PEX-361116 Board Support Package Installation on RedHawk

Release Notes Revision C

March 11,2025

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 PEX-361116 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

PEX- 361116 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。

- PEX- 361116 ボード
- RedHawk 6.x 以上
- Extmem version 10.1A 以上

PEX-361116/PEX-361116Nは、AD16ビットD8/S16CH /DA16ビット2CH /DIOカウンタ複合を持つ、PCI Expressマルチファンクション製品です。

3. インストール方法

PEX-361116 BSP は、MSI あるいは 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。MSI は、PEX-361116N でサポートされますが、両ボードの混在の場合には MSI は利用できません。
“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。
PEX-361116 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
PEX361116 BSP 実行パッケージのインストール
rpm -ivh bin-pex361116 -**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-pex361116 -**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

PEX361116 BSP 実行パッケージのインストール
apt install ./bin-pex361116 -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-pex361116 -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

PEX361116 BSP 実行パッケージのインストール
apt install ./bin-pex361116 -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-pex361116 -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。)

PEX-361116 BSP パッケージは ***/usr/local/CNC/drivers/extmem/interface/pex361116*** 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

PEX-361116 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/pex361116*** 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-pex361116 -**X.Y**_RH***x.y-z***.x86_64 *(開発パッケージの削除)*
rpm -e bin-pex361116 -**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-pex361116 -**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-pex361116 -rhx.y *(開発パッケージの削除)*
apt purge bin-pex361116 -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-pex361116 -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-pex361116 -rhx.y *(開発パッケージの削除)*
apt purge bin-pex361116 -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-pex361116 -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。
pex361116(3)　　Library Functions Manual　　pex361116(3)

NAME
　pex361116 - external memory board support library

SYNOPSIS
　[ボードの詳細は、各マニュアルを見てください]

DESCRIPTION
　pex361116 は、external memory ドライバを利用した pex361116 ボードアクセスライブラリです。

　#include <sys/pex361116.h>
　gcc [options ...] file -lpex361116 -lextmem ...

　***
　PEX361116
　***

　割り込みハンドラの登録
　int pex361116_setup_signal (int fd, void (*interrupt_handler)(int, siginfo_t *, void *));
　　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_handler)(int, siginfo_t *, void *) 割り込みハンドラ

　デバイスの非初期化処理
　int pex361116_uninit(int fd,PEX361116R *adc,PEX361116R *dac,PEX361116R *clk);
　　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
　　　adc AD 制御用メモリ領域へのポインタ。
　　　dac DA 制御用メモリ領域へのポインタ。
　　　clk カウンタ制御用メモリ領域へのポインタ。

　デバイスの初期化処理
　int pex361116_init(int fd,PEX361116R **adc,PEX361116R **dac,PEX361116R **clk, 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
　　　adc AD 制御用メモリ領域へのポインタ。128 バイト分占有。
　　　dac DA 制御用メモリ領域へのポインタ。32 バイト分占有。
　　　clk カウンタ制御用メモリ領域へのポインタ。32 バイト分占有。
　　　option 1を指定すると以下の情報が表示される
　　　ADC mapped 4096 bytes,at 0xf5ffd000 DAC mapped 4096 bytes,at 0xf5ffe000 CLK mapped 4096 bytes,at 0xf5fff000
　デバイスのリセット処理(pex361116_init(),pex361116_uninit()から呼び出される)
　void pex361116_reset(PEX361116R *adc,PEX361116R *dac,PEX361116R *clk);

戻り値
なし
引数
adc AD 制御用メモリ領域へのポインタ。
dac DA 制御用メモリ領域へのポインタ。
clk カウンタ制御用メモリ領域へのポインタ。
注意
デバイスのリセット処理、初期化、非初期化を行うと、自動的に割り込みは禁止になり、DIO モードに設定される。

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 pex361116_intr_service(int fd,unsigned long int *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 割り込み制御レジスタ値を戻す変数
割り込んだ際の以下の割り込み要因レジスタの値
((PEX361116_ADC_IFR_OFFSET レジスタ) & 0xFFFF)|
((PEX361116_DAC_IFR_OFFSET レジスタ) & 0xFF << 16)|
((PEX361116_CLK_IFR_OFFSET レジスタ) & 0xFF << 24);
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

void pex361116_disable_adc_intrrupt(PEX361116R *adc);
void pex361116_disable_dac_intrrupt(PEX361116R *dac);
void pex361116_disable_clk_intrrupt(PEX361116R *clk);
戻り値
なし
引数
adc AD 制御用メモリ領域へのポインタ。
dac DA 制御用メモリ領域へのポインタ。
clk カウンタ制御用メモリ領域へのポインタ。

