

μ ST-SH2

μ ST-VCP の使用方法

2.1 版 2023 年 10 月 02 日

目次

1. 概要	1
1.1 Linux について	1
2. μ ST-VCP について	2
2.1 μ ST-VCP の概要	2
2.2 μ ST-VCP の接続	3
2.3 デバイスドライバ	5
3. μ ST-VCP の動作	11
3.1 μ ST-VCP の動作環境	11
3.2 μ ST-VCP の設定	12
4. サンプルアプリケーションについて	13
4.1 サンプルアプリケーションの概要	13
4.2 サンプルアプリケーションの動作	14
5. サンプルプログラム	18
5.1 サンプルプログラム概要.....	18
5.2 サンプルプログラムのコンパイル	20
5.3 サンプルプログラムの動作	36
6. 保障とサポート	43

1. 概要

本アプリケーションノートは、ビデオキャプチャ拡張オプションボード『**μST-VCP**』をμST-SH2用Linuxで使用する方法について述べます。

μST-VCPは、ML86V7668(沖電気工業)デジタルビデオデコーダを搭載したビデオキャプチャ拡張オプションボードです。

μST-SH2とμST-VCPを組み合わせることで、ビデオキャプチャ用途に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-VCP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. μ ST-VCP について

2.1 μ ST-VCP の概要

μ ST-VCP は『 μ ST-SH2』にビデオキャプチャ機能を拡張するオプションボードです。 μ ST-SH2 用 Linux には μ ST-VCP 用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 μ ST-VCP にアクセスすることができます。

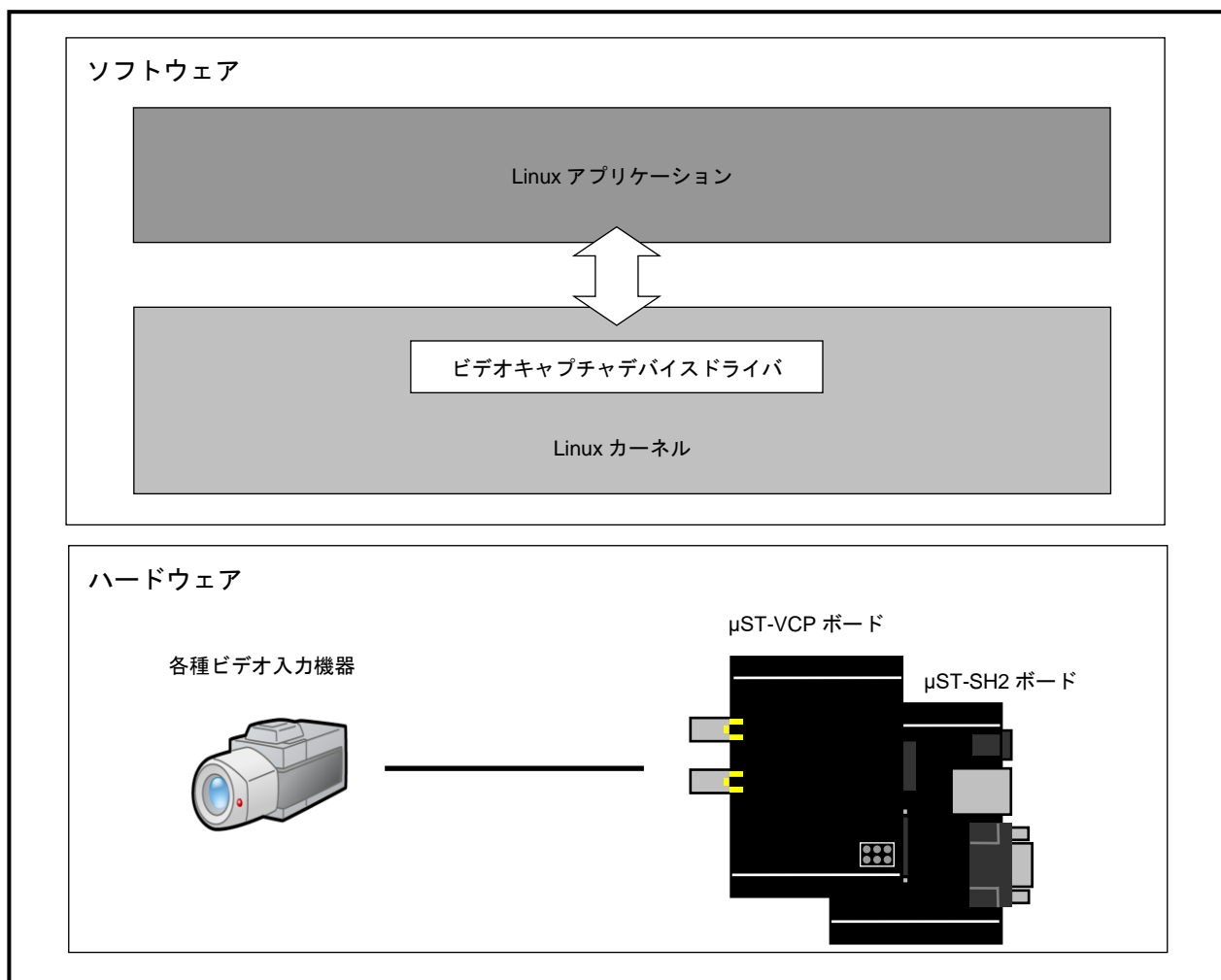


Fig 2.1-1 μ ST-VCP の概要

2.2 μ ST-VCP の接続

μ ST-VCP は μ ST-SH2 と組み合わせることにより、ビデオキャプチャが可能になります。
 下図に μ ST-VCP および μ ST-SH2 を使用したときの接続例を示します。

※ ビデオ入力機器及び RCA 端子 AV ケーブルは製品に付属していません。NTSC 出力を行えるビデオ入力機器を各自ご用意ください。

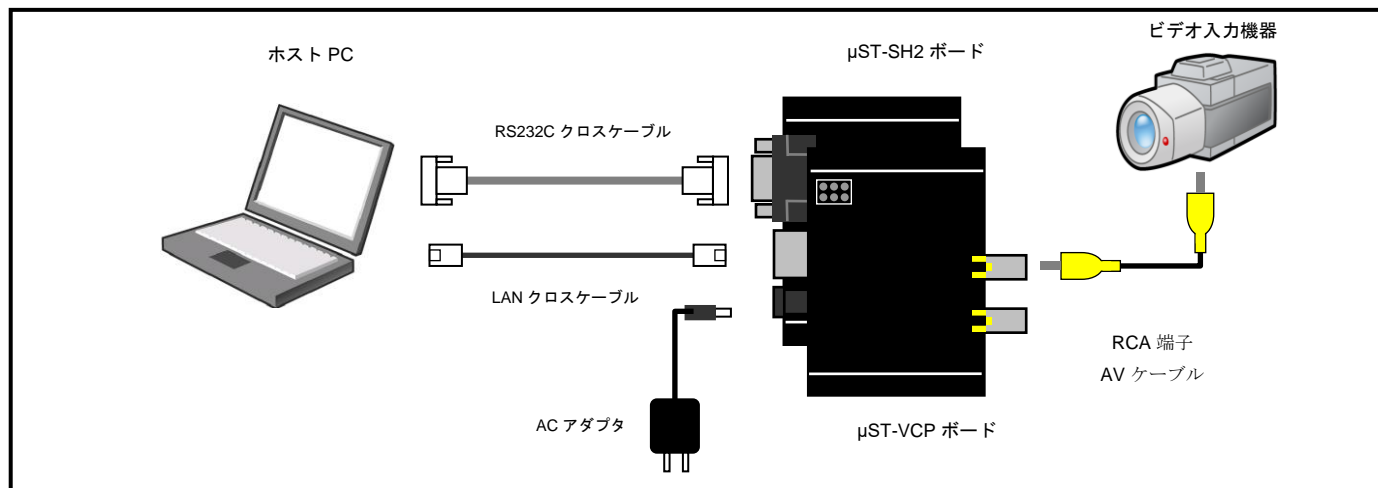


Fig 2.2-1 μ ST-VCP ボードの接続 (PC に接続する場合)

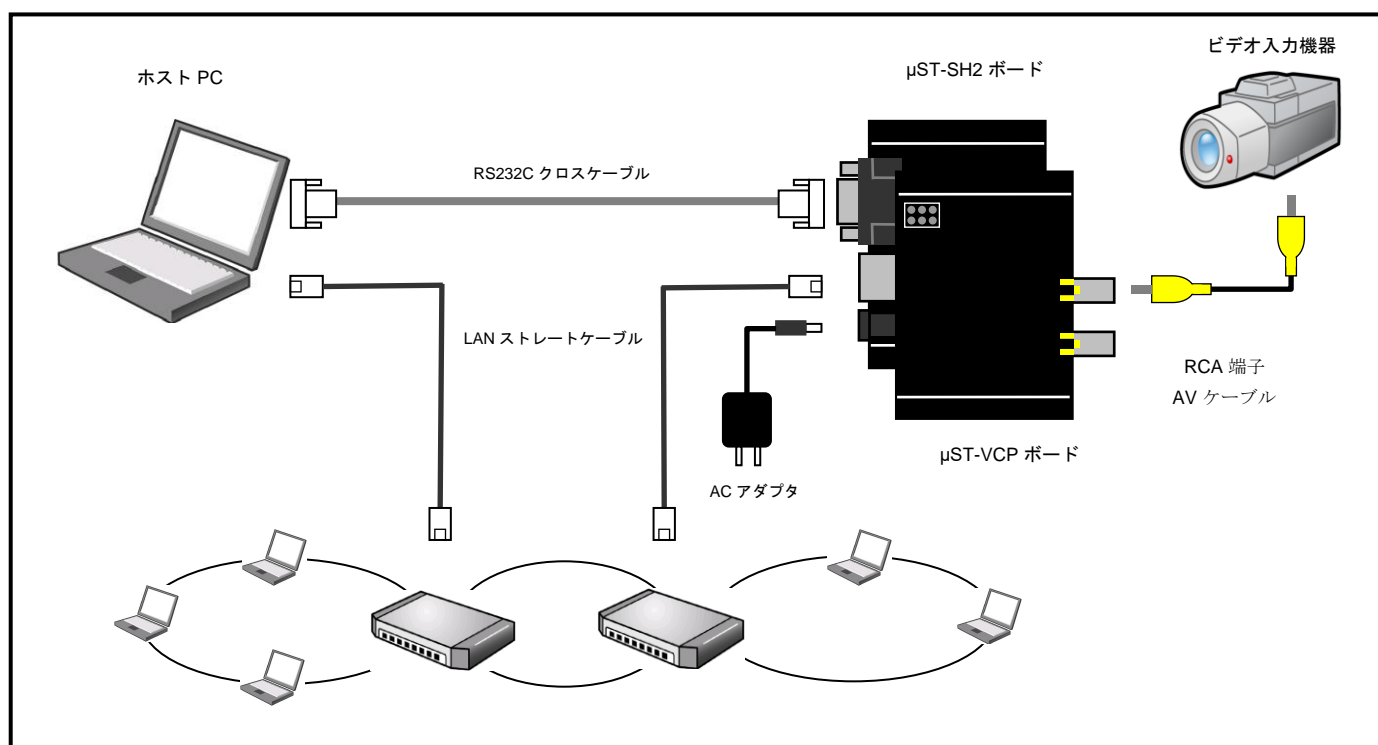


Fig 2.2-2 μ ST-VCP ボードの接続 (HUB に接続する場合)

