PCI-3133 Board Support Package Installation on RedHawk

Release Notes Revision B

September 9, 2022

concurrent
REAL-TIME

1. はじめに

本書は、Concurrent Real Time Inc(CCRT)の RedHawk 上で動作する、インターフェース社製 PCI-3133 PCI ボードサポートパッケージ 用リリースノートです。

2. インストールのための条件

PCI- 3133 BSP をインストールするためには、以下の製品がインストールされている事が必要です。

- PCI-3133 ボード
- RedHawk 6.x 以上
- Extmem version 8.3 以上

PCI-3133は、PCIバスに準拠した、12ビットAD変換製品です。

3. インストール方法

PCI-3133 BSP は、IRQ 共有するように設計されています。もしこのデバイスの IRQ が、別のデバイスによって共有されている場合に、このドライバの性能は損なわれる場合があります。そのため、可能な限り、このボードはその IRQ が他の装置と共有されていないPCIスロットの中に実装する事が奨励されます。“lspci -v”コマンドをシステムで種々の装置の IRQ を確認するために使用することができます。

PCI-3133 BSP は、CDROM/DVD 上の RPM/DEB フォーマットで供給され、別途 extmem デバイスドライバがインストールされていることが必要です。

以下に、インストールの手順を示します。:

x86_64 アーキテクチャの場合

=== root ユーザで実行してください===

mount /dev/cdrom /mnt あるいは mount /dev/dvd /mnt

cd /mnt

もし、extmem を同時にインストールする場合には、以下のコマンドを入力してください

rpm -ivh bin-extmem-**X.Y**_RH***x.y-z***.x86_64.rpm

PCI3133 BSP 実行パッケージのインストール

rpm -ivh bin-pci3133 -**X.Y**_RH***x.y-z***.x86_64.rpm

もし必要であれば、続けて開発パッケージのインストールを行ってください

rpm -ivh dev-pci3133 -**X.Y**_RH***x.y-z***.x86_64.rpm

umount /mnt

amd64 アーキテクチャの場合

=== root ユーザで実行してください===

mount /dev/cdrom /mnt あるいは mount /dev/dvd /mnt

cd /mnt

もし、extmem を同時にインストールする場合には、以下のコマンドを入力してください

```
# apt install ./bin-extmem-rhx.y_X.Y_amd64.deb
```

PCI3133 BSP 実行パッケージのインストール

apt install ./bin-pci3133 -rhx.y_X.Y_amd64.deb

もし必要であれば、続けて開発パッケージのインストールを行ってください

apt install ./dev-pci3133 -rhx.y_X.Y_amd64.deb

umount /mnt

arm64 アーキテクチャの場合

=== root ユーザで実行してください===

mount /dev/cdrom /mnt あるいは mount /dev/dvd /mnt

cd /mnt

もし、extmem を同時にインストールする場合には、以下のコマンドを入力してください

```
# apt install ./bin-extmem-rhx.y_X.Y_arm64.deb
```

PCI3133 BSP 実行パッケージのインストール
apt install ./bin-pci3133 -rhx.y_X.Y_arm64.deb

もし必要であれば、続けて開発パッケージのインストールを行ってください
apt install ./dev-pci3133 -rhx.y_X.Y_arm64.deb
umount /mnt

(***x.y*** は RedHawk のバージョン番号であり、 6.x,7.x または 8.x で、**X.Y** は、BSP のバージョン、**z** は、BSP のリリース番号を示し、予告なく変更することがあります。)

PCI-3133 BSP パッケージは ***/usr/local/CNC/drivers/extmem/interface/pci3133*** ディレクトリにインストールされ、必要な場所に展開されます。

4. アンインストール方法

PCI-3133 BSP パッケージは、以下のコマンドでアンインストールします。この作業により ***/usr/local/CNC/drivers/extmem/interface/ pci3133*** ディレクトリは削除されます。

x86_64 アーキテクチャの場合
=== root ユーザで実行してください===
開発パッケージをインストールしていた場合には、
rpm -e dev-pci3133 -**X.Y**_RH***x.y-z***.x86_64 *(開発パッケージの削除)*
rpm -e bin-pci3133 -**X.Y**_RH***x.y-z***.x86_64 *(実行パッケージの削除)*
実行パッケージのみをインストールしていた場合には、
rpm -e bin-pci3133 -**X.Y**_RH***x.y-z***.x86_64 *(実行パッケージの削除)*

amd64 アーキテクチャの場合
=== root ユーザで実行してください===
開発パッケージをインストールしていた場合には、
apt purge dev-pci3133 -rhx.y *(開発パッケージの削除)*
apt purge bin-pci3133 -rhx.y *(実行パッケージの削除)*
実行パッケージのみをインストールしていた場合には、
apt purge bin-pci3133 -rhx.y *(実行パッケージの削除)*

arm64 アーキテクチャの場合
=== root ユーザで実行してください===
開発パッケージをインストールしていた場合には、
apt purge dev-pci3133 -rhx.y *(開発パッケージの削除)*
apt purge bin-pci3133 -rhx.y *(実行パッケージの削除)*
実行パッケージのみをインストールしていた場合には、
apt purge bin-pci3133 -rhx.y *(実行パッケージの削除)*

