    sleep(1);
            }
            close(fd);
            return 0;
}
```

① ゲスト OS を起動し、guest ユーザでログインします。

```
LinuxKit login: guest  入力
Password:******  入力      パスワード「guest」を入力します。
```

② アプリケーションプログラムのディレクトリに移動します。
『cd linuxkit-ustsh2/samples/ustadio-sample/dio』を実行してください。

```
[guest@LinuxKit ~]$ cd linuxkit-ustsh2/samples/ustadio-sample/dio  入力
```

③ ゲスト OS 上でアプリケーションプログラム『sample_dio.c』をコンパイルします。
『sh2eb-linux-gcc -elf2flt="-ar" -I/home/guest/linuxkit-ustsh2/linux-2.6.33.1-ustsh2/include -o sample_dio sample_dio.c 』を改行を入れず 1 行で入力し、出来上がったプログラムに実行権を与えてください。

```
[guest@LinuxKit ad]$ sh2eb-linux-gcc -elf2flt="-ar"
-I/home/guest/linuxkit-ustsh2/linux-2.6.33.1-ustsh2/include -o sample_dio sample_dio.c  入力
[guest@LinuxKit ad]$ chmod a+x sample_dio  入力
```

※　『-I/home/guest/linux...』の『-I』はアルファベット大文字の『アイ』です。

④ ゲスト OS 上でアプリケーションプログラムを NFS 共有ディレクトリ『/nfs』にコピーします。

```
[guest@LinuxKit ad]$ cp -a sample_dio /nfs
```

4. μST-ADIO の動作

μST-ADIO と μST-SH2 を使用して、サンプルプログラムを動作させる手順について説明します。

4.1　μST-ADIO の動作環境

μST-ADIO の動作を確認するためには μST-SH2 を含め以下の環境が必要です。

●ホスト PC

μST-SH2 用 Linux では PC をコンソール端末として使用しますので、Linux の起動を確認するためにはシリアルポートが使用可能な PC が必要となります。その際、ホスト 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

μST-SH2 本体に必要な電源は DC5V±5%です。AC アダプタを用意してください。

●シリアル

μST-SH2 と PC をシリアルで接続する場合、付属の RS232C クロスケーブル（RxD、TxD、RTS、CTS、GND）をご使用ください。

●LAN

μST-SH2 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-T/TX 対応（UTP カテゴリ 5 以上）ケーブルをご使用ください。

Table 4.1-1　μST-ADIO 動作環境

使用機器等	環　　境
ADIO オプションボード	μST-ADIO
Linux ボード	μST-SH2
ホスト PC	PC/AT 互換機
ホスト OS	Windows7/XP（WindowsXP　推奨）※1
ソフトウェア	ターミナルソフト
ドライブ	DVD-ROM 読み込み可能なドライブ
シリアルポート	1 ポート
LAN ポート	1 ポート
RS232C ケーブル	クロスケーブルを使用
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	DC5V±5%　1A 以上　（AC アダプタ）

※1　WindowsVista につきましては VMware Player のサポート対象外のため、動作環境の対応 OS に含まれておりません。

4.2　μST-ADIO の設定

μST-SH2 用 Linux のためにμST-ADIO ボードの設定を行います。

※　ケース付きμST-SH2＋μST-ADIO ボードは出荷時設定が以下の設定になっています。

①　μST-ADIO のチップセレクトを CS5B に設定します。
μST-ADIO の JSW1 スイッチを『**CS5B**』に選択します。

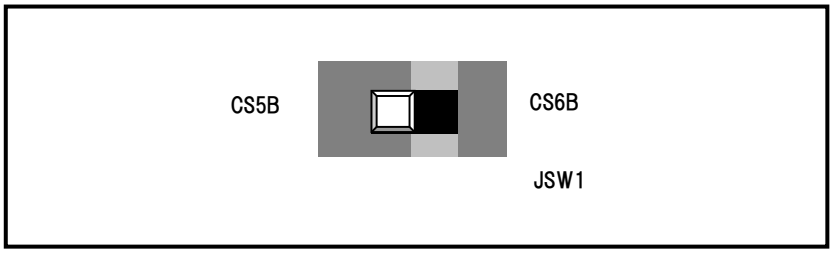

Fig 4.2-1　JSW1 の設定

②　μST-ADIO の A/D 変換入力切替をユニポーラに設定します。
μST-ADIO の SS1 スイッチを『**UNIP**』に選択します。

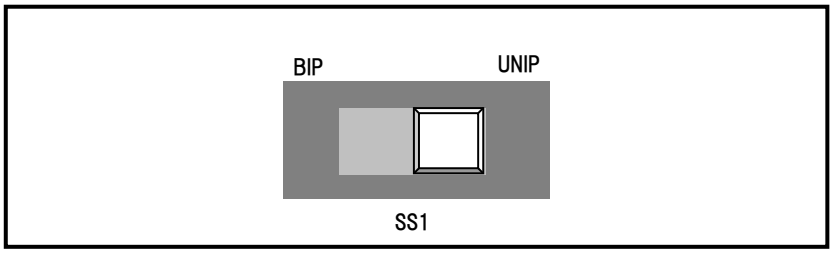

Fig 4.2-2　SS1 の設定

4.3　サンプルプログラムの動作

μST-SH2 用 Linux ではμST-ADIO のサンプルプログラムの動作方法について説明します。

μST-SH2 の IP アドレスは「192.168.128.200」、サブネットマスクは「255.255.255.0」と仮定します。
ゲスト OS の IP アドレスは「192.168.128.201」、サブネットマスクは「255.255.255.0」と仮定します。

μST-ADIO 対応 Linux カーネル

μST-ADIO を使用するには、μST-ADIO 対応 Linux カーネルを起動する必要があります。
μST-SH2 用 Linux カーネルはカーネルパラメータに『**ustadio**』を渡すことによりμST-ADIO 対応 Linux カーネルを起動することができます。
μST-SH2 では U-Boot の環境変数『**bootargs**』の値を設定することにより、カーネルパラメータを指定します。カーネルパラメータとして『**console=ttySC0,115200 ustadio**』を使用します。

環境変数『bootargs』の設定

