このページは「24環境」の情報です

第16回 デバイスドライバ入門編(CAT709) †

海老原祐太郎
ebihara@si-linux.com
2003/8/7

Linuxでのデバイスドライバの書き方は、とっつきさえ分かってしまえばさほど
難しいものではありません。今回はCAT709のLEDとDIPSWのドライバを書いてみます。

↑

基本的なこと †

Linuxでのデバイスドライバを書く前に、おさえて置きべき基本的なことをまとめました。

- デバイスドライバのビルドには普通のgcc を使います。
- カーネルソースが必要ですが特殊なパッケージは不要です。
- デバイスドライバは insmod コマンドでカーネルに常駐し、rmmod コマンドで外せます。
  (DOS時代の ADDDRV と DELDRV のようなものです。)
- デバイスドライバはカーネル空間に常駐しますが、ユーザープログラム(プロセス)の
  『サブルーチン』みたいに呼び出されて走行するとイメージすればOKです
- つまりデバイスドライバの各デバイスメソッド関数はプロセスとして走行します。
- linux-2.4現在のところ、カーネル非プリエンプションなので、デバイスドライバメソッドを
  走行中に、プロセスが勝手に切り替わることはありません。
- すなわち

```
while(1)
  ;
```

は際限の無い無限ループに突入して死んでしまいますが、

```
outb(レジスタ番号, インデックスレジスタ);
outb(本ちゃんデータ, データレジスタ);
```

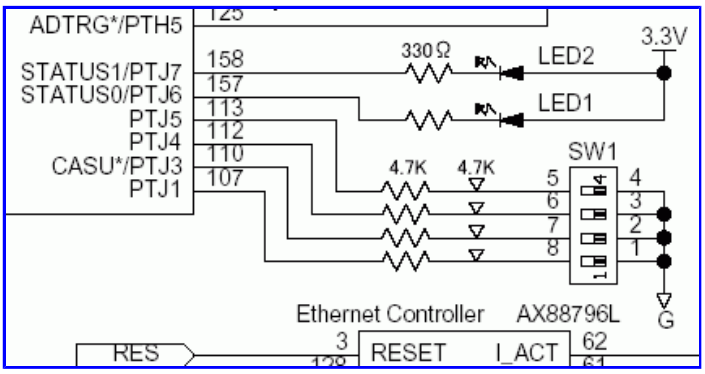
のような処理は必ず連続して実行できるので(間にかってに他のタスクに切り替わったりしない)
プログラマーはある程度楽ができます。

- デバイスドライバは『メジャー番号』と『マイナー番号』の2つの番号を使います。
  linuxカーネルソースの Documentation/devices.txt に一覧表があります。
- メジャー番号は装置の種類を示す番号で、240～254がローカルで使用可能です。
  まずはこの中から番号を選びます。今回は241にしました。
- マイナー番号は、『何枚目のカードなのか』を識別するために使います。
  たとえば同じ PCI の ADコンバータカードが複数毎刺さっていたときに、
  何枚目のカードなのかの識別に使用します。
  今回は、マイナー番号0: dipsw , 1:LED1 , 2:LED2 としました。

↑

CAT709ハードウェア的なこと †

linuxとは離れますが、CAT709のハードウェア的なことをおさらいします。
CAT709の LED と DIPSW はブロック回路図から以下のように抜粋できます。



CPUのポートJ に接続されています。
ポートJのアドレスは0xa4000134です。(詳しくは SH7709 ハードウェアマニュアルに載っています)
ポートJはビットごとに入出力方向を決定できます。
ただし ポートJ0はRAS3L, ポートJ2はCAS3L との共有ピンになっていて、
SDRAMに対する CAS3/RAS3として使用するので、DIPSWやLEDには接続されていません。
LED は 0を書き込むと点灯し、1を書き込むと消灯します。
DIPSWはONになっているビットが0になります。上記の理由によりJ2がとびになっているので注意