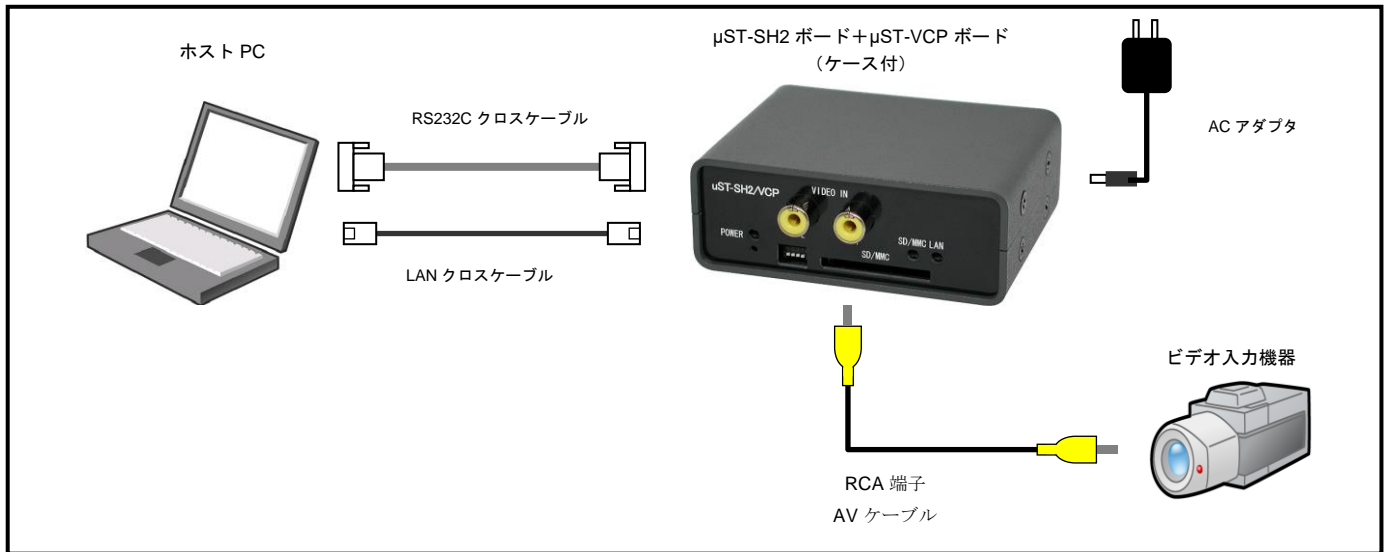


Fig 2.2-3 μST-VCP board connection (PC connection case: case included)

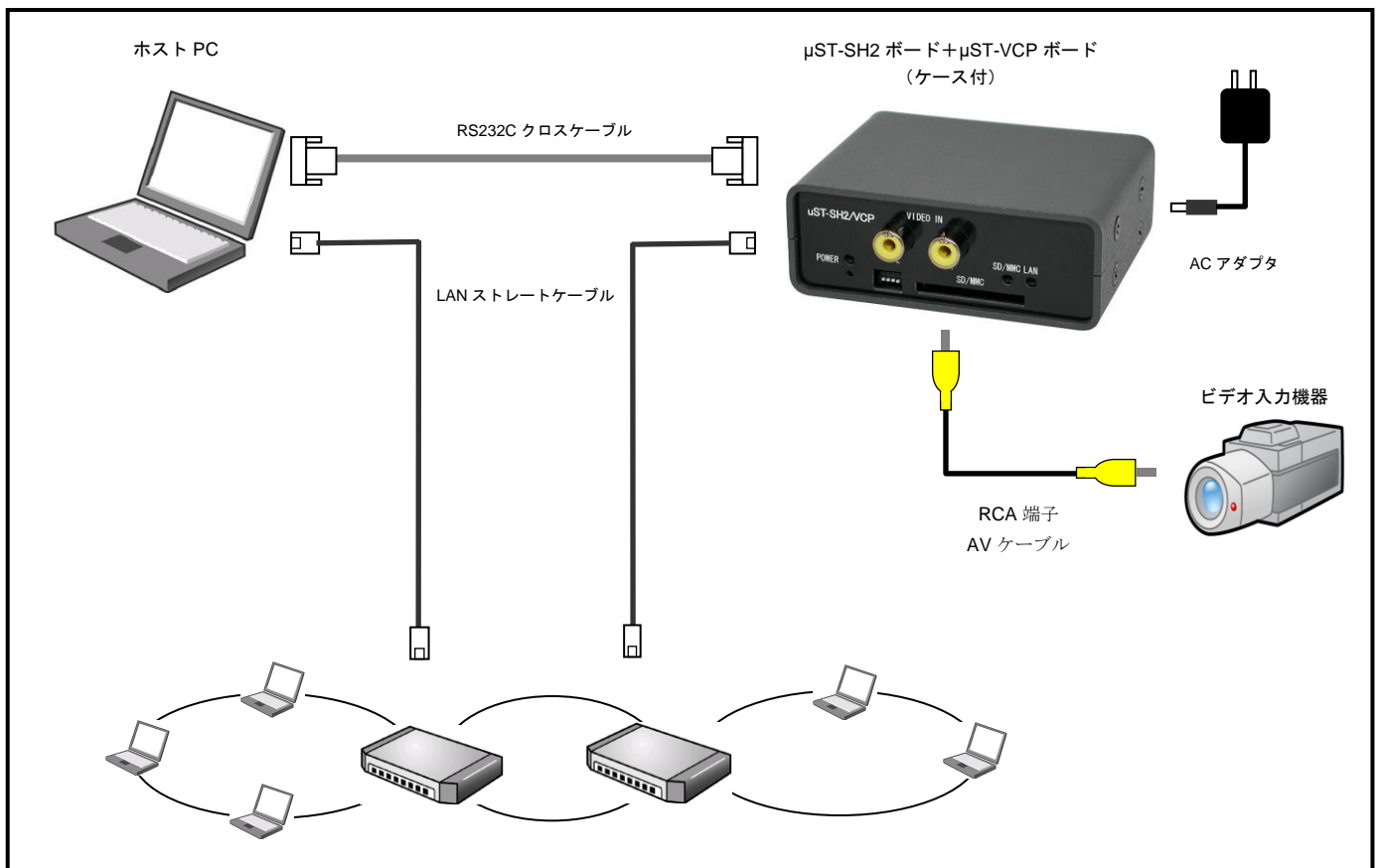


Fig 2.2-4 μST-VCP board connection (HUB connection case: case included)

2.3 デバイスドライバ

μ ST-VCP デバイスドライバ概要

μ ST-VCP のデバイスドライバはビデオキャプチャデバイスドライバで構成されています。μ ST-VCP のデバイスドライバは Video4Linux の後継仕様である Video4Linux2 に対応しています。

Video4Linux とは Linux 上でビデオキャプチャデバイスを制御するための API 仕様であり、対応しているドライバであればアプリケーション側からは統一された手続きでビデオキャプチャデバイスを制御することが可能となります。

μ ST-VCP デバイスドライバのソースファイルは『ustvcp.c』になります。『ustvcp.c』は Linux カーネルの一部としてコンパイルします。

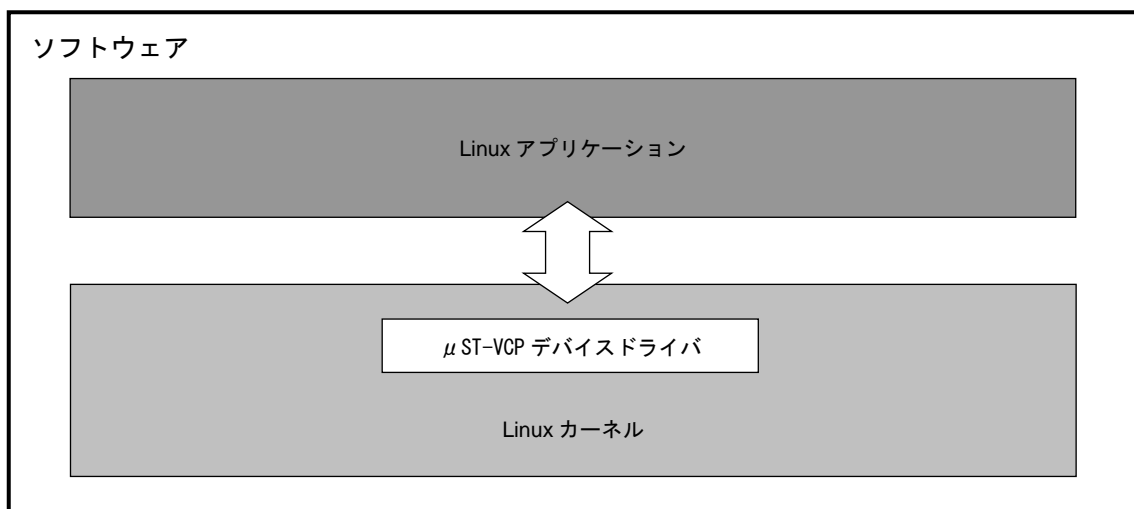


Fig 2.3-1 μ ST-VCP デバイスドライバ概要

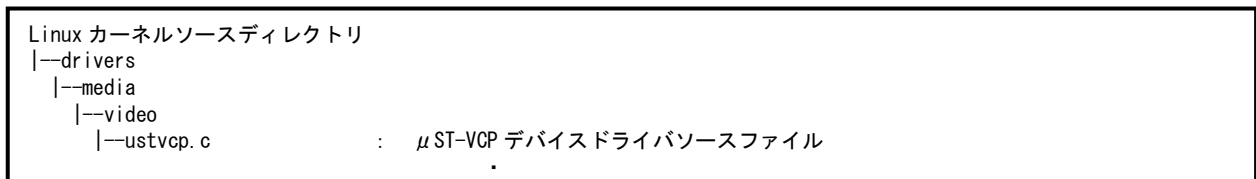


Fig 2.3-2 μ ST-VCP デバイスドライバソースファイル構成

ビデオキャプチャデバイスドライバ概要

ビデオキャプチャデバイスドライバが提供する主なシステムコール (API) は『open』、『close』、『ioctl』、『select』及び『mmap』になります。ビデオキャプチャデバイスを示すデバイスファイルは『/dev/video0』、メジャー番号とマイナー番号は『81』と『0』になります。