割り込みを許可する
void pex361116_enable_adc_intrrupt(PEX361116R *adc,int mask);
戻り値
なし
引数
adc AD 制御用メモリ領域へのポインタ。
mask 割り込みを許可するビットマスク。以下の値の論理和を指定する
PEX361116_ADC_ADDMA_DTE AD DMA データ転送
PEX361116_ADC_PRDMA_STE プリトリガ DMA スキャッタ
PEX361116_ADC_ADDMA_STE AD DMA スキャッタ
PEX361116_ADC_ADDMA_ORE AD DMA オーバランエラー
PEX361116_ADC_ADDMA_END AD DMA 転送完了
PEX361116_ADC_SCER サンプリングクロックエラー
PEX361116_ADC_EXTRG_IN 外部トリガ入力
PEX361116_ADC_ATRG アナログトリガ
PEX361116_ADC_EXINT_IN 外部割り込み入力
PEX361116_ADC_FF フルスケール検出
PEX361116_ADC_ADSMP AD サンプリング終了
PEX361116_ADC_ADSPS AD サンプリング開始
PEX361116_ADC_ADTMR サンプリングクロック

PEX361116_ADC_ICR_ALL　上記の全て

void pex361116_enable_dac_intrrupt(PEX361116R *dac,int mask);

戻り値

なし

引数

dac DA 制御用メモリ領域へのポインタ。

mask 割り込みを許可するビットマスク 以下の値の論理和を指定する

PEX361116_DAC_EXTRG_IN 外部トリガ割り込み入力

PEX361116_DAC_ATRG　アナログトリガ割り込み

PEX361116_DAC_EXINT_IN 外部割り込み入力

PEX361116_DAC_DASPS　DA 変換開始割り込み

PEX361116_DAC_DATMR　DA 出力更新クロック割り込み

PEX361116_DAC_ICR_ALL　上記の全て

void pex361116_enable_clk_intrrupt(PEX361116R *clk,int mask);

戻り値

なし

引数

clk カウンタ制御用メモリ領域へのポインタ。

mask 割り込みを許可するビットマスク 以下の値の論理和を指定する

PEX361116_CLK_PERR　異常入力検出での割り込み

PEX361116_CLK_C_B　キャリー/ボローでの割り込み

PEX361116_CLK_EXLT　外部ラッチでの割り込み

PEX361116_CLK_EQ　カウンタと比較カウンター致での割り込み

PEX361116_CLK_ICR_ALL　上記の全て

汎用関数 オフセット値を指定して 8 ビットレジスタの値を読み出す

int pex361116_get_byte_mmap(PEX361116R *dev ,int offset,unsigned long int *value);

戻り値

エラーなら-1 成功なら 0

引数

dev 制御用メモリ領域へのポインタ。

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 16 ビットレジスタの値を読み出す

int pex361116_get_word_mmap(PEX361116R *dev ,int offset,unsigned long int *value);

戻り値

エラーなら-1 成功なら 0

引数

dev 制御用メモリ領域へのポインタ。

offset レジスタオフセット

オフセットはバイトの位置で指定する。

また、0x80 以上の値はエラーになる。

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 32 ビットレジスタの値を読み出す

int pex361116_get_long_mmap(PEX361116R *dev ,int offset,unsigned long int *value);

戻り値

エラーなら-1 成功なら 0

引数

dev 制御用メモリ領域へのポインタ。

offset レジスタオフセット

オフセットはバイトの位置で指定する。

また、0x80 以上の値はエラーになる。

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 8 ビットレジスタに値を書き出す
int pex361116_set_byte_mmap(PEX361116R *dev ,int offset,unsigned long int *value);
戻り値
エラーなら-1 成功なら 0
引数
dev 制御用メモリ領域へのポインタ。
offset レジスタオフセット
オフセットはバイトの位置で指定する。
また、DMA レジスタの存在する 0x20 から 0x37 と 0x80 以上の値はエラーになる。
value 値を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 16 ビットレジスタに値を書き出す
int pex361116_set_word_mmap(PEX361116R *dev ,int offset,unsigned long int *value);
戻り値
エラーなら-1 成功なら 0
引数
dev 制御用メモリ領域へのポインタ。
offset レジスタオフセット
オフセットはバイトの位置で指定する。
また、DMA レジスタの存在する 0x20 から 0x37 と 0x80 以上の値はエラーになる。
value 値を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 32 ビットレジスタに値を書き出す
int pex361116_set_long_mmap(PEX361116R *dev ,int offset,unsigned long int *value);
戻り値
エラーなら-1 成功なら 0
引数
dev 制御用メモリ領域へのポインタ。
offset レジスタオフセット
オフセットはバイトの位置で指定する。
また、DMA レジスタの存在する 0x20 から 0x37 と 0x80 以上の値はエラーになる。
value 値を出す変数へのポインタ