5. ライブラリマニュアル

ライブラリマニュアルは、オンラインで提供されます。
# man pci3133
pci3133(3)　　pci3133(3)

NAME
　pci3133 - external memory device access library

SYNOPSIS
　[ボードの詳細は、各マニュアルを見てください]

DESCRIPTION
　pci3133 は、external memory ドライバを利用した pci3133 ボードアクセスライブラリです。

　#include <sys/pci3133.h>
　gcc [options ...] file -lpci3133 -lextmem ...

　**********
　PCI3133
　**********

　DIP スイッチの読み込み
　int pci3133_get_sw(int fd,unsigned int *data);
　　戻り値
　　　エラーなら-1 成功なら 0
　　引数
　　　fd　ファイルディスクリプタ番号
　　　data　　出力変数へのポインタ

　割り込みハンドラの登録
　int pci3133_setup_signal
　　(
　　　int fd,
　　　void (*interrupt_hadler)(int, siginfo_t *, void *)),
　　　int mask
　　);
　　戻り値
　　　エラーなら-1 成功なら 0
　　引数
　　　fd　ファイルディスクリプタ番号
　　　void (*interrupt_hadler)(int, siginfo_t *, void *)) 割り込みハンドラ
　　　mask 割り込みを許可するビットマスク 以下のいずれかを指定する
　　　PCI3133_IMASK_TMR インターバルタイマー
　　　PCI3133_IMASK_BSY AD 変換終了割り込み
　　　PCI3133_IMASK_TRG 外部割り込み(EXINT IN)
　　　PCI3133_IMASK_DIV 標準分周器割り込み
　　　PCI3133_IMASK_ALL
(PCI3133_IMASK_TMR|PCI3133_IMASK_BSY|PCI3133_IMASK_TRG|PCI3133_IMASK_DIV)

デバイスの非初期化処理
int pci3133_reset(int fd);
int pci3133_uninit(int fd);　　戻り値
　　エラーなら-1 成功なら 0
　引数
　　fd ファイルディスクリプタ番号
　2つの関数は同じ処理、全ての制御レジスタに 0 値を設定する。

デバイスの初期化処理
int pci3133_init
　(
　　int fd,
　　int option
　);
　戻り値
　　エラーなら-1 成功なら 0
　引数
　　fd ファイルディスクリプタ番号
　　option 1を指定すると以下の情報が表示される
　　BAR0 I/O Region addr 0x00004480 offset 0x00000000 16 bytes
Switch 1

割り込みサービス関数 割り込んだ際の割り込み要因レジスタ(オフセット 0x05)
の値を戻す
int pci3133_intr_service
　(
　　int fd,
　　unsigned int *iflag,
　　int *pending
　);

　戻り値
　　エラーなら-1 成功なら 0
　引数
　　fd ファイルディスクリプタ番号
　iflag 値を戻す変数
　pending 保留されている割り込みの数を戻す変数

割 り込んだ際の割り込み要因レジスタ(オフセット 0x05)の値を戻し、AD データ
を読み込む
関数とペアで使用する（下記使用例を参照）
int pci3133_intr_service_and_read
　(
　　int fd,
　　unsigned int *iflag,
　　int *pending
　　unsigned short int *data
　);

　戻り値
　　エラーなら-1 成功なら 0
　引数
　　fd ファイルディスクリプタ番号
　iflag 値を戻す変数

pending 保留されている割り込みの数を戻す変数
data AD 値を戻す変数(最上位ビットは変換終了フラグ)

割り込みを禁止する

```
int pci3133_disable_intrrupt
        (
            int fd,
            int mask
        );
```

戻り値

エラーなら-1 成功なら 0

引数

fd ファイルディスクリプタ番号

mask 割り込みを禁止するビットマスク 以下のいずれかを指定する

PCI3133_IMASK_TMR インターバルタイマー

PCI3133_IMASK_BSY AD 変換終了割り込み

PCI3133_IMASK_TRG 外部割り込み(EXINT IN)