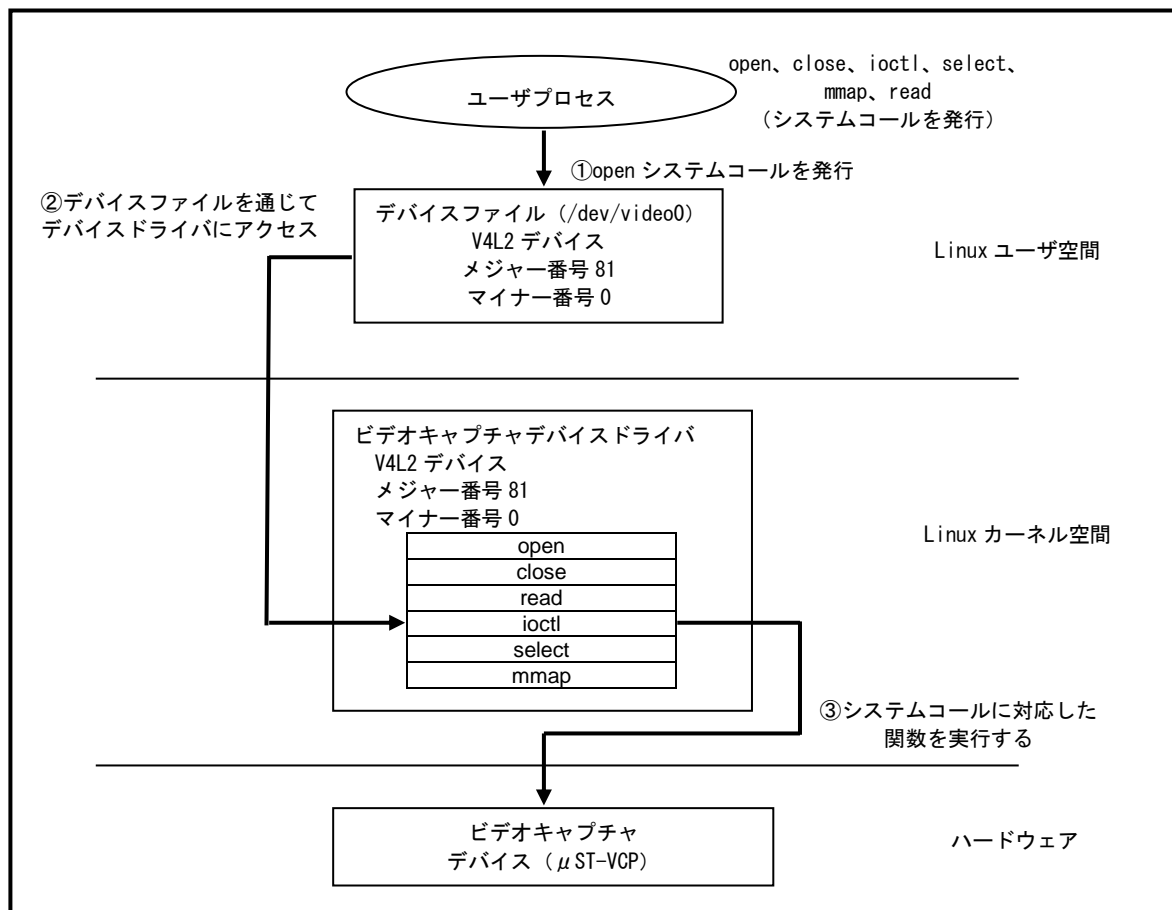


Fig 2.3-3 ビデオキャプチャデバイスドライバの概要

ビデオキャプチャデバイスドライバにアクセスするために各アプリケーションで利用する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	デバイスのオープン
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	成功時にはファイルディスクリプタ番号 エラー時は-1
備考	論理デバイス名は/dev/video0を指定 フラグはO_RDWRを指定

●close システムコール

機能	デバイスのクローズ
書式	int close(int fd)
引数	fd : ファイルディスクリプタ
戻り値	成功時には0、エラー時は-1
備考	

●ioctl システムコール (VIDIOC_S_INPUT)

機能	入力チャンネルの選択
書式	int ioctl(int fd, VIDIOC_S_INPUT, *argp)
引数	fd : ファイルディスクリプタ argp : 入力チャンネル値の整数を指すポインタ
戻り値	成功時には0、エラー時は-1
備考	CH1を選択するときは0を、CH2を選択するときは1を指定 <input type="checkbox"/> 入力チャンネル : CH1 int input = 0; ioctl(fd, VIDIOC_S_INPUT, &input); <input type="checkbox"/> 入力チャンネル : CH2 int input = 1; ioctl(fd, VIDIOC_S_INPUT, &input);

● ioctl システムコール (VIDIOC_S_FMT)

機能	ビデオキャプチャフォーマットの設定
書式	int ioctl(int fd, VIDIOC_S_FMT, struct v4l2_format *argp)
引数	fd : ファイルディスクリプタ argp : struct v4l2_format のアドレス
戻り値	成功時には0、エラー時は-1
備考	<p>v4l2_format 構造体の type と fmt を設定</p> <pre>struct v4l2_format { enum v4l2_buf_type type; union { struct v4l2_pix_format fmt; ... }; };</pre> <p>type メンバは V4L2_BUF_TYPE_VIDEO_CAPTURE を指定 fmt メンバはキャプチャする画像のフォーマットを指定</p> <p><input type="checkbox"/> 画像フォーマット : RGB565、キャプチャサイズ : VGA (640x480) argp->type = V4L2_BUF_TYPE_VIDEO_CAPTURE; argp->fmt.pix.width = 640; argp->fmt.pix.height = 480; argp->fmt.pix.pixelformat = V4L2_PIX_FMT_RGB565; argp->fmt.pix.field = V4L2_FIELD_INTERLACED;</p> <p><input type="checkbox"/> 画像フォーマット : YCbCr16bit、キャプチャサイズ : VGA (640x480) argp->type = V4L2_BUF_TYPE_VIDEO_CAPTURE; argp->fmt.pix.width = 640; argp->fmt.pix.height = 480; argp->fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV; argp->fmt.pix.field = V4L2_FIELD_INTERLACED;</p> <p><input type="checkbox"/> 画像フォーマット : RGB565、キャプチャサイズ : QVGA (320x240) argp->type = V4L2_BUF_TYPE_VIDEO_CAPTURE; argp->fmt.pix.width = 320; argp->fmt.pix.height = 240; argp->fmt.pix.pixelformat = V4L2_PIX_FMT_RGB565; argp->fmt.pix.field = V4L2_FIELD_INTERLACED;</p> <p><input type="checkbox"/> 画像フォーマット : YCbCr16bit、キャプチャサイズ : QVGA (320x240) argp->type = V4L2_BUF_TYPE_VIDEO_CAPTURE; argp->fmt.pix.width = 320; argp->fmt.pix.height = 240; argp->fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV; argp->fmt.pix.field = V4L2_FIELD_INTERLACED;</p>

●ioctl システムコール (VIDIOC_REQBUF)

機能	メモリマッピング I/O の初期化
書式	int ioctl(int fd, VIDIOC_REQBUF, struct v4l2_requestbuffers *argp)
引数	fd : ファイルディスクリプタ argp : struct v4l2_requestbuffers のアドレス
戻り値	成功時には 0、エラー時は -1
備考	

●ioctl システムコール (VIDIOC_QUERYBUF)

機能	バッファのステータスを取得
書式	int ioctl(int fd, VIDIOC_QUERYBUF, struct v4l2_buffer *argp)
引数	fd : ファイルディスクリプタ argp : struct v4l2_buffer のアドレス
戻り値	成功時には 0、エラー時は -1
備考	

●ioctl システムコール (VIDIOC_QBUF)

機能	空のバッファをキューに挿入
書式	int ioctl(int fd, VIDIOC_QBUF, struct v4l2_buffer *argp)
引数	fd : ファイルディスクリプタ argp : struct v4l2_buffer のアドレス
戻り値	成功時には 0、エラー時は -1
備考	

●ioctl システムコール (VIDIOC_DQBUF)

機能	キャプチャした画像データをキューから取得
書式	int ioctl(int fd, VIDIOC_DQBUF, struct v4l2_buffer *argp)
引数	fd : ファイルディスクリプタ argp : struct v4l2_buffer のアドレス
戻り値	成功時には 0、エラー時は -1
備考	

●ioctl システムコール (VIDIOC_STREAMON)

機能	ストリーミング I/O の開始
書式	int ioctl(int fd, VIDIOC_STREAMON, enum v4l2_buf_type *argp)
引数	fd : ファイルディスクリプタ argp : enum v4l2_buf_type の値のアドレス
戻り値	成功時には 0、エラー時は -1
備考	argp には V4L2_BUF_TYPE_VIDEO_CAPTURE を指定 enum v4l2_buf_type type = V4L2_BUF_TYPE_VIDEO_CAPTURE; ioctl(fd, VIDIOC_STREAMON, &type);

●select システムコール

機能	デバイスドライバのキャプチャの待ち合わせ
書式	int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);
引数	nfds : ファイルディスクリプタの最大値に 1 を足したもの readfds : 読み込みの可能性の監視を行うディスクリプタ集合 writefds : 書き込みの可能性の監視を行うディスクリプタ集合 exceptfds : 例外の発生の監視を行うディスクリプタ集合 timeout : select が復帰するまでの経過時間の上限
戻り値	成功時にはディスクリプタの数、エラー時は-1
備考	writefds と exceptfds には NULL を指定

●read システムコール

機能	画像データの読み込み
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	読み出されるデータ形式及びサイズは ioctl(VIDIOC_S_FMT) システムコールで設定したものととなります

3. μ ST-VCP の動作

μ ST-VCP と μ ST-SH2 を使用して、サンプルアプリケーション及びサンプルプログラムを動作させる環境及び設定について説明します。

3.1 μ ST-VCP の動作環境

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

- ホスト PC

μ ST-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。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

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

- シリアル

μ ST-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

μ ST-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。

- ビデオ入力機器

μ ST-VCP の動作を確認するために、NTSC 出力に対応したビデオ入力機器 (ビデオカメラ等) が必要になります。

Table 3.1-1 μ ST-VCP 動作環境

使用機器等	環 境
VCP オプションボード	μ ST-VCP
Linux ボード	μ ST-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
ビデオ入力機器	NTSC 出力可能なビデオ入力機器
AV ケーブル	RCA 端子
電源	DC5V \pm 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。

