PCI-3346A Board Support Package Installation on RedHawk

Release Notes Revision B

Sepetember 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- 3346A 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- 3346A 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- 3346A ボード
- RedHawk 6.x 以上
- Extmem version 8.3 以上

PCI-3346Aは、PCIバスに準拠した、12ビットDA変換機能を持った製品です。

3. インストール方法

PCI-3346A 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-3346A 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

PCI3346A BSP 実行パッケージのインストール

rpm -ivh bin-pci3346a -**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-pci3346a -**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
```

PCI3346A BSP 実行パッケージのインストール

apt install ./bin-pci3346a -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-pci3346a -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
```

PCI3346A BSP 実行パッケージのインストール
apt install ./bin-pci3346a -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-pci3346a -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-3346A BSP パッケージは ***/usr/local/CNC/drivers/extmem/interface/pci3346a*** 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-3346A 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/pci3346a 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-pci3346a -**X.Y**_RH***x.y-z***.x86_64 *(開発パッケージの削除)*
rpm -e bin-pci3346a -**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-pci3346a -**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-pci3346a -rhx.y *(開発パッケージの削除)*
apt purge bin-pci3346a -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-pci3346a -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-pci3346a -rhx.y *(開発パッケージの削除)*
apt purge bin-pci3346a -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-pci3346a -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 pci3346a
pci3346a(3) pci3346a(3)

NAME
pci3346a - 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
pci3346a は、external memory ドライバを利用した pci3346a ボードアクセスライブラリです。

#include <sys/pci3346a.h>
gcc [options ...] file -lpci3346a -lextmem ...

****
PCI3346
****

DIP スイッチの読み込み
int pci3346a_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

****
PCI3346a
****

割り込みハンドラの登録
int pci3346a_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
PCI3346A_IMASK_TMR インターバルタイマー
PCI3346A_IMASK_SPS DA 変換開始割り込み

PCI3346A_IMASK_TRG 外部割り込み(EXINT IN)
PCI3346A_IMASK_ALL
(PCI3346A_IMASK_TMR|PCI3346A_IMASK_TRG|PCI3346A_IMASK_SPS)

デバイスの非初期化処理
int pci3346a_reset(int fd);
int pci3346a_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 pci3346a_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 0x0000a8c0 offset 0x00000000 32 bytes
Switch 0

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 0x0D)
の値を戻す
int pci3346a_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

割り込みを禁止する

int pci3346a_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
PCI3346A_IMASK_TMR インターバルタイマー

PCI3346A_IMASK_SPS DA 変換開始割り込み
PCI3346A_IMASK_TRG 外部割り込み(EXINT IN)
PCI3346A_IMASK_ALL
(PCI3346A_IMASK_TMR|PCI3346A_IMASK_TRG|PCI3346A_IMASK_SPS)

割り込みを許可する
int pci3346a_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
PCI3346A_IMASK_TMR インターバルタイマー
PCI3346A_IMASK_SPS DA 変換開始割り込み
PCI3346A_IMASK_TRG 外部割り込み(EXINT IN)
PCI3346A_IMASK_ALL
(PCI3346A_IMASK_TMR|PCI3346A_IMASK_TRG|PCI3346A_IMASK_SPS)

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 pci3346a_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 pci3346a_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

データを同時出力モードで出力する
int pci3346a_write_data_all(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
ch チャンネル
data 出力変数へのポインタ

チャネルを指定してデータを出力する
int pci3346a_write_data(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 出力データ

出力データを読み出す
int pci3346a_read_data(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 出力変数へのポインタ

チャネル切り替え
int pci3346a_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 pci3346a_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
PCI3346A_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)
PCI3346A_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)
PCI3346A_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 pci3346a_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
PCI3346A_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
PCI3346A_TRIG_NONE なし

DA 変換タイマーをセットする
int pci3346a_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

DA 変換タイマーをスタートまたは停止する

int pci3346a_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

PCI3346A_GATE_ON　0x01　//タイマーイネーブル

PCI3346A_GATE_OFF　0x00　// タイマーディセーブル

DA 変換のレンジをチャンネル指定して設定する/読み出す

int pci3346a_set_channel_range(int fd,int ch,int range);

int pci3346a_get_channel_range(int fd,int ch,int *range);

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

range　以下のいずれかを指定する

PCI3346A_RANGE_SETTING_0TOP10	0V から +10V
PCI3346A_RANGE_SETTING_M5TOP5	-5V から +5V
PCI3346A_RANGE_SETTING_M10TOP10	-10V から +10V

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 int DATA[MAXDATA];
```

```
static void pci3346a_interrupt_handler(int signo, siginfo_t *info, void *ptr)
{
          int pending=0;
          unsigned int iflag=0;
          static int ch=0;

          pci3346a_intr_service(fd ,&iflag,&pending);
          pci3346a_set_channel(fd,ch);
          ch ++;ch %= 16;
        pci3346a_write_data(fd,ch,DATA[count]);

          if (count<MAXDATA) count++;
}
int main( int argc, char **argv)
{
        int      ch,ret;
        static int cycle=1000,Us;
        char devname[1024];
        unsigned int value;
        int i,error=0;
        unsigned short int data[16];