DIO を読み出す
int pex361116_read_dio(PEX361116R *adc,unsigned char *data)
戻り値
エラーなら-1 成功なら 0
引数
adc AD 制御用メモリ領域へのポインタ。
data 8bit 変数へのポインタ
DIO に出力する
int pex361116_write_dio(PEX361116R *adc,unsigned char *data);
戻り値
エラーなら-1 成功なら 0
引数
adc AD 制御用メモリ領域へのポインタ。
data 8bit 変数へのポインタ
注意
デジタル入出力は、オフセット PEX361116_CDIO_SELECT_DIO のデジタル入出力機能が有効(値が 0)でないと機能しない
また、値を書き出して、直後に読み出した場合、値が正しく反映されないため、待ちが必要である。
使用例のコードを参照

DAC クロックの設定
int pex361116_dac_clk_setup(PEX361116R *dac,unsigned int clk_type, unsigned int clk_count)

戻り値
エラーなら-1 成功なら 0
引数
clk_type
PEX361116_DAC_CLOCK_80000_CK または PEX361116_DAC_CLOCK_24576_CK
clk_count
PEX361116_DAC_CLOCK_80000_CK の場合 16 ビットカウンタで基準クロックは 8MHz です。
0000h~0027h までは設定禁止です。(0x0028=40,8000000/40=200KHz)

DAC データの同時出力
int pex361116_dac_update(PEX361116R *dac)
戻り値
エラーなら-1 成功なら 0
引数
dac DA 制御用メモリ領域へのポインタ。

DAC データの同時出力データの設定
int pex361116_dac_write(PEX361116R *dac,unsigned short int *array,int number)
戻り値
エラーなら-1 成功なら 0
引数
dac DA 制御用メモリ領域へのポインタ。
array DA 出力データの配列サイズは 2
number DA 出力データのサイズ 2 でなくてはならない

adc AD 制御用メモリ領域へのポインタ。

AD 変換クロックセットアップ
int pex361116_adc_clk_setup(PEX361116R *adc,unsigned int clk_type, unsigned int clk_count,unsigned int scan_count)
戻り値
エラーなら-1 成功なら 0
引数
adc
AD 制御用メモリ領域へのポインタ。
clk_type
PEX361116_ADC_CLOCK_80000_CK または PEX361116_ADC_CLOCK_24576_CK
clk_count
PEX361116_ADC_CLOCK_80000_CK の場合 16 ビットカウンタで基準クロックは 8MHz。
0000h~0027h までは設定禁止。(0x0028=40,8000000/40=200KHz)

int pex361116_adc_channel_settings(PEX361116R *adc,unsigned char *setup,unsigned int channels)
AD 変換チャンネル設定
本設定は、サンプリングクロックを内部クロックに設定した状態(pex361116_adc_clk_setup()後)で行う事。
戻り値
エラーなら-1 成功なら 0
引数
adc AD 制御用メモリ領域へのポインタ。
*setup
下記に示すサンプリング順序|物理チャンネルの channels サイズの配列
サンプリング順序|物理チャンネル
7～4 | 3～0

channels
setup のチャンネル数

設定例

CH16→ CH12→ CH8→ CH4→ CH1→ CH5→ CH9→ CH13:合計チャンネル数 8 のオートチャンネル切り替えを行う場合。

```
#define ADC_CHANNELS 8
unsigned char adc_setup[ADC_CHANNELS] =
{
	0x0F, //1st チャンネルを CH16 に設定
	0x1B, //2nd チャンネルを CH12 に設定
	0x27, //3rd チャンネルを CH8 に設定
	0x33, //4th チャンネルを CH4 に設定
	0x40, //5th チャンネルを CH1 に設定
	0x54, //6th チャンネルを CH5 に設定
	0x68, //7th チャンネルを CH9 に設定
	0x7C  //8th チャンネルを CH13 に設定
};
:
pex361116_adc_channel_settings(adc,adc_setup,ADC_CHANNELS);
:
```

32 ビット領域の DMA バッファのマップ,アンマップ。
コンフィグ時に EXTMEM_DMA0=65536 で確保しなければならない。

```
int pex361116_mmap_prealloc_buffer(int fd,unsigned char **virtual_addr)
int pex361116_unmmap_prealloc_buffer(int fd,unsigned char *virtual_addr)
```

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
virtual_addr DMA 用メモリ領域へのポインタ(仮想アドレス)。

AD 変換 DMA セットアップ

```
int pex361116_adc_dma_setup(int fd,PEX361116R *adc,int channels,int iteration)
```

コンフィグ時に EXTMEM_DMAn=65536 で確保した 32bit address 領域の DMA セットアップを行う。
EXTMEM_DMAn => ((channels * sizeof(short)) * iteration)でなくてはならない。
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
adc AD 制御用メモリ領域へのポインタ。
channels AD 変換のチャンネル数
iteration DMA の回数

AD 変換 DMA 開始

```
int pex361116_adc_dma_start(int fd,PEX361116R *adc,int channels,int iteration)
```

コンフィグ時に EXTMEM_DMAn=65536 で確保した 32bit address 領域へ、DMA を行う。
EXTMEM_DMAn => ((channels * sizeof(short)) * iteration)でなくてはならない。
実行前に pex361116_adc_dma_setup()を最低 1 回呼び出す必要がある。
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
adc AD 制御用メモリ領域へのポインタ。
channels AD 変換のチャンネル数
iteration DMA の回数