3.2 μ ST-VCP の設定

μ ST-SH2 用 Linux のために μ ST-VCP ボードの設定を行います。

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

- ① μ ST-VCP のチップセレクトを CS6 に設定します。
 μ ST-VCP の JSW1 スイッチを『CS6』に選択します。

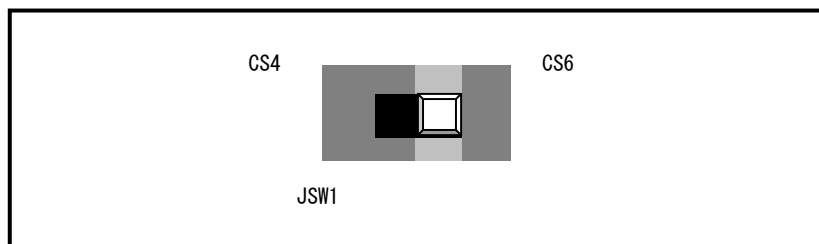


Fig 3.2-1 JSW1 の設定

- ② μ ST-VCP のビデオデコーダの設定をします。
 μ ST-VCP の SW1 スイッチを『OFF/OFF/OFF/OFF』に選択します。

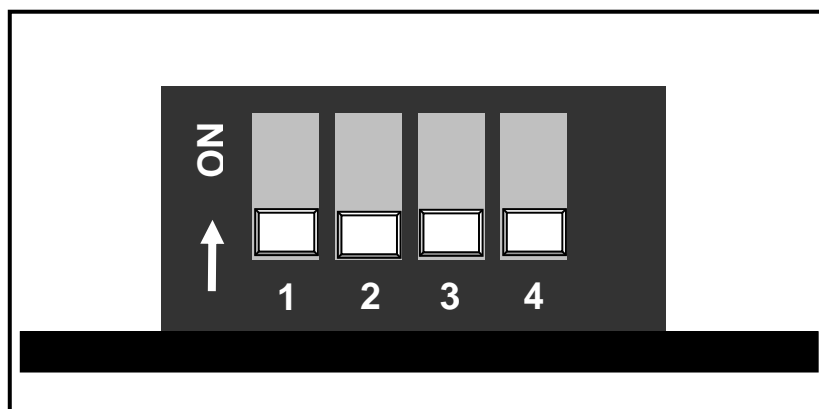


Fig 3.2-2 SW1 の設定

4. サンプルアプリケーションについて

4.1 サンプルアプリケーションの概要

サンプルアプリケーションは μ ST-VCP でキャプチャした画像を、Web ブラウザ上に表示します。 μ ST-SH2 用 Linux 上では Web サーバが動作し、Web ブラウザからのアクセスで Flash アプリケーションをダウンロードします。Web ブラウザ上ではダウンロードした Flash アプリケーションを FlashPlayer で実行します。 μ ST-SH2 用 Linux 上の画像配信サーバ『vcp_serv』から Flash アプリケーションにキャプチャした画像データが送信されます。

Web ブラウザは Firefox や Internet Explorer など FlashPlayer8 以上に対応したものをご利用ください。

※ サンプルアプリケーションのソースコードは公開していません。

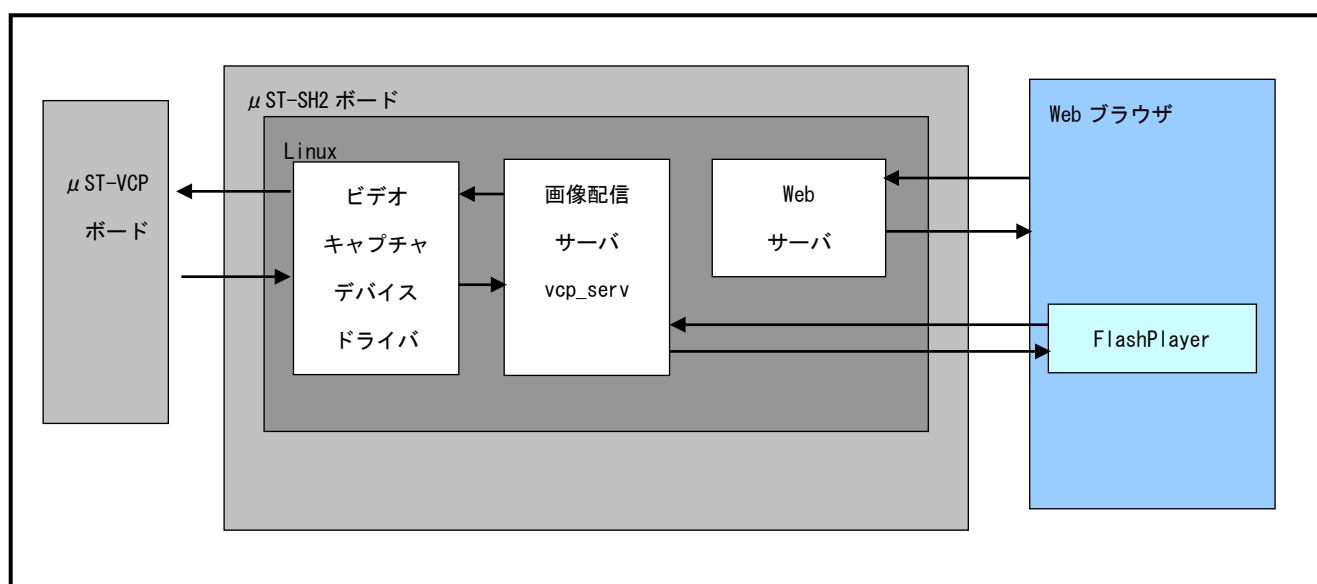


Fig 4.1-1 サンプルアプリケーションの概要

4.2 サンプルアプリケーションの動作

μ ST-SH2 用 Linux での μ ST-VCP のサンプルアプリケーションの動作方法について説明します。

μ ST-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。

PC の IP アドレスは「192.168.128.202」、サブネットマスクは「255.255.255.0」と仮定します。

μ ST-VCP 対応 Linux カーネル

μ ST-VCP を使用するには、 μ ST-VCP 対応 Linux カーネルを起動する必要があります。

μ ST-SH2 用 Linux カーネルはカーネルパラメータに『ustvcp』を渡すことにより μ ST-VCP 対応 Linux カーネルを起動することができます。

μ ST-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=ttyS0,115200 ustvcp』を使用します。

環境変数『bootargs』の設定

```
=> setenv bootargs console=ttyS0,115200 ustvcp
```

Table4.2-1 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 ttyS0 シリアルデバイスをボーレート 115200 bps で使用することを示しています。
ustvcp	-	μ ST-VCP 対応 Linux カーネルを起動します。

- 『2.2 μ ST-VCP の接続』にしたがって、 μ ST-VCP と μ ST-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。
- μ ST-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  ← ← 何かキーを入力する
=>
```

- 環境変数『bootargs』の値を『console=ttyS0,115200 ustvcp』と設定します。

```
=> setenv bootargs console=ttyS0,115200 ustvcp  ← ←
=>
```

- 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 ←  
## 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:
```


サンプルアプリケーションの動作

- ① 『2.2 μ ST-VCP の接続』にしたがって、 μ ST-VCP と μ ST-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。ビデオ入力機器と μ ST-VCP のビデオ信号入力端子 P1 を AV ケーブルで接続し、 μ ST-VCP 対応 Linux カーネルで Linux を起動します。

```
=> setenv bootargs console=ttyS0,115200 ustvcp ◀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・・・
```

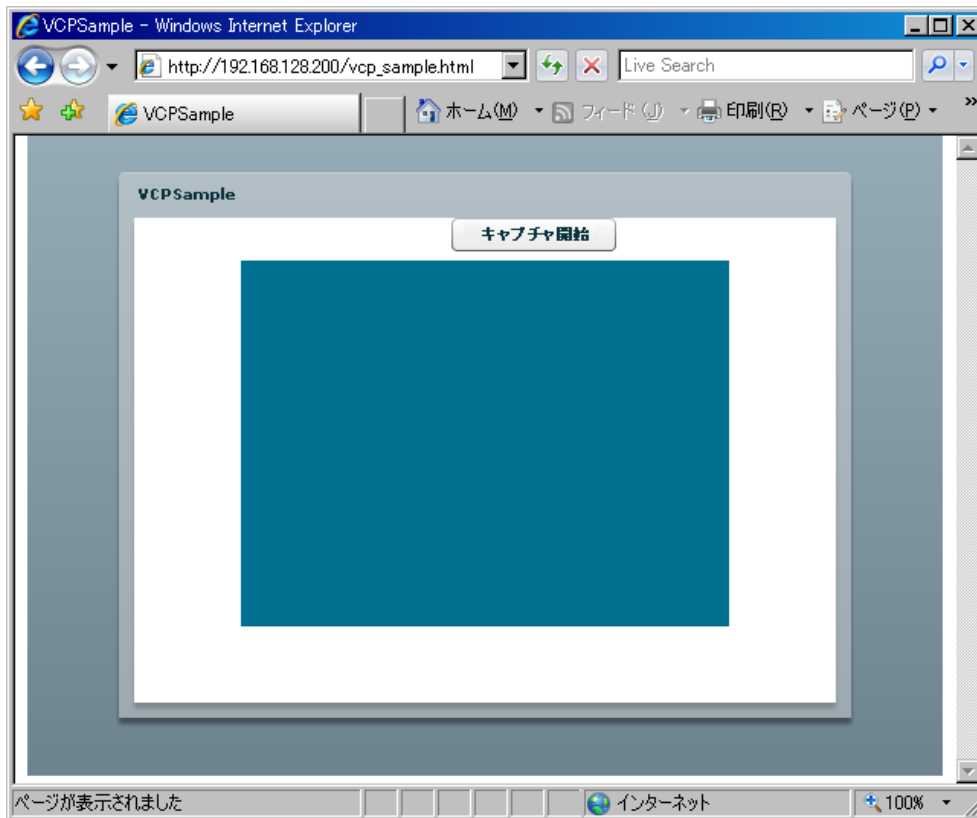
- ② μ ST-SH2 上で Linux の起動を確認し、root 権限でログインします。