          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 ARGUMENT!!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/pci3346a/0");
          } else {
                  strcpy(devname,argv[optind]);
          }
          for(i=0;i<MAXDATA; i++)
          {
               DATA[count]=i;
          }


          if ((fd = open(devname,O_RDWR)) == -1)
        {
                fprintf(stderr,"Device not found %s(%s)0,
                devname,strerror(errno));
```

```
                        exit(0);
        }
        printf("%s ",devname);
        if (pci3346a_init(fd,1)<0)
        {
                fprintf(stderr,"Device initialize error %s(%s)0,
                devname,strerror(errno));
                        exit(0);
        }
        ret = pci3346a_set_sync(fd,PCI3346A_SYNC_NORMAL);
        if(ret) fprintf(stderr,"pci3346a_set_sync() error0);
        for(ch=0;ch<16;ch++)
        {
                //ret =  pci3346a_set_channel_range(fd,ch,PCI3346A_RANGE_SETTING_0TOP10);
                //ret=pci3346a_set_channel_range(fd,ch,PCI3346A_RANGE_SETTING_M5TOP5);
                ret= pci3346a_set_channel_range(fd,ch,PCI3346A_RANGE_SETTING_M10TOP10);
                if(ret) fprintf(stderr,"pci3346a_set_channel_range() error0);
                //ret = pci3346a_get_channel_range(fd,ch,&value);  printf("ch
%d %x0,ch,value);
        }


        value = PCI3346A_CV_MODE_ALLDISABLE;
        ret = pci3346a_set_ioport(fd,PCI3346A_CV_MODE_OFFSET,&value);
        if(ret) fprintf(stderr,"pci3346a_set_ioport() error0);

          value = PCI3346A_AOUT_ON;
          ret = pci3346a_set_ioport(fd,PCI3346A_AOUT_OFFSET,&value);
        if(ret) fprintf(stderr,"pci3346a_set_ioport() error0);

          for(ch=0;ch<16;ch++)
          {
                pci3346a_write_data(fd,ch,0x00F|ch<<4);
                     pci3346a_read_data(fd,ch,&data[ch]);
                if ( data[ch] != (0x00F|ch<<4))
                {
                    error++;
                    fprintf(stderr,"ch%d data error %X0,ch,data[ch]);
                }
          }

        value = PCI3346A_CV_MODE_ALLOUT;
        ret = pci3346a_set_ioport(fd,PCI3346A_CV_MODE_OFFSET,&value);
        if(ret) fprintf(stderr,"pci3346a_set_ioport() error0);

          for(ch=0;ch<16;ch++)
                data[ch]=0xCC0|ch;
        ret = pci3346a_write_data_all(fd,data);
        if(ret) fprintf(stderr,"pci3346a_write_data_all() error0);

          for(ch=0;ch<16;ch++)
          {
                     ret = pci3346a_read_data(fd,ch,&data[ch]);
                if(ret) fprintf(stderr,"pci3346a_read_data() error0);
                if ( data[ch] != (0xCC0|ch))
                {
                    error++;
                    fprintf(stderr,"ch%d data error %X0,ch,data[ch]);
                }
          }
```

```
        printf("Data read/write test results.0);
          printf("Error %d0,error);

        error =0;
          Us = 1000000/cycle;
          printf("%d Hz is %d  us0,cycle,Us);
          if (pci3346a_set_convert_timer(fd,8,Us)<0)
          {
                  fprintf(stderr,"Interrupt initialize error %s(%s)0,
                  devname,strerror(errno));
                  exit(0);
          }
          if        (pci3346a_setup_signal(fd,pci3346a_interrupt_handler,PCI3346A_IMASK_TMR)<0)
          {
                  fprintf(stderr,"Interrupt initialize error %s(%s)0,
                  devname,strerror(errno));
                  exit(0);
          }
          pci3346a_set_gate(fd,PCI3346A_GATE_ON);
          for(;count<MAXDATA;)
          {
                  sleep(1); ",count);
                  printf("%d
          }
          pci3133_set_gate(fd,PCI3346A_GATE_OFF);
        pci3346a_disable_intrrupt(fd ,PCI3346A_IMASK_ALL);

        printf("Interrupt test results.0);
                printf("Error %d0,error);
                printf("Success %d0,count-error);

        value = PCI3346A_CV_MODE_ALLCLEAR;
        ret = pci3346a_set_ioport(fd,PCI3346A_CV_MODE_OFFSET,&value);
        if(ret) fprintf(stderr,"pci3346a_set_ioport() error0);

        if (pci3346a_uninit(fd)<0)
        {
                 fprintf(stderr,"%s0,strerror(errno));
        }
        close(fd);
    }
```

SEE ALSO

/usr/local/CNC/drivers/extmem/interface/pci3346a/下のプログラム

AUTHORS