サンプルプログラム

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <memory.h>
#include <signal.h>
#include <curses.h>
#include <stdlib.h>
#include <time.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <sys/file.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/extmem.h>
#include <sys/pex361116.h>
#ifdef USESELECT
#include <sys/select.h>
#else
#if REDHAWK_VERSION >= 80
#include <poll.h>
extern __sighandler_t sigset (int __sig, __sighandler_t __disp) __THROW;
#else
#include <asm/poll.h>
#endif
#endif
#define ADC_CHANNELS (16)
#define REALTIME_SIGNAL
static struct timespec now,old;
static float itime;

static int RunRun=1;
static int shared=0;
static unsigned long int adc_iflag=0;
static unsigned long int dac_iflag=0;
static unsigned long int clk_iflag=0;
static unsigned long int pex361116_pending=0;
static int cycle=10;
static unsigned long int interrupts=0LL;
static unsigned long int update=0LL,old_update;
static float sampling_sec;
static float scan_sec;
static int  Iteration=1;

void ingetstr( int  y , int   x , char  *buff );
int inpdt();

/*****************************************************************************
*                       << sig >>
*     Function :
*****************************************************************************/
static void
sigint(int no)
{
        RunRun=0;
}

static void
```

```
sigbus(int no)
{
        struct timespec  rqtp={0,100000000},rmtp={0,0};

        nanosleep(&rqtp, &rmtp);
}

static void pex361116_interrupt_handler() ;
static void do_main();

static PEX361116R  *pex361116_adc;
static PEX361116R  *pex361116_dac;
static PEX361116R  *pex361116_clk;
static int pex361116_fd;
static unsigned short int     DAC_DATA[2]={0x8000,0x8000};
static PEX361116R *pex361116_dma;

#define readtime(nowtime)       clock_gettime(CLOCK_MONOTONIC,nowtime)
static void adjust(struct timespec *start,struct timespec *finish, float *realtime)
{
          register int  sec,nsec;

          sec = finish->tv_sec - start->tv_sec;
          nsec = finish->tv_nsec - start->tv_nsec;
          if (nsec<0)
          {
                    sec --;
                    nsec += 1000000000;
          }
          *realtime = (float)(nsec/1000)+(float)(sec*1000000);
}

/*************************************************************************
*                    <<< MAIN >>>
*      Function          : This Test program's     main Routine
*************************************************************************/
int main( int argc, char **argv)
{
          extern int optind,errno;
          extern char *optarg;
          int     c,sigio;
        int current=0;
        unsigned int itimer;
        unsigned char *virtual_addr;
        int scan_count;
        int clk_count;
        unsigned char val8;
        unsigned short int val16;
        unsigned int val32;
        static unsigned char adc_setup[ADC_CHANNELS] =
        {
         0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee,
0xff
        };
        char devname[1024];

        sigset( SIGINT, sigint ) ;
          while((c = getopt(argc, argv, "sc:i:")) != EOF){
                    switch(c) {
                              case 'i':
```

```
                        Iteration = atoi( optarg ) ;
                        if (( Iteration < 0 )||((Iteration)>16))
                        {
                                fprintf(stderr,"INVALID ARGUMENT!!0) ;
                                exit( 0 ) ;
                        }
                    break;

                    case 'c':
                        cycle = atoi( optarg ) ;
                        if (( cycle <= 0 )||((cycle)>1000))
                        {
                                fprintf(stderr,"INVALID ARGUMENT!!0) ;
                                exit( 0 ) ;
                        }
                    break;

                    case 's':
                  shared=1;
                    break;
                    default:
                        fprintf(stderr, "Usage: %s [options]0,argv[0]);
                        fprintf(stderr, "Options:0);
                        fprintf(stderr,"        -h          : This massage0);
                        fprintf(stderr,"        -s          : shared mode0);
                        fprintf(stderr,"        -c cycle speed0);
                        fprintf(stderr,"        -i adc iteratioan(0:dac timer ,1-1000 adc)0);
                        exit(-1);
            }
    }
    if (optind == argc) {
            /* no more option */
        strcpy(devname,"/dev/pex361116/0");
    } else {
        strcpy(devname,argv[optind]);
    }
    if ((pex361116_fd = open(devname,O_RDWR)) == -1)
{
        fprintf(stderr,"Device not found %s(%s)0,
        devname,strerror(errno));
        exit(0);
}
if (pex361116_init(pex361116_fd,&pex361116_adc,&pex361116_dac,&pex361116_clk,0)<0)
{
        fprintf(stderr,"Device initialize error %s(%s)0,
        devname,strerror(errno));
        exit(0);
}

if (pex361116_mmap_prealloc_buffer(pex361116_fd,&virtual_addr))
{
        fprintf(stderr,"pex361116_get_prealloc_buffer() error %s(%s)0,
        devname,strerror(errno));
        exit(0);
}