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

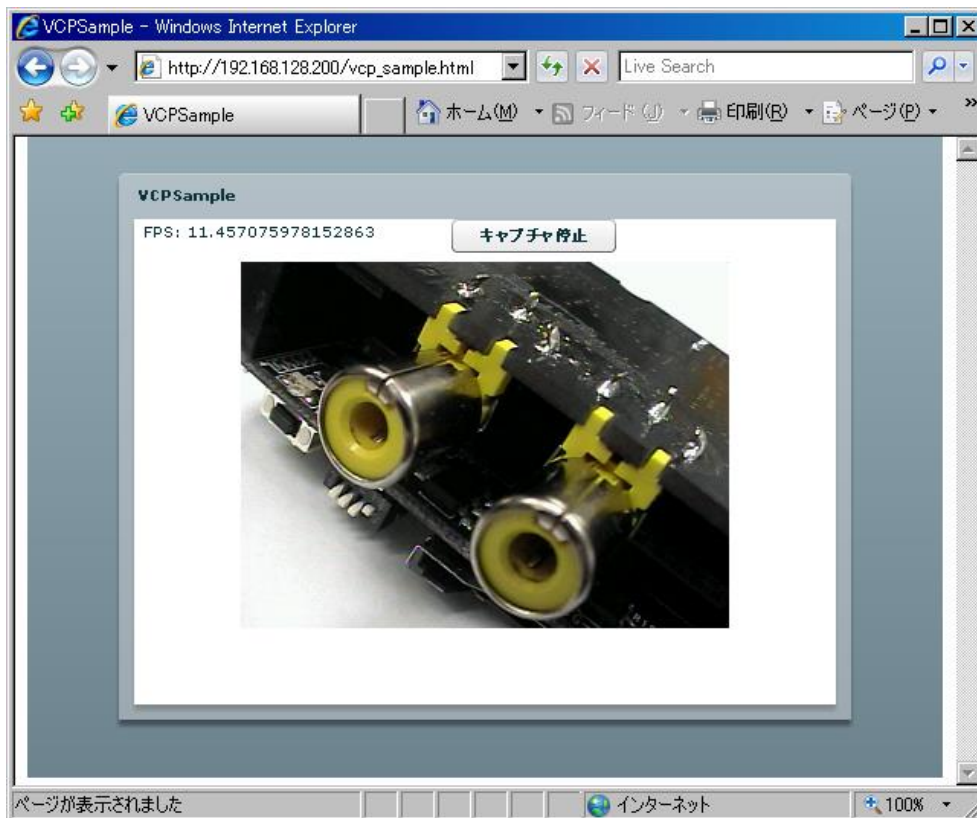
- ③ μ ST-SH2 から画像配信サーバを起動します。

```
~ # /root/vcp_serv & ◀入力
```

- ④ PC上のWebブラウザから μ ST-SH2上のWebサーバにアクセスします。
URLは『http://192.168.128.200/vcp_sample.html』になります。



- ⑤ 画面中の『キャプチャ開始』ボタンを押すと、FlashPlayer上でキャプチャした画像が表示されます。



5. サンプルプログラムについて

μ ST-VCP と μ ST-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。

5.1 サンプルプログラムの概要

μ ST-SH2 用 Linux 上で動作する μ ST-VCP 用サンプルプログラムを作成します。

μ ST-VCP 用サンプルプログラムは、 μ ST-VCP デバイスドライバ『ustvcp.c』を使用してキャプチャした画像をファイルシステムに保存する『ustvcp-sample.c』と、 μ ST-SH2 のシリアル通信を使用してビデオデコーダの設定を変更する『vcp_read.c』『vcp_write.c』『vcp_allread.c』『vcp_save.c』『vcp_clear.c』があります。

μ ST-VCP デバイスドライバは『/home/guest/linuxkit-ustsh2/linux-2.6.33.1-ustsh2/drivers/media/video』ディレクトリ、ビデオキャプチャサンプルプログラムは『/home/guest/linuxkit-ustsh2/samples/ustvcp-sample』ディレクトリにあります。ビデオキャプチャ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。また、キャプチャした画像をビットマップファイルに変換するビットマップ変換プログラム『rgb2bmp.c』のコンパイルも行います。

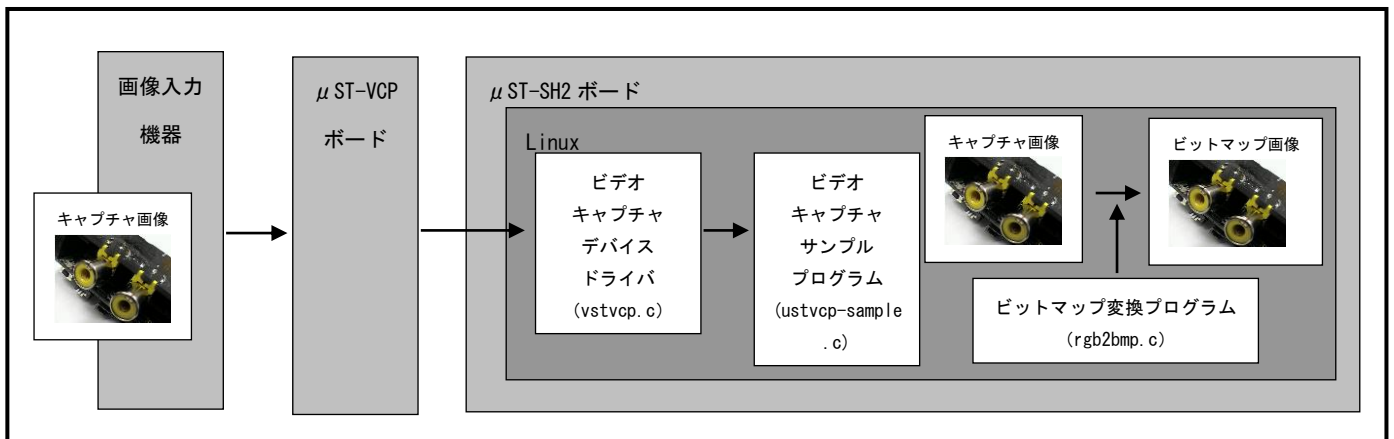


Fig 5.1-1 ビデオキャプチャサンプルプログラムの概要

ビデオデコーダ設定サンプルプログラムは『/home/guest/linuxkit-ustsh2/samples/ustvcp-serial-sample』ディレクトリにあります。こちらのサンプルプログラムのコンパイルは『make』で行います。ビデオデコーダの設定変更はビデオデコーダ IC (ML86V7768) の内部レジスタを μ ST-SH2 のシリアル通信を使用して行います。ビデオデコーダの設定についての詳細は『 μ ST-VCP ハードウェアマニュアル 3.4 ビデオデコーダの設定』をご覧ください。

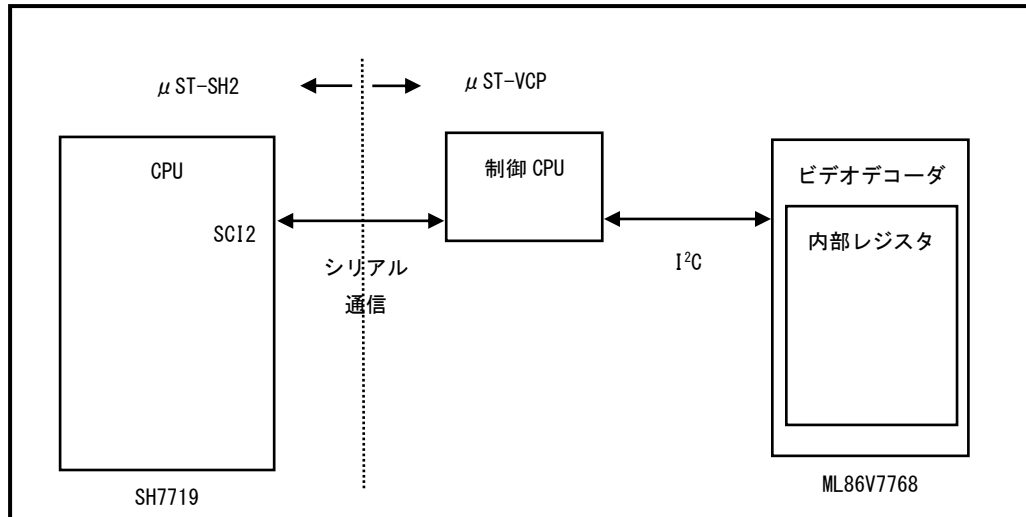


Fig 5.1-2 ビデオデコーダ設定サンプルプログラム

5.2 サンプルプログラムのコンパイル

ビデオキャプチャ

ビデオキャプチャサンプルプログラム『ustvcp-sample.c』をゲスト OS 上でコンパイルします。

『ustvcp-sample.c』は割り込みを使用し連続 4 枚の画像データをカレントディレクトリに保存します。

フォーマットは RGB565、サイズは VGA (640x480) となります。

Video4Linux2 API におけるキャプチャ方法には『read』を使用する方法及び『mmap』を使用する方法の 2 つがあります。

ここで作成するサンプルプログラムは『read』を使用する方法であるため、『mmap』及びそれに関連するシステムコールは使用しません。

※ 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』をご覧ください。

●ustvcp-sample.c ファイル

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/select.h>
#include <linux/videodev.h>

#define SZ_VGA (640*480*2)
#define SZ_QVGA (320*240*2)

#define CLEAR(x) memset(&x, 0, sizeof(x))

#define DEBUG 0

#ifdef DEBUG
#define DBG(...) do {
    fprintf(stderr, "%s:%i ", __func__, __LINE__);
    fprintf(stderr, __VA_ARGS__);
    fflush(stderr);
} while(0)
#else
#define DBG(...)
#endif

#define DBG_EXIT(...) do {
    DBG(__VA_ARGS__);
    exit(1);
} while(0)

static ssize_t imagesize = SZ_VGA;

static void
errno_exit(const char *s)
{
    fprintf(stderr, "%s error %d, %s\n", s, errno, strerror(errno));

    exit(EXIT_FAILURE);
}

static int
xioctl(int fd, int request, void *arg)
{
```

```
int r;

do
    r = ioctl(fd, request, arg);
while (-1 == r && EINTR == errno);

return r;
}

ssize_t
xread(int fd, void *buf, size_t count)
{
    size_t nleft = count;
    char *ptr = buf;
    ssize_t nread;

    while (nleft > 0)
    {
        if ((nread = read(fd, ptr, nleft)) < 0)
        {
            if (errno == EINTR)
                continue;
            return -1;
        }
        else if (nread == 0)
            break;

        nleft -= nread;
        ptr += nread;
    }

    return count - nleft;
}

static ssize_t
xwrite(int fd, const void *buf, size_t count)
{
    const char *ptr = buf;
    size_t nleft = count;
    ssize_t nwritten;

    while (nleft > 0)
    {
        if ((nwritten = write(fd, ptr, nleft)) <= 0)
        {
            if (errno == EINTR)
                nwritten = 0;
            else
                return -1;
        }

        nleft -= nwritten;
        ptr += nwritten;
    }

    return count;
}

static void
save_file(char *buf, int n)
{
    char filename[16];
    int fd;

    sprintf(filename, "vcp.data.%d", n);

    fd = open(filename, O_WRONLY | O_CREAT, 00666);

    xwrite(fd, buf, imagesize);
}
```

```
printf("%d byte written to ./%s¥n", imagesize, filename);
close(fd);
}

int main(int argc, char **argv)
{
    int fd;
    int ret;
    fd_set allfds;
    /* 保存する画像データの枚数 */
    int nsave = 4;
    int i;
    struct v4l2_format fmt;
    char *device = "/dev/video0";
    char *buffer = NULL;
    int length = 640 * 480 * 2;
    int input;

    if (argc >= 2)
        device = argv[1];

