PIO-1616 Board Support Package Installation on RedHawk

Release Notes Revision B

September 12, 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、コンテック社製 PIO-1616 PCI Express 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

PIO-1616 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。

- PIO-1616 ボード
- RedHawk 6.x 以上
- Extmem version 8.3 以上

PIO-1616 TTLレベルデジタル信号の入出力を行う、PCIバス準拠のインターフェイスボードです。 このボードは、1枚で最大16点の入力と最大16点の出力ができます。

3. インストール方法

PIO-1616 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。

PIO-1616 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。:

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
```

PIO1616 BSP 実行パッケージのインストール

```
# apt install ./bin-pio1616 -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-pio1616 -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
```

PIO1616 BSP 実行パッケージのインストール

```
# apt install ./bin-pio1616 -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-pio1616 -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。)

PIO-1616 BSP パッケージは ***/usr/local/CNC/drivers/extmem/interface/pio1616*** 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

PIO-1616 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/pio1616 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-pio1616 -**X.Y**_RH***x.y***-***z***.x86_64 *(開発パッケージの削除)*
rpm -e bin-pio1616 -**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-pio1616 -**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-pio1616 -rhx.y *(開発パッケージの削除)*
apt purge bin-pio1616 -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-pio1616 -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-pio1616 -rhx.y *(開発パッケージの削除)*
apt purge bin-pio1616 -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-pio1616 -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 pio1616
pio1616(3) pio1616(3)

NAME
pio1616 - 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
pio1616 は、external memory ドライバを利用した pio1616 ボードアクセスライブラリです。

#include <sys/pio1616.h>
gcc [options ...] file -lpio1616 -lextmem ...

****
PIO1616
****

割り込みハンドラの登録
int pio1616_setup_signal
(
int fd,
void (*interrupt_hadler)(int, siginfo_t *, void *),
unsigned long 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マスク値

デバイスの非初期化処理
int pio1616_reset(int fd);
int pio1616_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 pio1616_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

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 0x10-0x13)の値を戻す
int pio1616_intr_service
(
int fd,
unsigned long intt *iflag,
unsigned long 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 pio1616_disable_intrrupt
(
int fd,
unsigned long 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

割り込みを許可する
int pio1616_enable_intrrupt
(
int fd,
unsigned long 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 割り込みを許可するビットマスク

インターバルタイマーをセットする
int pio1616_set_system_timer(int fd,unsigned long 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 値を設定する変数へのポインタ

インターバルタイマーの現在値を読み出す
int pio1616_get_system_timer(int fd,unsigned long 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 pio1616_get_ioport(int fd,int offset,unsigned long 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 pio1616_set_ioport(int fd,int offset,unsigned long 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
int pio1616_read_data(int fd,int ch,unsigned char *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

チャネルを指定してデータを出力する
int pio1616_write_data(int fd,int ch,unsigned char *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

すべてのチャネルの入力データを読み出す
int pio1616_read_data_all(int fd,unsigned char *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 入力配列変数へのポインタ
LP すべてのチャネルのデータを出力する
int pio1616_write_data_all(int fd,unsigned char *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 出力配列変数へのポインタ

SEE ALSO

/usr/local/CNC/drivers/extmem/contec/pio1616 下のプログラム

AUTHORS

Copyright (C) 1995-2016 Concurrent Real Time Inc.