PCI3133_IMASK_DIV 標準分周器割り込み

PCI3133_IMASK_ALL
(PCI3133_IMASK_TMR|PCI3133_IMASK_BSY|PCI3133_IMASK_TRG|PCI3133_IMASK_DIV)

割り込みを許可する

```
int pci3133_enable_intrrupt
        (
            int fd,
            int mask
        );
```

戻り値

エラーなら-1 成功なら 0

引数

fd ファイルディスクリプタ番号

mask 割り込みを禁止するビットマスク 以下のいずれかを指定する

PCI3133_IMASK_TMR インターバルタイマー

PCI3133_IMASK_BSY AD 変換終了割り込み

PCI3133_IMASK_TRG 外部割り込み(EXINT IN)

PCI3133_IMASK_DIV 標準分周器割り込み

PCI3133_IMASK_ALL
(PCI3133_IMASK_TMR|PCI3133_IMASK_BSY|PCI3133_IMASK_TRG|PCI3133_IMASK_DIV)

インターバルタイマーをセットする

```
int pci3133_set_interval_timer(int fd,unsigned  int  base,unsigned  int
div);
```

戻り値

エラーなら-1 成功なら 0

引数

fd ファイルディスクリプタ番号

base ベースクロック値 以下のいずれかを指定する

PCI3133_TIMER_BASE_STOP	停止
PCI3133_TIMER_BASE_010USEC	10 マイクロ秒
PCI3133_TIMER_BASE_100USEC	100 マイクロ秒
PCI3133_TIMER_BASE_001MSEC	1 ミリ秒
PCI3133_TIMER_BASE_010MSEC	10 ミリ秒
PCI3133_TIMER_BASE_100MSEC	100 ミリ秒

div ベースクロックを分周する値 カウントダウンし 0 の時割り込みが発生する

最大15分周しかできない
注意
このタイマーでは　AD 変換は開始しない
変換をさせるためには、pci3133_set_convert_timer()を使用すること

インターバルタイマーの現在値を読み出す
int pci3133_get_interval_timer(int fd,unsigned int *count);
戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
count 値を読み出す変数へのポインタ

汎用関数 オフセット値を指定してレジスタの値を読み出す
int pci3133_get_ioport(int fd,int offset,unsigned int *value);
戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
offset レジスタオフセット
value 値を読み出す変数へのポインタ

汎用関数 オフセット値を指定してレジスタに値を書き出す
int pci3133_set_ioport(int fd,int offset,unsigned int *value);
戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
offset レジスタオフセット
value 値を出す変数へのポインタ

チャネルを指定して入力データを読み出す(変換終了フラグをポーリングする)
int pci3133_read_data_poll(int fd,int ch,unsigned short int *data);
戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
ch チャンネル
data 入力変数へのポインタ

入力データを読み出す
int pci3133_read_data(int fd,unsigned short int *data);
戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
data 出力変数へのポインタ

チャネル切り替え＋AD 変換開始
int pci3133_set_channel(int fd,int ch);
戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
ch チャンネル

同期サンプリング設定
int pci3133_set_sync(int fd,unsigned int data);
戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
data 以下のいずれかを指定する
PCI3133_SYNC_NORMAL 複数枚同期サンプリングを使用しない場合
(同期信号はスルーされる)
PCI3133_SYNC_MASTER 複数枚同期サンプリングを使用する
(同期信号を出力するマスターになる)
PCI3133_SYNC_SLAVE 複数枚同期サンプリングを使用する
(同期信号を入力するスレーブになる)

割り込みのトリガー設定
int pci3133_set_trigger(int fd,unsigned int trigger);
戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
trigger 以下のいずれかを指定する
PCI3133_TRIG_DI DI トリガー(1:有効/0:無効)
PCI3133_TRIG_TMST タイマーによる AD 変換スタート有効
PCI3133_TRIG_EXINT EXINT IN 入力(立ち上がりエッジ:1/立 ち
下がりエッジ:0)有効
PCI3133_TRIG_EXTRG EXTRG IN 入力(立ち上がりエッジ:1/立ち
下がりエッジ:0)有効
PCI3133_TRIG_NONE なし

AD 変換タイマーをセットする
int pci3133_set_convert_timer(int fd,unsigned int div1,unsigned int
div2);
戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
div1 8MHz のベースクロックを分周する値 カウントダウンする
最大 65535 分周しかできない
div2 div1 の出力を分周する値 カウントダウンし 0 の時割り込みが
発生する
最大 65535 分周しかできない