    /* デバイスファイルのオープン */
    fd = open(device, O_RDWR);
    if (fd == -1)
    {
        fprintf(stderr, "open %s: %s¥n", device, strerror(errno));
        exit(1);
    }

    /* ビデオキャプチャモードの設定
       フォーマット: RGB565, キャプチャサイズ: VGA(640x480) */
    fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    fmt.fmt.pix.width = 640;
    fmt.fmt.pix.height = 480;
    fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_RGB565;
    fmt.fmt.pix.field = V4L2_FIELD_INTERLACED;

    if (-1 == xioctl(fd, VIDIOC_S_FMT, &fmt))
        errno_exit("VIDIOC_S_FMT");

    /* 入力チャンネルの設定 CH1 0 => CH1, 1 => CH2 */
    input = 0;
    if (-1 == xioctl(fd, VIDIOC_S_INPUT, &input))
        errno_exit("VIDIOC_S_INPUT");

    buffer = malloc(length);

    if (buffer == NULL)
        exit(EXIT_FAILURE);

    FD_ZERO(&allfds);
    FD_SET(fd, &allfds);

    for (i = 0; i < nsave; i++)
    {
        struct timeval tv = { .tv_sec = 2, .tv_usec = 0 };
        fd_set rfds = allfds;



        ret = select(fd + 1, &rfds, NULL, NULL, &tv);

        DBG("ret=%d¥n", ret);

        if (ret < 0)
        {
            fprintf(stderr, "select: %s¥n", strerror(errno));
            exit(1);
        }
    }
}
```

```
printf("Take %d !!¥n", i);  
  
xread(fd, buffer, length);  
  
if (ret == 0)  
    continue;  
  
save_file(buffer, i);  
  
    //    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   
Password:*****  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/ustvcp-sample`』を実行してください。

```
[guest@LinuxKit ~]$ cd linuxkit-ustsh2/samples/ustvcp-sample 
```


- ③ ゲスト OS 上でビデオキャプチャサンプルプログラム『`ustvcp-sample.c`』をコンパイルします。

『`sh2eb-linux-gcc -elf2flt="-ar" -o ustvcp-sample ustvcp-sample.c`』を実行し、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 ustvcp-sample]$ sh2eb-linux-gcc -elf2flt="-ar" -o ustvcp-sample ustvcp-sample.c 
```

```
[guest@LinuxKit ustvcp-sample]$ chmod a+x ustvcp-sample 
```


- ④ ゲスト OS 上でビデオキャプチャサンプルプログラムを NFS 共有ディレクトリ『`/nfs`』にコピーします。

```
[guest@LinuxKit ustvcp-sample]$ cp -a ustvcp-sample /nfs 
```


- ⑤ ゲスト OS 上でビットマップ変換プログラム『`rgb2bmp.c`』をコンパイルします。

『`sh2eb-linux-gcc -elf2flt="-ar" -o rgb2bmp rgb2bmp.c`』を実行し、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 ustvcp-sample]$ sh2eb-linux-gcc -elf2flt="-ar" -o rgb2bmp rgb2bmp.c 
```

```
[guest@LinuxKit ustvcp-sample]$ chmod a+x rgb2bmp 
```

- ⑥ ゲスト OS 上でビットマップ変換プログラムを NFS 共有ディレクトリ『`/nfs`』にコピーします。

```
[guest@LinuxKit ustvcp-sample]$ cp -a rgb2bmp /nfs 
```

ビデオデコーダ設定変更

シリアル通信を使用したビデオデコーダの設定変更サンプルプログラム『vcp_read』『vcp_write』『vcp_allread』『vcp_save』『vcp_clear』をゲスト OS 上でコンパイルします。

『vcp_read 引数 1』は指定のビデオデコーダのレジスタ設定値を読み出します。引数 1 は読み出し先レジスタのアドレスを示します。16 進数で指定してください。

『vcp_write 引数 1 引数 2』は指定のビデオデコーダのレジスタに設定値を書き込みます。引数 1 は書き込み先レジスタのアドレスを示し、引数 2 は書き込む設定値を示します。両方とも 16 進数で指定してください。

『vcp_allread』は全てのビデオデコーダのレジスタ設定値を読み出します。

『vcp_save』はその時点でのビデオデコーダのレジスタ設定値を保存します。次回起動時に保存値をビデオデコーダに適応させます。

『vcp_clear』は『vcp_save』で保存したレジスタ設定値を消去し、初期値に戻します。

●vcp_read.c ファイル

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include "scif2.h"

/* プロトタイプ宣言 */
char register_get(unsigned char, unsigned char *);

/* メイン関数 */
int main(int argc, char **argv)
{
    unsigned char data, g_add;

    char check, *dummy;

    if(argc != 2) {
        printf("%s INCORRECT ARGUMENT\n", argv[0]);
        return 1;
    }

    g_add = (unsigned char) strtoul(argv[1], &dummy, 0);

    printf("READ PROGRAM START\n");

    check = SCIF2_COMOpen();

    if(check != 0x00) {
        printf("COM OPEN ERROR\n");
        return 1;
    }

    check = register_get(g_add, &data);

    if(check == 0)
        printf("READ OK 0x%02hhx = 0x%02hhx\n", g_add, data);
    else if(check == 1)
        printf("RETURN ERROR MESSAGE = 0x%02hhx\n", data);
    else if(check == -1)
        printf("RETURN MESSAGE CHECKSUM ERROR\n");
    else
        printf("READ ERROR\n");

    printf("READ PROGRAM END\n");

    SCIF2_COMClose();

    return 0;
}

/* レジスタ設定値読み出し */
```

```
char register_get(unsigned char add, unsigned char* data)
{
    unsigned char count, chk, w_data[4], r_data[4];
    int i;

    w_data[0] = 0x01;
    w_data[1] = add;
    w_data[2] = (0x01 ^ add);

    count = 0;

    for(i=0; i <= 1000; i++) {
        chk = SCIF2_Write(&w_data[count], 1);
        if(chk == 0x00)
            count++;
            if(count >= 3)
                break;
        usleep(1000); /* 1ms ウェイト */
    }

    count = 0;

    for(i=0; i <= 1000; i++) {
        chk = SCIF2_Read(&r_data[count], 1);
        if(chk == 0x00)
            count++;
            if(count >= 3)
                break;
        usleep(1000); /* 1ms ウェイト */
    }

    if(r_data[0] == 0x01) {
        if( (r_data[0] ^ r_data[1]) == r_data[2]) {
            *data = r_data[1];
            return 0;
        }
        else
            return -1;
    }
    else if(r_data[0] == 0xFF) {
        if( (r_data[0] ^ r_data[1]) == r_data[2]) {
            *data = r_data[1];
            return 1;
        }
        else
            return -1;
    }
    else
        return -2;
}
```

●vcp_write.c ファイル

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include "scif2.h"

/* プロトタイプ宣言 */
char register_set(unsigned char, unsigned char, unsigned char *);

/* メイン関数 */
int main(int argc, char **argv)
{
    unsigned char data, s_add, value;
    char check, *dummy;

    if(argc != 3) {
        printf("%s INCORRECT ARGUMENT\n", argv[0]);
        return 1;
    }

    s_add = (unsigned char) strtol(argv[1], &dummy, 0);
    value = (unsigned char) strtol(argv[2], &dummy, 0);

    printf("WRITE PROGRAM START\n");

    check = SCIF2_COMOpen();

    if(check != 0x00) {
        printf("COM OPEN ERROR\n");
        return 1;
    }

    check = register_set(s_add, value, &data);

    if(check == 0)
        printf("WRITE OK 0x%02hhx\n", value);
    else if(check == 1)
        printf("RETURN ERROR MESSAGE = 0x%02hhx\n", data);
    else if(check == -1)
        printf("RETURN MESSAGE CHECKSUM ERROR\n");
    else
        printf("WRITE ERROR\n");

    printf("WRITE PROGRAM END\n");

    SCIF2_COMClose();

    return 0;
}

/* レジスタ設定値書き込み */
char register_set(unsigned char add, unsigned char set, unsigned char* data)
{
    unsigned char count, sum, chk, w_data[5], r_data[4];
    int i;

    sum = 0;

    w_data[0] = 0x00;
    sum ^= w_data[0];
    w_data[1] = add;
    sum ^= w_data[1];
    w_data[2] = set;
    sum ^= w_data[2];
    w_data[3] = sum;

    count = 0;
```

```
for(i=0; i <= 1000; i++) {
    chk = SCIF2_Write(&w_data[count], 1);
    if(chk == 0x00)
        count++;
        if(count >= 4)
            break;
    usleep(1000); /* 1ms ウェイト */
}

count = 0;

for(i=0; i <= 1000; i++) {
    chk = SCIF2_Read(&r_data[count], 1);
    if(chk == 0x00)
        count++;
        if(count >= 3)
            break;
    usleep(1000); /* 1ms ウェイト */
}

if(r_data[0] == 0x00) {
    if( (r_data[0] ^ r_data[1]) == r_data[2])
        return 0;
    else
        return -1;
}
else if(r_data[0] == 0xFF) {
    if( (r_data[0] ^ r_data[1]) == r_data[2]) {
        *data = r_data[1];
        return 1;
    }
    else
        return -1;
}
else
    return -2;
}
```