↑

コンパイル方法 †

基本的にはデバイスドライバの開発は開発機で行います。
NFSでCAT709からマウントし、ロードと実行はCAT709で行います。

説明するよりもコードを載せたほうが早いので、コードを載せます。
cat709port.c というファイル名で開発機に保存してください。

```
/*
 * cat709port.c
 *   for linux-2.4 kernel
 *   written by Y.Ebihara
 */

/* Makefile は次のようになる


# cat709のカーネルを展開してビルドしたディレクトリを指定する
KERNEL=/home/ebihara/cat709-root-dir/usr/src/linux
CFLAGS=-O2 -I${KERNEL}/include
CC=sh3-linux-gcc

all:cat709port.o

clean:
        rm *.o
*/

#define MODULE
#define __KERNEL__

#define DEFAULT_MAJOR_NUM   241
#define DEVNAME             "cat709port"

/* control register */
#define PCCR 0xa4000104
#define PDCR 0xa4000106
#define PLCR 0xa4000114

/* data register */
#define PCDR 0xa4000124
#define PDDR 0xa4000126
#define PJDR 0xa4000130
#define PLDR 0xa4000134

enum miscport_minor{
  DIPSW,    // 0
  LED1,     // 1
```

```
  LED2,    // 2
  MINOR_MAX
};

#include <linux/module.h>
#include <linux/fs.h>
#include <asm/io.h>
#include <asm/uaccess.h>

/* ------------------------------------------------------- */
/* I/O access                                              */
/* 実際のI/O操作を行う関数                                 */
/* ------------------------------------------------------- */

// inb(), outb() はcat709_port_map()によるアドレス変換の対象となる
// 一方、ctrl_outb(), ctrl_inb()はアドレス変換の対象とならない。
// 従って物理番地を直接書きたいときは ctrl_ を使う (速度も速い)
//   unsigned char ctrl_inb(unsigned long 物理番地)
//   ctrl_outb(unsigned char データ,  unsigned long 物理番地)

static void init_cat709port(){
  // initialize
}


/* LED OUTPUT */
/* led=1 or 2, data= 1 or 0 */
static void cat709_led(int led, int data){
  int x;
  led &= 0x3;
  led <<= 6;
  x=ctrl_inb(PJDR);
  if(data)
    x &= ~led;
  else
    x |= led;
  ctrl_outb(x,PJDR);
}


/* DIPSW INPUT */
static int cat709_dipsw(void){
  int x1,x2;
  x1 = ~ctrl_inb(PJDR) >> 1;
  x2 = x1 >> 1;
  return (x1 & 0x01) | (x2 & 0x0e);
}

/* ------------------------------------------------------- */
/* file operation method                                   */
/* キャラクター型デバイスドライバ                         */
/* ------------------------------------------------------- */

// 大雑把に言うと ユーザープロセスはデバイスファイルを「ファイルとみなして」
// open() し read(), write()を行う。
// デバイスメソッドは プロセスから呼び出される『サブルーチン』と思えば
// 大体あっている
// 従って open()やread(), write() は カーネル空間のテキストだが
// 『pidを持ったまま』プロセスのコンテキストとして走行する。


// このドライバをプロセスが open() したときに呼び出される

static int cat709port_open(struct inode *inode, struct file *filp){
  int minor;
  // マイナー番号取得マクロ
  minor=MINOR(inode->i_rdev);
  if(minor > MINOR_MAX){
    printk("<1>cat709port open() error");
    return -ENODEV;
  }

  // モジュールの『利用度』を１あげる
  MOD_INC_USE_COUNT;

  // filp->private_data は void* 型の変数で、プロセンス単位(openされる毎)の
  // データは private_data に保存する。
  // *filpは すべてのデバイスメソッドで引数としてやってくるので
```

```
  // filp->private_data はすべてのデバイスメソッドから参照することができる。
  // 変数がひとつで足りないときは 構造体の形をあらかじめ定義しておいて
  // open()時にkmalloc() で構造体の実態を作り、ポインタを private_data に格納する。
  // もちろん デバイスをclose() するときに kfree()することを忘れないように
  filp->private_data = (void*)minor;
  return 0;
}

static int cat709port_close(struct inode *inode, struct file *filp){
  // モジュールの『利用度』を１下げる
  MOD_DEC_USE_COUNT;
  return 0;
}

static int cat709port_read(struct file *filp, char *user_buf,size_t  count,loff_t *ppos){
  char buffer[100];
  int len;
  int data;
  static int  flag=0;

  if(flag==0){
    flag=1;
    switch((int)(filp->private_data)){
    case DIPSW:
      data=cat709_dipsw();
      len=sprintf(buffer,"%d¥n",data);
      // copy_to_user(転送先,転送元,バイト長) で
      // ユーザープロセスにデータを返す
      copy_to_user(user_buf,buffer,len);
      return len;  // 転送バイト数を返す
    }
  }
  flag=0;
  return 0;
  // 戻り値は
  //   負:エラー
  //   0 :end-of-file に到達
  //   正:転送したバイト数
  // としなければならない
  // flag変数で１回おきに0を返すことによって、catコマンドで読んだときに
  // 1回目の呼び出しはデータを取得し、2回目の呼び出しでeofになるので終了する
}


static int cat709port_write(struct file *filp, const char *user_buf,size_t  count,loff_t *ppos){
  char buffer[100];
  int data;

  switch((int)(filp->private_data)){
  case LED1:
    copy_from_user(buffer,user_buf,1);
    if(buffer[0]=='0')
      cat709_led(1, 0);
    else if(buffer[0]=='1')
      cat709_led(1, 1);
    return count;
  case LED2:
    copy_from_user(buffer,user_buf,1);
    if(buffer[0]=='0')
      cat709_led(2, 0);
    else if(buffer[0]=='1')
      cat709_led(2, 1);
    return count;

  }
  return count;
  // 戻り値は
  //   負:エラー
  //   0 :end-of-file に到達
  //   正:転送したバイト数
  // としなければならない
  // write()の時は、ユーザープロセスが書き出ししたかったバイト数(count)を
  // 全部返してあげるのでユーザープロセスは大満足する(^-^)
}

// file_operations構造体とはジャンプテーブルやベクターテーブルのようなもので
// デバイスファイルを open() read() write() close() されたときに呼び出す
// 関数一覧である
```