```
=> setenv bootargs console=ttySC0,115200 ustadio
```

Table4.3-1　カーネルパラメータ

カーネルパラメータ	値	概要
console	ttyS0,115200	コンソールの指定を行います。 コンソールとして ttySC0 シリアルデバイスをボーレート 115200bps で使用することを示しています。
ustadio	-	μST-ADIO 対応 Linux カーネルを起動します。

① 『**2.2　μST-ADIO ボードの接続**』にしたがって、μST-ADIO とμST-SH2 を接続し、ホスト PC のシリアルポートとイーサネットポートを接続します。

② μST-SH2 の電源を投入しホスト PC のコンソール上で何らかのキーを入力します。
ブートローダの自動起動がキャンセルされ、コンソールの起動ログが表示されます。

```
U-Boot 2010.03 ustsh2-1.0 ( 6 月 16 2010 - 13:09:33)

CPU: SH2
BOARD: ALPHAPROJECT uST-SH2
DRAM:  32MB
FLASH: 8MB
In:    serial
Out:   serial
Err:   serial
Net:   sh_eth
Hit any key to stop autoboot:  0    入力    <-- 何かキーを入力する
=>
```

③ 環境変数『**bootargs**』の値を『**console=ttySC0,115200 ustadio**』と設定します。

```
=> setenv bootargs console=ttySC0,115200 ustadio    入力
=>
```

④　フラッシュ ROM から Linux システムを起動します。

『**bootm a0100000**』を実行してください。

```
=> bootm a0100000     入力
## Booting kernel from Legacy Image at a0100000 ...
   Image Name:   Linux-2.6.33.1
   Created:      2010-06-16   2:23:39 UTC
   Image Type:   SuperH Linux Kernel Image (gzip compressed)
   Data Size:    2552311 Bytes =  2.4 MB
   Load Address: 0c001000
   Entry Point:  0c002000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
以下省略・・

Welcome to Buildroot
buildroot login:
```

A/D 変換

A/D 変換サンプルプログラムを動作します。

① 『2.2　μST-ADIO ボードの接続』にしたがって、μST-ADIO とμST-SH2 を接続し、ホスト PC のシリアルポートとイーサネットポートを接続し、μST-ADIO 対応 Linux カーネルで Linux を起動します。