この間数を呼び出すと割り込みのロジックが変更になり、割り込み処理で
データを読み出すため、
割 り込み関数では、pci3133_intr_service_and_read()を必ず使用するこ
と

AD 変換タイマーをスタートまたは停止する
int pci3133_set_gate(int fd,unsigned int on);

戻り値
エラーなら-1 成功なら 0
引数
fd ファイルディスクリプタ番号
on 以下のいずれかを指定する
PCI3133_GATE_ON 0x01 //タイマーイネーブル
PCI3133_GATE_OFF 0x00 //タイマーディセーブル
使用例

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <memory.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <sys/file.h>
#include <sys/types.h>
#include <sys/extmem.h>
#include <signal.h>
#include <stdlib.h>
#include <sys/libinterface.h>

#define MAXDATA     1600
int  fd;
int count=0;
int error=0;
unsigned short DATA[MAXDATA];

static void pci3133_interrupt_handler(int signo)
{
          int pending=0;
          unsigned int iflag=0;
        static int ch=0;

        pci3133_intr_service_and_read(fd ,&iflag,&pending,&DATA[count]);
        if ((DATA[count]&0x8000)==0)
        {
                error++;
                DATA[count]=0xFFFF;
        }
        ch ++;ch %= 16;
        pci3133_set_channel(fd,ch);
        if (count<MAXDATA) count++;
}

int
main(int argc, char **argv)
{
        char devname[1024];
        int mem_size;
        int ch,value;
        static int cycle=1000,Us;
        unsigned short int data;
        int i;

          extern int optind,errno;
          extern char *optarg;
```

```
    int     c;

    while((c = getopt(argc, argv, "s:")) != EOF)
  {
        switch(c) {
            case 's':
                cycle = atoi( optarg ) ;
                if (( cycle <= 0 )||((cycle)>1666))
                {
                    fprintf(stderr,"INVALID          ARGU-
MENT!!0) ;
                    exit( 0 ) ;
                }
            break;
            default:
                fprintf(stderr,          "Usage:          %s
[options]0,argv[0]);
                fprintf(stderr, "Options:0);
                fprintf(stderr,"          -s  msec(msec: 0  or
1<=cycle<=16)0);
                exit(-1);
        }
    }
    if (optind == argc) {
        /* no more option */
        strcpy(devname,"/dev/pci3133/0");
    } else {
        strcpy(devname,argv[optind]);
    }

    if ((fd = open(devname,O_RDWR)) == -1)
  {
      fprintf(stderr,"Device not found %s(%s)0,
      devname,strerror(errno));
          exit(0);
  }
  printf("%s ",devname);
  if (pci3133_init(fd,1)<0)
  {
      fprintf(stderr,"Device initialize error %s(%s)0,
      devname,strerror(errno));
          exit(0);
  }

  ch =0;
  value = PCI3133_AI_MODE_SINGLEEND;
  pci3133_set_ioport(fd,PCI3133_AD_OFFSET1,&value);

  pci3133_set_channel(fd,0);
  Us = 1000000/cycle;
  printf("%d Hz is %d     us0,cycle,Us);
    if (pci3133_set_convert_timer(fd,8,Us)<0)
    {
        fprintf(stderr,"Interrupt initialize error %s(%s)0,
        devname,strerror(errno));
        exit(0);
    }

    if (pci3133_set_trigger(fd,PCI3133_TRIG_TMST)<0)
    {
```

```
                fprintf(stderr,"Interrupt initialize error %s(%s)0,
                devname,strerror(errno));
                exit(0);
        }
        if
(pci3133_setup_signal(fd,pci3133_interrupt_han-
dler,PCI3133_IMASK_BSY)<0)
        {
                fprintf(stderr,"Interrupt initialize error %s(%s)0,
                devname,strerror(errno));
                exit(0);
        }
        pci3133_set_gate(fd,PCI3133_GATE_ON);
        for(i=0;count<MAXDATA; i++)
        {
                sleep(1); ",count);
                printf("%d
        }
        pci3133_set_gate(fd,PCI3133_GATE_OFF);

        pci3133_disable_intrrupt(fd ,PCI3133_IMASK_ALL);

        if (pci3133_uninit(fd)<0)
        {
                 fprintf(stderr,"%s0,strerror(errno));
        }
        close(fd);
        if (error)
        {
                for(i=0;i<count;i++)
                {
                    printf("count %d:%04X0,i,DATA[i]);
                }
        }
        printf("Error %d0,error);
        printf("Success %d0,count-error);
}
```

SEE ALSO

/usr/local/CNC/drivers/extmem/interface/pci3133 下のプログラム

AUTHORS