pex361116_dma = (PEX361116R *)virtual_addr;

if (shared)
{
```

```
                    if(ioctl(pex361116_fd,IOCTL_EXTMEM_SHARED,NULL)<0)
                    {
                        fprintf(stderr,"can not set shared mode %s(%s)0,
                        devname,strerror(errno));
                    }
                }
                  if(ioctl(pex361116_fd,IOCTL_EXTMEM_GET_SHARE_USERS,&current)<0)
                  {
                          fprintf(stderr,"extmem IOCTL_EXTMEM_GET_SHARE_USERS
fail %s0,strerror(errno));
                          return(-1);
                  }
                  if ( current==1)
                {
                        unsigned char value8 = PEX361116_DAC_GATE_CONTROL_ON;
                        int set =1;
                        sigio = SIGRTMIN;

                        if (ioctl(pex361116_fd,IOCTL_EXTMEM_SET_SIGNAL_NUMBER,&sigio)<0)
                        {
                                fprintf(stderr,"Change signal number error %s(%s)0,devname,strerror(errno));
                            exit(0);
                        }
      #ifdef REALTIME_SIGNAL
                        if (ioctl(pex361116_fd,IOCTL_EXTMEM_SET_SIGNAL_INFO,&set)<0)
                        {
                                fprintf(stderr,"IOCTL_EXTMEM_SET_SIGNAL_INFO
error %s(%s)0,devname,strerror(errno));
                            exit(0);
                        }
      #endif
                        if (pex361116_setup_signal(pex361116_fd,pex361116_interrupt_handler)<0)
                        {
                            fprintf(stderr,"Interrupt initialize error %s(%s)0,
                            devname,strerror(errno));
                            exit(0);
                        }

                        if (Iteration)
                        {
                            /* 入力仕様設定 (シングルエンド入力,サンプリングステータスなし) */
                            val8 = PEX361116_ADC_INP_ADDST0|PEX361116_ADC_INP_SD0; /* サンプリング
ステータスなし|シングルエンド入力*/

pex361116_set_byte_mmap(pex361116_adc,PEX361116_ADC_INP_SETTINGS_OFFSET,&val8);
                            {
                            /* ※ 入力仕様設定レジスタ設定後、10ms の Wait が必要となります。*/
                                    struct timespec  w10m={0,10000000},rmtp={0,0};
                                    nanosleep(&w10m, NULL);
                            }
                            /* 変換の scan 周期を cycle に設定*/
                            //scan_count = 8; /*MAX 1000.0KHz 1us   1ch*/
                            scan_count = 128; /*MAX     62.5KHz 16us 16ch*/

                            clk_count = (unsigned int) (8000000 / cycle);
                            sampling_sec   = 1.0/8.0 * (float)clk_count;
                            scan_sec     = 1.0/8.0 * (float)(scan_count);
                            /*
```

```
                        サンプリングクロック周期(clk_count) >= スキャンクロック周期(scan_count) x サンプリングチャンネル数
                    */
                    //printf("サンプリングクロック周期(%f us) >= スキャンクロック周期(%f us)0,sampling_sec, scan_sec);

                    if (pex361116_adc_clk_setup(pex361116_adc,PEX361116_ADC_CLOCK_80000_CK, clk_count,scan_count)<0)
                    {
                        fprintf(stderr,"pex361116_adc_clk_setup() error %s(%s)0,
                        devname,strerror(errno));
                        exit(0);
                    }
                    /* 入力チャンネル設定 (サンプリングチャンネル,サンプリング順序, サンプリングチャンネル数) */
                    pex361116_adc_channel_settings(pex361116_adc,adc_setup,ADC_CHANNELS);

                    if (pex361116_adc_dma_setup(pex361116_fd,pex361116_adc,ADC_CHANNELS,Iteration)==0)
                    {

pex361116_enable_adc_intrrupt(pex361116_adc,PEX361116_ADC_ADSMP|PEX361116_ADC_SCER);

pex361116_adc_dma_start(pex361116_fd,pex361116_adc,ADC_CHANNELS,Iteration);
                        /* AD 変換コマンド設定 (AD 変換スタート) */
                        val8 = PEX361116_ADC_SCMD_START;

pex361116_set_byte_mmap(pex361116_adc,PEX361116_ADC_SCMD_OFFSET,&val8);
                    }
                }
                else
                {
                    /* DAC タイマの周期を 100ms に設定*/
                    itimer = (unsigned int) (8000000 / cycle); /* 100ms */
                    if (pex361116_dac_clk_setup(pex361116_dac,PEX361116_DAC_CLOCK_80000_CK,itimer)<0)
                    {
                        fprintf(stderr,"pex361116_dac_clk_setup() error %s(%s)0,
                        devname,strerror(errno));
                        exit(0);
                    }
                    pex361116_enable_dac_intrrupt(pex361116_dac,PEX361116_DAC_DATMR);
                }
                pex361116_enable_clk_intrrupt(pex361116_clk,PEX361116_CLK_ICR_ALL);
                {/* デジタル入出力選択 */
                    unsigned char value=PEX361116_CDIO_SELECT_DIO;