メニュー

- 会社トップ
- Wikiトップページ
- ソフトウェア情報
- ハードウェア情報
- 組み込みLinux-Tips
- ご購入はこちら

最新の0件

```
static struct file_operations cat709port_fops = {
  owner:THIS_MODULE,            // おまじない
  read:cat709port_read,         // read
  write:cat709port_write,       // write
  open:cat709port_open,         // open
  release:cat709port_close      // close
};


/* ------------------------------------------------------ */
/* driver module load / unload */
/* ------------------------------------------------------ */

// モジュールがロードされたとき(insmod されたとき)は
// init_module()が呼ばれる
// 戻り値は
//   負:エラー発生
//   0 :正常終了
// とする。負を返すとモジュールは常駐しない

static int init_module(void){
  printk("<1>cat709port hello¥n");

  // register_chrdev(メジャー番号, デバイス名,
  //                 ファイルオペレーションズ構造体へのポインタ)
  // で、カーネルに対してキャラクタ型デバイスドライバの登録を行う
  if(register_chrdev(DEFAULT_MAJOR_NUM,DEVNAME,&cat709port_fops) != 0){
    printk("<1>cat709port register fail¥n");
    return -1;
  }
  // ハードウェア的な初期化
  init_cat709port();
  return 0;
}

// モジュールがアンロードされたときき (rmmod されたとき)は
// cleanup_module()が呼び出される

static void cleanup_module(void){
  printk("<1>cat709port goodby¥n");
  unregister_chrdev(DEFAULT_MAJOR_NUM,DEVNAME);
}

MODULE_LICENSE("GPL");

/*
ちなみにこのドライバの使い方は以下の通り

■ 準備

# rommode rw
# mkdir /lib/modules/2.4.21/kernel/drivers/misc
# cp cat709port.o /lib/modules/2.4.21/kernel/drivers/misc
# depmod -a
# modprobe cat709port

メジャー番号241, マイナー番号0
# mknod /dev/cat709dipsw c 241 0
# chmod 444 /dev/cat709dipsw

メジャー番号241, マイナー番号1
# mknod /dev/cat709led1 c 241 1
# chmod 666 /dev/cat709led1

メジャー番号241, マイナー番号2
# mknod /dev/cat709led2 c 241 2
# chmod 666 /dev/cat709led2

■ 使い方

$ cat /dev/cat709dipsw
0〜15が読める

$ echo 1 > /dev/cat709led1
LED1が点灯する

$ echo 0 > /dev/cat709led2
```

```
LED2が消灯する

*/
```