```
=> setenv bootargs console=ttySC0,115200 ustadio  入力
=> bootm a0100000  入力
## Booting kernel from Legacy Image at a0100000 ...
   Image Name:   Linux-2.6.33.1
   Created:      2010-06-16   2:23:39 UTC
   Image Type:   SuperH Linux Kernel Image (gzip compressed)
   Data Size:    2552311 Bytes =  2.4 MB
   Load Address: 0c001000
   Entry Point:  0c002000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
以下省略・・
```

② μST-SH2 上で Linux の起動を確認し、root 権限でログインします。

```
Welcome to Buildroot
buildroot login: root  入力
~ #
```

③ ゲスト OS を起動した状態にしておき、μST-SH2 からゲスト OS 上の NFS 共有ディレクトリにマウントします。

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

④ μST-SH2 上で作成した A/D 変換サンプルプログラムを NFS 共有ディレクトリからコピーします。

```
~ # cp -a /mnt/nfs/sample_ad .  入力
```

⑤ μST-SH2 上で NFS 共有ディレクトリをアンマウントします。

```
~ # umount /mnt/nfs  入力
```

⑥ μST-SH2 上でアプリケーションプログラム『sample_ad』を実行します。

```
~ # ./sample_ad  入力
```

⑦ コンソール上に 1 秒毎に A/D 変換値が出力されることを確認してください。

```
A/D Data
 AIN1 : 831
 AIN2 : 1016
 AIN3 : 7
 AIN4 : 96
```

⑧ プログラムを終了するには『Ctrl+c』を実行してください。

I/O

I/O サンプルプログラムを動作します。

① 『2.2　μST-ADIO ボードの接続』にしたがって、μST-ADIO とμST-SH2 を接続し、ホスト PC のシリアルポートとイーサネットポートを接続し、μST-ADIO 対応 Linux カーネルで Linux を起動します。

```
=> setenv bootargs console=ttySC0,115200 ustadio  入力
=> bootm a0100000  入力
## Booting kernel from Legacy Image at a0100000 ...
   Image Name:   Linux-2.6.33.1
   Created:      2010-06-16   2:23:39 UTC
   Image Type:   SuperH Linux Kernel Image (gzip compressed)
   Data Size:    2552311 Bytes =  2.4 MB
   Load Address: 0c001000
   Entry Point:  0c002000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
以下省略・・
```

② μST-SH2 上で Linux の起動を確認し、root 権限でログインします。

```
Welcome to Buildroot
buildroot login: root  入力
~ #
```

③ ゲスト OS を起動した状態にしておき、μST-SH2 からゲスト OS 上の NFS 共有ディレクトリにマウントします。

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

④ μST-SH2 上で作成した A/D 変換サンプルプログラムを NFS 共有ディレクトリからコピーします。

```
~ # cp -a /mnt/nfs/sample_dio .  入力
```

⑤ μST-SH2 上で NFS 共有ディレクトリをアンマウントします。

```
~ # umount /mnt/nfs  入力
```

⑥ μST-SH2 上でアプリケーションプログラム『sample_dio』を実行します。

```
~ # ./sample_dio  入力
```

⑦ μST-ADIO の I/O 出力ポートに 1 秒毎にデータが出力されます。また、I/O 入力ポートに“1”が入力されると I/O 入力ポートのデータがコンソールに出力されます。

```
DIO input 0x4
DIO input 0x2
DIO input 0x1
```

⑧ プログラムを終了するには『Ctrl+c』を実行してください。

5. 保障とサポート

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。
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。

Linux など付属する GPL ソフトウェアのソースコードは、ユーザ登録をすることにより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、これらの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。

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　https://www.apnet.co.jp

参考文献

「LINUX デバイスドライバ　第 3 版」　Alessandro rubini, Jonathan corbet, lGreg Krooah-Hartman 著
山崎康宏、山崎邦子、長原宏治、長原陽子 訳／オライリージャパン

「詳解 LINUX カーネル　第 3 版」　Daniel P. Bovet, Marco Cesati 著　高橋弘和 監訳
岡島順治朗、田宮まや、三浦広志　訳／オライリージャパン

その他　各社データシート

謝辞

Linux、SH-Linux 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
株式会社アルファプロジェクト
〒431-3114
静岡県浜松市中央区積志町 834
https://www.apnet.co.jp
E-Mail: query@apnet.co.jp