●vcp_allread.c ファイル

```
#include <stdio.h>
#include <unistd.h>
#include "scif2.h"

/* プロトタイプ宣言 */
char register_get(unsigned char, unsigned char *);

/* メイン関数 */
int main(void)
{
    unsigned char data, g_add;
    unsigned char all_add[58] = {0x00, 0x01, 0x02, 0x04, 0x05, 0x08, 0x10, 0x11, 0x12, 0x20, 0x21, 0x22, 0x23,
                                0x24, 0x25, 0x26, 0x27, 0x28, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
                                0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x50, 0x51, 0x54, 0x58, 0x68,
                                0x69, 0x6A, 0x70, 0x71, 0x72, 0x73, 0x78, 0x79, 0x7A, 0x80, 0x81, 0x82, 0x83,
                                0x84, 0x85, 0x86, 0x87, 0x88, 0x89};

    char check;
    int i;

    printf("ALLREAD PROGRAM START\n");

    check = SCIF2_COMOpen();

    if(check != 0x00) {
        printf("COM OPEN ERROR\n");
        return 1;
    }

    for(i=0; i < 58; i++) {
        g_add = all_add[i];

        check = register_get(g_add, &data);

        if(check == 0)
            printf("READ OK 0x%02hhx = 0x%02hhx\n", g_add, data);
        else if(check == 1)
            printf("RETURN ERROR MESSAGE = 0x%02hhx\n", data);
        else if(check == -1)
            printf("RETURN MESSAGE CHECKSUM ERROR\n");
        else
            printf("READ ERROR\n");
    }

    printf("ALLREAD PROGRAM END\n");

    SCIF2_COMClose();

    return 0;
}

/* レジスタ設定値読み出し */
char register_get(unsigned char add, unsigned char* data)
{
    unsigned char count, chk, w_data[4], r_data[4];
    int i;

    w_data[0] = 0x01;
    w_data[1] = add;
    w_data[2] = (0x01 ^ add);

    count = 0;

    for(i=0; i <= 1000; i++) {
        chk = SCIF2_Write(&w_data[count], 1);
        if(chk == 0x00)
            count++;
    }
}
```

```
                if(count >= 3)
                    break;
                usleep(1000); /* 1ms ウェイト */
            }
count = 0;
for(i=0; i <= 1000; i++) {
    chk = SCIF2_Read(&r_data[count], 1);
    if(chk == 0x00)
        count++;
        if(count >= 3)
            break;
        usleep(1000); /* 1ms ウェイト */
}
if(r_data[0] == 0x01) {
    if( (r_data[0] ^ r_data[1]) == r_data[2]) {
        *data = r_data[1];
        return 0;
    }
    else
        return -1;
}
else if(r_data[0] == 0xFF) {
    if( (r_data[0] ^ r_data[1]) == r_data[2]) {
        *data = r_data[1];
        return 1;
    }
    else
        return -1;
}
else
    return -2;
}
```

●vcp_save.c ファイル

```
#include <stdio.h>
#include <unistd.h>
#include "scif2.h"

/* プロトタイプ宣言 */
char memory_save(unsigned char *);

#define SAVE_SIG 0xFE

/* メイン関数 */
int main(void)
{
    unsigned char data;
    char check;

    printf("SAVE PROGRAM START\n");

    check = SCIF2_COMOpen();

    if(check != 0x00) {
        printf("COM OPEN ERROR\n");
        return 1;
    }

    check = memory_save(&data);

    if(check == 0)
        printf("SAVE OK\n");
    else if(check == 1)
        printf("RETURN ERROR MESSAGE = 0x%02hhx\n", data);
    else if(check == -1)
        printf("RETURN MESSAGE CHECKSUM ERROR\n");
    else
        printf("SAVE ERROR\n");

    printf("SAVE PROGRAM END\n");

    SCIF2_COMClose();

    return 0;
}

/* 設定値保存 */
char memory_save(unsigned char* data)
{
    unsigned char count, sum = 0, chk, w_data[7], r_data[4];
    int i;

    for(i=0; i <= 4; i++)
    {
        w_data[i] = SAVE_SIG;
        sum ^= w_data[i];
    }

    w_data[5] = sum;

    count = 0;

    for(i=0; i <= 1000; i++) {
        chk = SCIF2_Write(&w_data[count], 1);
        if(chk == 0x00)
            count++;
        if(count >= 6)

```



```
                break;
        usleep(1000); /* 1ms ウェイト */
    }
    count = 0;
    for(i=0; i <= 1000; i++) {
        chk = SCIF2_Read(&r_data[count], 1);
        if(chk == 0x00)
            count++;
            if(count >= 3)
                break;
        usleep(1000); /* 1ms ウェイト */
    }
    if(r_data[0] == 0x00) {
        if( (r_data[0] ^ r_data[1]) == r_data[2])
            return 0;
        else
            return -1;
    }
    else if(r_data[0] == 0xFF) {
        if( (r_data[0] ^ r_data[1]) == r_data[2]) {
            *data = r_data[1];
            return 1;
        }
        else
            return -1;
    }
    else
        return -2;
}
```

●vcp_clear.c ファイル

```

#include <stdio.h>
#include <unistd.h>
#include "scif2.h"

/* プロトタイプ宣言 */
char memory_clr(unsigned char *);

#define CLR_SIG 0xFF

/* メイン関数 */
int main(void)
{
    unsigned char data;
    char check;

    printf("CLEAR PROGRAM START\n");

    check = SCIF2_COMOpen();

    if(check != 0x00) {
        printf("COM OPEN ERROR\n");
        return 1;
    }

    check = memory_clr(&data);

    if(check == 0)
        printf("CLEAR OK\n");
    else if(check == 1)
        printf("RETURN ERROR MESSAGE = 0x%02hx\n", data);
    else if(check == -1)
        printf("RETURN MESSAGE CHECKSUM ERROR\n");
    else
        printf("CLEAR ERROR\n");

    printf("CLEAR PROGRAM END\n");

    SCIF2_COMClose();

    return 0;
}

/* 設定値初期化 */
char memory_clr(unsigned char* data)
{
    unsigned char count, sum = 0, chk, w_data[7], r_data[4];
    int i;

    for(i=0; i <= 4; i++) {
        w_data[i] = CLR_SIG;
        sum ^= w_data[i];
    }

    w_data[5] = sum;

    count = 0;

    for(i=0; i <= 1000; i++) {
        chk = SCIF2_Write(&w_data[count], 1);
        if(chk == 0x00)
            count++;
            if(count >= 6)
                break;

        usleep(1000);
    }
}
/* 1ms ウェイト */

```

```
    }  
  
    count = 0;  
  
    for(i=0; i <= 1000; i++) {  
        chk = SCIF2_Read(&r_data[count], 1);  
        if(chk == 0x00)  
            count++;  
            if(count >= 3)  
                break;  
        usleep(1000); /* 1ms ウェイト */  
    }  
  
    if(r_data[0] == 0x00) {  
        if( (r_data[0] ^ r_data[1]) == r_data[2])  
            return 0;  
        else  
            return -1;  
    }  
    else if(r_data[0] == 0xFF)  
    {  
        if( (r_data[0] ^ r_data[1]) == r_data[2]) {  
            *data = r_data[1];  
            return 1;  
        }  
        else  
            return -1;  
    }  
    else  
        return -2;  
}
```

- 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/ustvcp-serial-sample`』を実行してください。

```
[guest@LinuxKit ~]$ cd linuxkit-ustsh2/samples/ustvcp-serial-sample ←入力
```

- ③ ゲスト OS 上で各アプリケーションプログラムをコンパイルします。

『`make`』を実行すると、自動的にコンパイルが行われ、各アプリケーションに実行権が与えられます。

```
[guest@LinuxKit ustvcp-serial-sample]$ make ←入力
```

- ④ ゲスト OS 上で各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 ustvcp-serial-sample]$ cp vcp_read vcp_write vcp_allread vcp_save vcp_clear /nfs ←入力
```

5.3 サンプルプログラムの動作

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

μ ST-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。

ビデオキャプチャ

ビデオキャプチャサンプルプログラム『ustvcp-sample』は、割り込みを使用し連続 4 枚の画像データをカレントディレクトリに保存します。保存された画像は RGB565 フォーマットのバイナリデータであるため、これをビットマップファイルに変換して閲覧します。RGB565 フォーマットのバイナリデータで保存された画像を bmp フォーマットに変換するために、『rgb2bmp』コマンドを使用します。

以下では μ ST-SH2 のフラッシュROMに μ ST-VCP 対応 Linux カーネルが書かれていることを前提としています。

- ① 『2.2 μ ST-VCP の接続』にしたがって、 μ ST-VCP と μ ST-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。ビデオ入力機器と μ ST-VCP のビデオ信号入力端子 P1 を AV ケーブルで接続し、 μ ST-VCP 対応 Linux カーネルで Linux を起動します。

```

=> setenv bootargs console=ttyS00,115200 ustvcp
=> bootm a0100000
## 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 μ ST-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
~ #

```

- ③ ゲスト OS を起動した状態にしておき、 μ ST-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 192.168.128.201:/nfs /mnt/nfs』を実行してください。

```

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

```

- ④ ゲスト OS 上で作成したビデオキャプチャ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/rgb2bmp /mnt/nfs/ustvcp-sample .』を実行してください。

```

~ # cp -a /mnt/nfs/rgb2bmp /mnt/nfs/ustvcp-sample .

```

- ⑤ μ ST-SH2 上でサンプルプログラムを実行します。『./ustvcp-sample』を実行してください。

```
~ # ./ustvcp-sample
614400 byte written to ./vcp.data.0
614400 byte written to ./vcp.data.1
614400 byte written to ./vcp.data.2
614400 byte written to ./vcp.data.3
```

- ⑥ ファイルが作成されているのを確認します。

```
~ # ls -l
total 2516
-rwxr--r-- 1 root root 30024 Jun 19 08:25 rgb2bmp
-rwxr--r-- 1 root root 26136 Jun 19 08:26 ustvcp-sample
-rw-r--r-- 1 root root 614400 Jun 19 08:26 vcp.data.0
-rw-r--r-- 1 root root 614400 Jun 19 08:26 vcp.data.1
-rw-r--r-- 1 root root 614400 Jun 19 08:26 vcp.data.2
-rw-r--r-- 1 root root 614400 Jun 19 08:26 vcp.data.3
-rwxr--r-- 1 root root 56024 Jun 11 2010 vcp_serv
```

- ⑦ 1つのファイルをBMPフォーマットに変換します。

『./rgb2bmp vcp.data.0 out.bmp』を実行してください。

```
~ # ./rgb2bmp vcp.data.0 out.bmp
write to out.bmp
```

- ⑧ 生成されたBMPファイルをゲストOS上のNFS共有ディレクトリに移動します。

『mv out.bmp /mnt/nfs』を実行してください。

```
~ # mv out.bmp /mnt/nfs
```

- ⑨ 『 μ ST-SH2 Linux 開発キット ソフトウェアマニュアル VMware Player 編』を参照し、FTP又はSambaを使用したファイル転送の準備をします。

以降はゲストOS上での作業となります。

- ⑩ ゲストOS上のNFS共有ディレクトリにあるBMPファイルを、guestユーザのホームディレクトリに移動します。

『mv /nfs/out.bmp /home/guest』を実行してください。

```
[guest@LinuxKit ~]$ mv /nfs/out.bmp /home/guest
```

- ⑪ guestユーザのホームディレクトリに移動したBMPファイルを、FTPやSamba等を使用してWindows上にダウンロードし、画像表示ソフト（ペイントやWindows Picture and fax Viewer等）で確認します。

シリアル通信 ビデオデコーダ設定変更 前半

シリアル通信を使ったビデオデコーダの設定変更サンプルプログラムを動作します。

前半では『vcp_write』コマンドでビデオデコーダ IC (ML86V7668) の内部レジスタの値を変更し、『vcp_read』で内部レジスタを読み出し、値が変更されているか確認します。そして『vcp_save』コマンドでビデオデコーダ設定を保存します。

ビデオデコーダ IC (ML86V7668) の内部レジスタの構成については『ML86V7668 データシート ML86V7668.pdf』をご覧ください。

以下では μ ST-SH2 のフラッシュROMに μ ST-VCP 対応 Linux カーネルが書かれていることを前提としています。

- ① 『2.2 μ ST-VCP の接続』にしたがって、 μ ST-VCP と μ ST-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。ビデオ入力機器と μ ST-VCP のビデオ信号入力端子 P1 を AV ケーブルで接続し、 μ ST-VCP 対応 Linux カーネルで Linux を起動します。