pex361116_set_byte_mmap(pex361116_clk,PEX361116_CDIO_SELECT_OFFSET,&value);
                }
                val8=0;

pex361116_set_byte_mmap(pex361116_dac,PEX361116_DAC_DATA_CH_OFFSET,&val8);

pex361116_get_word_mmap(pex361116_dac,PEX361116_DAC_DATA_OFFSET,&DAC_DATA[0]);
                val8=1;

pex361116_set_byte_mmap(pex361116_dac,PEX361116_DAC_DATA_CH_OFFSET,&val8);
```

```
pex361116_get_word_mmap(pex361116_dac,PEX361116_DAC_DATA_OFFSET,&DAC_DATA[1]);
        }
        readtime(&old);
        do_main();
        if (current==1)
        {
                pex361116_disable_clk_intrrupt(pex361116_clk);
                pex361116_disable_dac_intrrupt(pex361116_dac);
                pex361116_disable_adc_intrrupt(pex361116_adc);
        }
        if (shared)
        {
                if(ioctl(pex361116_fd,IOCTL_EXTMEM_PRIVATE,NULL)<0)
                {
                    fprintf(stderr,"can not set private mode %s(%s)0,
                    devname,strerror(errno));
                }
        }

        pex361116_unmmap_prealloc_buffer(pex361116_fd,virtual_addr);
        pex361116_uninit(pex361116_fd,pex361116_adc,pex361116_dac,pex361116_clk);
        close(pex361116_fd);
}

static void
make_disp(void)
{
        unsigned char value;

        pex361116_get_byte_mmap(pex361116_clk,PEX361116_DIPSW_OFFSET,&value);
        initscr() ;
        cbreak();
        noecho();
        nonl();

        timeout(1);

        /*-------------------- display of initialize screen ----------------*/
          move( 0, 0 ) ; addstr("+--------------------------------------------------------------------------------+" ) ;
          move( 1, 0 ) ; addstr("| PEX36116 Register Monitor Program Version 2.0
         |" ) ;
          move( 2, 0 ) ; addstr("| i:Input register address for output(Note:DIO address is 0x48,DAC
0x00,0x01)  |" ) ;
          move( 3, 0 ) ; addstr("|
         |" ) ;
          move( 4, 0 ) ; addstr("| q:Quit
         |" ) ;
          move( 5, 0 ) ; addstr("+--------------------------------------------------------------------------------+" ) ;
        move( 3, 2 ) ;printw("DIPSW 0x%X ADC sampling_sec:%10.0f us scan_sec:%10.0f us",value
& 0xF,sampling_sec, scan_sec);
        refresh() ;
}
static char *bits(int val)
{
          static char bits_string[16];
          int i;

          for(i=0;i<8;i++)
          {
```

```
                if (val & (0x80>>i)) bits_string[i]='1'; else  bits_string[i]='0';
            }
            bits_string[8]=0;
            return(&bits_string[0]);
    }

    static float volts(unsigned short int val)
    {
            float volts;
            volts = ((float)(val-0x8000) * 20.0)/ 65536.0 ;
            return(volts);
    }

    static void do_disp(void)
    {
        unsigned char value;
        int ch;

        pex361116_get_byte_mmap(pex361116_adc,PEX361116_CDIO_MODE_OFFSET,&value); /*
デジタル入出力状態確認 */
        move( 4, 9 ) ;
        if ((int)(value &
PEX361116_CDIO_MODE_COUNTER)==PEX361116_CDIO_MODE_COUNTER)
        {
            printw("CLK MODE");
        }
        else
        {
            unsigned char data;
            printw("DIO MODE");
            pex361116_read_dio(pex361116_adc,&data);
            move( 4, 18) ; printw("0x%02x %sB",data,bits(data));
        }
        move(          4,        33)          ;          printw("DAC0          0x%04X(%+9.5f)
       DAC1
    0x%04X(%+9.5f)",DAC_DATA[0],volts(DAC_DATA[0]),DAC_DATA[1],volts(DAC_DATA[1]));
        if (update!=old_update)
        {
            old_update = update;
           for ( ch=0;ch<16;ch++)
           {
            switch(Iteration)
            {
                case 16: move(21, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*15]);
                case 15: move(20, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*14]);
                case 14: move(19, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*13]);
                case 13: move(18, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*12]);
                case 12: move(17, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*11]);
                case 11: move(16, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*10]);
                case 10: move(15, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*9]);
                case  9: move(14, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*8]);
                case  8: move(13, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*7]);
                case  7: move(12, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*6]);
                case  6: move(11, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*5]);
                case  5: move(10, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*4]);
                case  4: move( 9, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*3]);
                case  3: move( 8, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*2]);
                case  2: move( 7, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*1]);
                case  1: move( 6, ch*5 ) ; printw(" %04X",pex361116_dma->access.d16[ch+16*0]);
                default:
```