Makefileは以下のようになります。
KERNEL=行は、CAT709のカーネルを展開し、config し、make したことのあるディレクトリを指定します（超重要）

```
KERNEL=/home/ebihara/cat709-root-dir/usr/src/linux
CFLAGS=-O2 -I${KERNEL}/include
CC=sh3-linux-gcc

all:cat709port.o

clean:
        rm *.o      # rmの前にはTABを入れてね
```

コンパイルは make するだけです。
出来上がった cat709port.o がデバイスドライバになります。
fileコマンドで確かめると Hitachi SH の relocateble（再配置可能）オブジェクトになっていることが分かります。

```
$ make
$ file cat709port.o
cat709port.o: ELF 32-bit LSB relocatable, Hitachi SH, version 1 (SYSV), not stripped
```

↑

デバイスドライバのロードと実行 †

以下の作業は CAT709 で行います。

```
ドライバを組み込む
# insmod /NFSマウント/開発機のどこかのディレクトリ/cat709port.o
cat709port hello

組み込まれたか確認する
# lsmod
Module                  Size  Used by    Not tainted
cat709port              1468   0
ds                      6332   0  (unused)
sh3_ss                  3212   3
pcmcia_core            34880   0  [ds sh3_ss]

メジャー番号を調べる
# cat /proc/devices
Character devices:
  1 mem
  2 pty
  3 ttyp
  4 ttyS
  5 cua
 10 misc
 90 mtd
108 ppp
128 ptm
136 pts
162 raw
204 ttySC
205 cusc
241 cat709port   ←241番に登録された
254 pcmcia
```

そうしたらデバイスファイルも作ります。
デバイスファイルへのファイル操作が、デバイスドライバへの操作になります。

```
# rommode rw

メジャー番号241, マイナー番号0
# mknod /dev/cat709dipsw c 241 0
# chmod 444 /dev/cat709dipsw

メジャー番号241, マイナー番号1
```

```
# mknod /dev/cat709led1 c 241 1
# chmod 666 /dev/cat709led1

メジャー番号241, マイナー番号2
# mknod /dev/cat709led2 c 241 2
# chmod 666 /dev/cat709led2

# rommode ro
```

実際に使ってみます

```
$ cat /dev/cat709dipsw
0～15が読める

$ echo 1 > /dev/cat709led1
LED1が点灯する

$ echo 0 > /dev/cat709led2
LED2が消灯する
```

↑

組 込 み †

ドライバの動作検証ができたのでCAT709に保存します。
出来上がったドライバは、/lib/modules/カーネルバージョン/kernel/drivers/misc/
ディレクトリに保存します。最初 misc ディレクトリが無いので作っておきます。

以下の作業は CAT709で行います。

```
# rommode rw
# mkdir /lib/modules/2.4.21/kernel/drivers/misc
# cp cat709port.o /lib/modules/2.4.21/kernel/drivers/misc
# depmod -a
# vi /etc/modules

1行追加する
cat709port


# rommode ro
```

/etc/modules ファイルにモジュール名を追加しておくと、
起動時に自動的にロードしてくれます。
ただし、モジュール名とファイル名を関連付けるため、
depmod -a を1回実行しておく必要があります(超重要)

↑

詳しくは †

ここで取り上げたのはlinuxデバイスドライバの一番簡単な例です。
勉強するなら オライリー出版 のLINUXデバイスドライバ 第2版 が最適です。左のリンクで購入できます。

Last-modified: 2006-01-19 (木) 16:40:04