```

=> setenv bootargs console=ttyS00,115200 ustvcp
=> bootm a0100000
## 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 μ ST-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
~ #

```

- ③ ゲスト OS を起動した状態にしておき、 μ ST-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 192.168.128.201:/nfs /mnt/nfs』を実行してください。

```

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

```

- ④ ゲスト OS 上で作成したビデオデコーダの設定変更サンプルプログラム 3 つを NFS 共有ディレクトリからコピーします。

『cp /mnt/nfs/vcp_write /mnt/nfs/vcp_read /mnt/nfs/vcp_save .』を実行してください。

```

~ # cp /mnt/nfs/vcp_write /mnt/nfs/vcp_read /mnt/nfs/vcp_save .

```

- ⑤ μ ST-SH2 上でアプリケーションプログラム『vcp_write』を実行して、ビデオデコーダのレジスタ『0x01』に設定値『0xa2』を書き込みます。

『./vcp_write 0x01 0xa2』を実行してください。

```

~ # ./vcp_write 0x01 0xa2

```

- ⑥ コンソール上に『WRITE OK 0xa2』と表示されることを確認してください。

```

~ # ./vcp_write 0x01 0xa2
WRITE PROGRAM START
WRITE OK 0xa2
WRITE PROGRAM END

```

- ⑦ μ ST-SH2 上でアプリケーションプログラム『vcp_read』を実行して、ビデオデコーダのレジスタ『0x01』の設定値を読み出します。

『./vcp_read 0x01』を実行してください。

```
~ # ./vcp_read 0x01
```

- ⑧ コンソール上に『READ OK 0x01 = 0xa2』と表示されることを確認してください。

```
~ # ./vcp_read 0x01
READ PROGRAM START
READ OK 0x01 = 0xa2
READ PROGRAM END
```

- ⑨ μ ST-SH2 上でアプリケーションプログラム『vcp_save』を実行して、現在の設定値を保存します

『./vcp_save』を実行してください。

```
~ # ./vcp_save
```

- ⑩ コンソール上に『SAVE OK』と表示されることを確認してください。

```
~ # ./vcp_save
SAVE PROGRAM START
SAVE OK
SAVE PROGRAM END
```

- ⑪ μ ST-SH2 の電源を切ります。

- ⑫ 引き続き、「シリアル通信 ビデオデコーダ設定変更 後半」を行ってください。

シリアル通信 ビデオデコーダ設定変更 後半

ビデオデコーダの設定変更サンプルプログラムの後半では一度、電源を落とした後にユーザモードを ON にして起動し、『vcp_save』コマンドで保存した設定データを確認します。その際にビデオデコーダの全レジスタを読み出す『vcp_allread』コマンドを使用します。そして、『vcp_clear』コマンドを使用して保存したデータをクリアし、ビデオデコーダのレジスタの値を出荷時設定に戻します。最後にユーザモードを OFF にして起動し、出荷時設定に復元されていることを確認します。

- ① μ ST-VCP の SW1 スイッチを『OFF/ON/OFF/OFF』に設定します。

ケース付き μ ST-SH2 + μ ST-VCP ボードを御使用のお客様はケースから μ ST-VCP を取り出して、SW1 スイッチの設定を行ってください。ケースからの取り出し方法は、『 μ ST-VCP ハードウェアマニュアル 5.2 ケースへの組み込み方法』を参照してください。

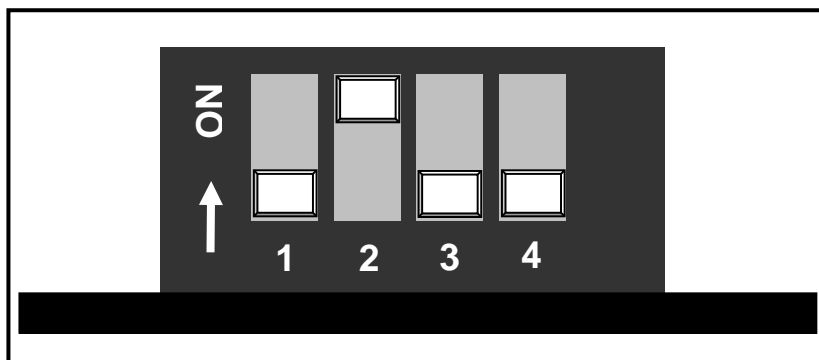


Fig 5.3-1 SW1 の設定

- ② 『2.2 μ ST-VCP の接続』にしたがって、 μ ST-VCP と μ ST-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。ビデオ入力機器と μ ST-VCP のビデオ信号入力端子 P1 を AV ケーブルで接続し、 μ ST-VCP 対応 Linux カーネルで Linux を起動します。

```
=> setenv bootargs console=ttyS00,115200 ustvcp
=> 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・・・
```

- ③ μ ST-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
~ #
```

- ④ ゲスト OS を起動した状態にしておき、 μ ST-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 192.168.128.201:/nfs /mnt/nfs』を実行してください。

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

- ⑤ ゲスト OS 上で作成したビデオデコーダの設定変更サンプルプログラム 2 つを NFS 共有ディレクトリからコピーします。

『cp /mnt/nfs/vcp_allread /mnt/nfs/vcp_clear .』を実行してください。

```
~ # cp /mnt/nfs/vcp_allread /mnt/nfs/vcp_clear . ←入力
```

- ⑥ μ ST-SH2 上でアプリケーションプログラム『vcp_allread』を実行して、ビデオデコーダの全レジスタの設定値を読み出します。

『./vcp_allread』を実行してください。

```
~ # ./vcp_allread ←入力
```

- ⑦ コンソール上に『 READ OK 0x01 = 0xa2 ... (以下省略) 』と表示されることを確認してください。

```
~ # ./vcp_allread
ALLREAD PROGRAM START
READ OK 0x00 = 0x13
READ OK 0x01 = 0xa2 ←入力
.
.
(途中省略)
ALLREAD PROGRAM END
```

- ⑧ μ ST-SH2 上でアプリケーションプログラム『vcp_clear』を実行して、保存されている設定値を消去し、初期値に戻します。

『./vcp_clear』を実行してください。

```
~ # ./vcp_clear ←入力
```

- ⑨ コンソール上に『CLEAR OK』と表示されることを確認してください。

```
~ # ./vcp_clear
CLEAR PROGRAM START
CLEAR OK ←入力
CLEAR PROGRAM END
```

- ⑩ μ ST-SH2 の電源を切ります。

- ⑪ μ ST-VCP の SW1 スイッチを『OFF/OFF/OFF/OFF』に設定します。
 ケース付き μ ST-SH2 + μ ST-VCP ボードを御使用のお客様はケースから μ ST-VCP を取り出して、SW1 スイッチの設定を行ってください。その後、 μ ST-SH2 + μ ST-VCP ボードをケースに組み込んでください。ケースへの組み込み方法は、『 μ ST-VCP ハードウェアマニュアル 5.2 ケースへの組み込み方法』を参照してください。

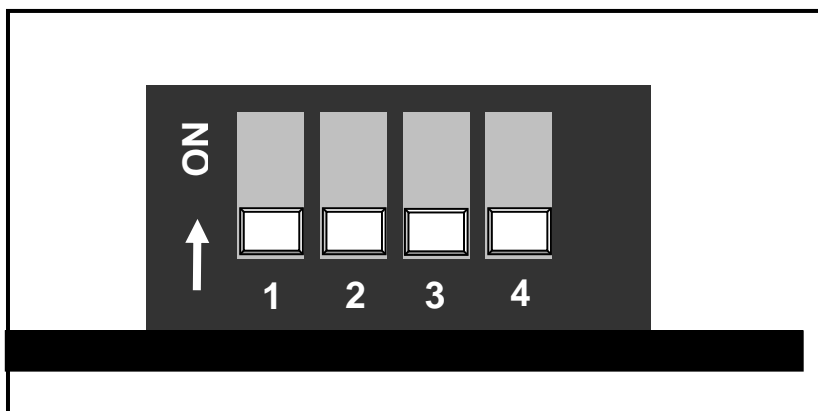


Fig 5.3-2 SW1 の設定

- ⑫ ②～④の動作をもう一度行います。
- ⑬ ゲスト OS 上で作成したビデオデコーダの設定変更サンプルプログラム 1 つを NFS 共有ディレクトリからコピーします。
 『`cp /mnt/nfs/vcp_allread .`』を実行してください。
- ```
~ # cp /mnt/nfs/vcp_allread .
```
- ⑭  $\mu$  ST-SH2 上でアプリケーションプログラム『`vcp_allread`』を実行して、ビデオデコーダの全レジスタの設定値を読み出します。  
 『`./vcp_allread`』を実行してください。
- ```
~ # ./vcp_allread
```
- ⑮ コンソール上に『`READ OK 0x01 = 0x31 ... (以下省略)`』と表示されることを確認してください。

```
~ # ./vcp_allread
ALLREAD PROGRAM START
READ OK 0x00 = 0x13
READ OK 0x01 = 0x31
.
.
(途中省略)
ALLREAD PROGRAM END
```

6. 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。

サンプルアプリケーションなど非 GPL ソフトウェアのソースコードは公開しておりません。

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すので、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, |Greg 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

- ・本文書の著作権は、株式会社アルファプロジェクトが保有します。
- ・本文書の内容を無断で転載することは、一切禁止します。
- ・本文書の内容は、将来予告なしに変更されることがあります。
- ・本文書の内容については万全を期して作成いたしました。万が一不審な点、誤りなどお気付きの点がありましたら弊社までご連絡下さい。
- ・本文書の内容に基づきアプリケーションを運用した結果、万が一損害が発生しても、弊社では一切責任を負いませんのでご了承下さい。

商標について

- ・ SuperH は、ルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
- ・ Linux は、Linus Torvalds の米国およびその他の国における登録商標または商標です。
- ・ Windows® の正式名称は、Microsoft® Windows® Operating System です。
- ・ Microsoft、Windows は、米国 Microsoft Corporation の米国およびその他の国における商標または登録商標です。
- ・ Windows® XP、Windows® Vista、Windows® 7 は、米国 Microsoft Corporation の商品名称です。

本文書では下記のように省略して記載している場合がございます。ご了承下さい。

Windows® XP は WindowsXP もしくは WinXP

Windows® Vista は WindowsVista もしくは WinVista

Windows® 7 は Windows7 もしくは Win7

- ・その他の会社名、製品名は、各社の登録商標または商標です。



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