```
                    break;
                }
               }
        }
        move( 22, 1 ) ; printw("ADC IFLAG:0x%04X",adc_iflag) ;
        move( 22, 18) ; printw("DAC IFLAG:0x%02X",0xff & dac_iflag) ;
        move( 22, 33) ; printw("CLK IFLAG:0x%02X",0xff & clk_iflag) ;
        move( 22, 48) ; printw("INTERRUPT PENDING:%d",pex361116_pending) ;
        move( 23, 1 ) ; printw("Interval time %10.1f us % 10lld  ",itime,interrupts);
          refresh() ;
}

static void do_main()
{
        int     ret             ; /* Return Value of Called Function*/
        /*struct timespec  rqtp={0,100000000},rmtp={0,0};*/
                int  error,nfds=0;
        volatile int   n,key;
        struct timespec wait200={0,200000};
#ifdef USESELECT
        fd_set readfds,writefds,execptfds;
          struct timeval timeout={0,100000};
#else
        struct pollfd  fdarray[1];
#endif

        make_disp();
#ifdef USESELECT
        FD_ZERO(&readfds);
        FD_ZERO(&writefds);
        FD_ZERO(&execptfds);
#else
        nfds=1;
        fdarray[0].fd = STDIN_FILENO;
        fdarray[0].events = POLLIN|POLLRDNORM|POLLRDBAND|POLLPRI;
#endif
        while( RunRun )
        {
#ifdef USESELECT
                FD_SET(STDIN_FILENO,&readfds);nfds=1;
                if (select(nfds,&readfds,&writefds,&execptfds,&timeout)>0)
#else
                if (poll(fdarray,nfds,100)>0)
#endif
                {
                    key = getch() ;
                    if( key == 'q' ){
                            break ;
                    }
                    else if( key == 'i' ){
                            ret = inpdt() ;
                            move( 20, 80 ) ;
                            if( ret )
                                    break ;
                    }
                }
                do_disp();
        }
        move( 23, 0 ) ;
        echo() ;
```

```
        nl() ;
        endwin();
}

/*****************************************************************************
*               << INPDT >>
*   FUNCTION : Input Data to mapping aria
*****************************************************************************/
int inpdt()
{
        static char         buff[32] ;
        char ch   ;
        int      ret        ;
        int      xpos ;
        /*int     i ;*/
        static int          idx=0 ;
        int      inpflg ;
        int      incnt ;
        int      key ;
        /*char    buf[32] ;*/
        struct timespec  rqtp={0,100000000},rmtp={0,0};
                int nfds=0;
#ifdef USESELECT
        fd_set readfds,writefds,execptfds;
          struct timeval timeout={0,100000};
#else
        struct pollfd  fdarray[1];
#endif
        unsigned int data;
        unsigned char val8;

        key = 1 ;
        ret = 0 ;
        inpflg = 0 ;
        incnt = 0 ;
        xpos = 56 ;

        move( 23, 46 ) ; addstr("Hex INPUT:") ;
          move( 23, 56 ) ; addstr("      ") ;
        move( 23, 65 ) ; addstr("DATA:") ;
        move( 23, 70 ) ; addstr("              ") ;
        move( 23, 56 ) ;
        refresh() ;
#ifdef USESELECT
        FD_ZERO(&readfds);
        FD_ZERO(&writefds);
        FD_ZERO(&execptfds);
#else
        nfds=1;
        fdarray[0].fd = STDIN_FILENO;
        fdarray[0].events = POLLIN|POLLRDNORM|POLLRDBAND|POLLPRI;
#endif
        while( RunRun )
        {
                noecho() ;
                nonl() ;
#ifdef USESELECT
                FD_SET(STDIN_FILENO,&readfds);nfds=1;
                if(select(nfds,&readfds,&writefds,&execptfds,&timeout)<0)
#else
```

```
                if (poll(fdarray,nfds,100)<0)
    #endif
                {
                     nanosleep(&rqtp, &rmtp);
                }
                else {
                    key = getch() ;
                    if( key == 'i' ){
                            move( 23, 46 ) ; addstr("                ") ;
                            move( 23, 65 ) ; addstr("            ") ;
                            refresh() ;
                            break ;
                    }
                    else if ( ( '0' <= key && key <= '9' ) ||
                            ( 'a' <= key && key <= 'f' ) ){
                            ch = (char)key ;
                            if( !inpflg && incnt > 7 )
                                    incnt -- ;
                            else if( inpflg && incnt > 7 )
                                    incnt -- ;
                            move( 23, xpos+incnt ) ;
                            incnt++ ;
                            addch( ch ) ;
                    }
                    else if ( key == 13 ){              /*----- Input Index ------*/
                            if( inpflg == 0 ){
                                    ingetstr( 23, 56, buff ) ;
                                    sscanf( buff, "%x",(int*) &idx ) ;
                                    xpos = 70 ;
                                    move( 23, xpos ) ;
                                    inpflg = 1 ;
                                    if( (idx < 0) || ((0xff) < idx) ){
                                        xpos = 56 ;
                                        inpflg = 1 ;
                                        move( 23 , xpos ) ;
                                        addstr( "              " ) ;
                                        move( 23 , 56 ) ;
                                    }
                            }
                            else {                  /*----- Input Data ------*/
                                    xpos = 56 ;
                                    ingetstr( 23, 70, buff ) ;
                                        sscanf( buff, "%x",(int*) &data ) ;

                                    move( 23, 56 ) ; addstr(" ") ;
                                    move( 23, 70 ) ; addstr(" ") ;
                                    {
                                        static unsigned char cdata;
                                        cdata = data & 0xff;
                                        {
                                                switch(idx)
                                                {
                                                    case 0:
                                                            DAC_DATA[0] = data & 0xFFFF;
                                                            val8=0;

pex361116_set_byte_mmap(pex361116_dac,PEX361116_DAC_DATA_CH_OFFSET,&val8);

pex361116_set_word_mmap(pex361116_dac,PEX361116_DAC_DATA_OFFSET,&DAC_DATA[0]);
                                                            break;
```

```
                                        case 1:
                                                DAC_DATA[1] = data & 0xFFFF;
                                                val8=1;

pex361116_set_byte_mmap(pex361116_dac,PEX361116_DAC_DATA_CH_OFFSET,&val8);

pex361116_set_word_mmap(pex361116_dac,PEX361116_DAC_DATA_OFFSET,&DAC_DATA[1]);
                                                break;
                                                case PEX361116_DIO_OFFSET:
                                                        pex361116_write_dio(pex361116_adc,&cdata);
                                                break;
                                        }
                                    }
                                }
                                do_disp();

                                move( 23, xpos ) ;
                                inpflg = 0 ;
                        }
                        incnt = 0 ;
                }
                        else if(( key == 0x08 )||(key==0x7f)){
                                if (0<incnt)
                                {
                                        incnt -- ;
                                        move( 23, xpos+incnt ) ;
                                        addstr( " " ) ;
                                        move( 23, xpos+incnt ) ;
                                }

                            }
                else if( key == 'q' ){
                        ret = 1 ;
                        break ;
                }
                refresh() ;
            }
        }
        noecho() ;
        nonl() ;
        return( ret ) ;
}
void ingetstr( int  y , int   x , char  *buff )
{
        int     idx ;

        idx = 0 ;

        move( y, x ) ;
        while( ' ' != (buff[idx++] = inch()) ){
                x++ ;
                move( y, x ) ;
        }
}
/*****************************************************************************
*                   << intr0_handler >>
*         Function : Display when interrupt come!!
*
*****************************************************************************/
static
```

```
	void pex361116_interrupt_handler(int signo,siginfo_t *info, void *ptr)
	{
	#ifdef REALTIME_SIGNAL
		unsigned long int iflag = (unsigned long int)info->si_ptr;
		volatile unsigned char status;

		switch
	#if REDHAWK_VERSION >= 84
			( info->si_errno )
	#else
			( info->si_pid )
	#endif
		{
			case 0: /* ADC*/
			    adc_iflag = iflag & 0xFFFF;
				if (adc_iflag & PEX361116_ADC_SCER)
			    {
				fprintf(stderr,"サンプリングクロックエラー 0x%X0,adc_iflag);
				RunRun=0;
				return;
			    }
			    else
			    {
				update++;
				if (Iteration)
				{
					if (adc_iflag & PEX361116_ADC_ADSMP)
				    {
					readtime(&now);adjust(&old,&now,&itime); old.tv_sec=now.tv_sec;
old.tv_nsec=now.tv_nsec;

pex361116_adc_dma_start(pex361116_fd,pex361116_adc,ADC_CHANNELS,Iteration);
				    }
				}
			    }
			break;
			case 1: /* DAC*/
			    dac_iflag = iflag & 0xFF;
			    readtime(&now);adjust(&old,&now,&itime); old.tv_sec=now.tv_sec;
old.tv_nsec=now.tv_nsec;
			break;
			case 2: /* CLK*/
			    clk_iflag = iflag & 0xFF;
			break;
		}
	#else
		unsigned long int iflag;

		readtime(&now);adjust(&old,&now,&itime); old.tv_sec=now.tv_sec; old.tv_nsec=now.tv_nsec;
		pex361116_intr_service(pex361116_fd,&iflag,&pex361116_pending);
		adc_iflag = iflag & 0xFFFF;
		dac_iflag = (iflag>>16) & 0xFF;
		clk_iflag = (iflag>>24) & 0xFF;
	#endif
		interrupts++;
	}
```

SEE ALSO

/usr/local/CNC/drivers/extmem/interface/pex361116 下のプログラム

AUTHORS

6 Feb 2019

pex361116(3)