PEX-340416 Board Support Package Installation on RedHawk

Release Notes Revision A

January 31,2023

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-340416 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- 340416 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- 340416 ボード
- RedHawk 6.x 以上
- Extmem version 8.4E 以上

PEX-340416は、DA16ビット4CH /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。
しかし、本リリースでは、DAC/DIOとレジスタアクセスのみをサポートそれ以外のアクセスライブラリは提供されません。

3. インストール方法

PEX-340416 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。

PEX-340416 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
PEX340416 BSP 実行パッケージのインストール
rpm -ivh bin-pex340416 -**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-pex340416 -**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

PEX340416 BSP 実行パッケージのインストール
apt install ./bin-pex340416 -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-pex340416 -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

PEX340416 BSP 実行パッケージのインストール
apt install ./bin-pex340416 -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-pex340416 -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-340416 BSP パッケージは ***/usr/local/CNC/drivers/extmem/interface/pex340416*** 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-340416 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/pex340416*** 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-pex340416 -**X.Y**_RH***x.y-z***.x86_64 *(開発パッケージの削除)*
rpm -e bin-pex340416 -**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-pex340416 -**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-pex340416 -rhx.y *(開発パッケージの削除)*
apt purge bin-pex340416 -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-pex340416 -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-pex340416 -rhx.y *(開発パッケージの削除)*
apt purge bin-pex340416 -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-pex340416 -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 pex340416

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 ...

       ********************************************************************************
       PEX340416
       ********************************************************************************

       割り込みハンドラの登録
       int pex340416_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 pex340416_uninit(int fd,PEX340416R *dac,PEX340416R *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
                  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 pex340416_init(int fd,PEX340416R **dac,PEX340416R **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
                  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
                  DAC mapped 4096 bytes,at 0xb8601000 CLK mapped 4096 bytes,at 0xb8600000
       デバイスのリセット処理(pex340416_init(),pex340416_uninit()から呼び出される)
       void pex340416_reset(PEX340416R *dac,PEX340416R *clk);
            戻り値
                  なし
            引数
                  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。
                  DAC 出力リレーをすべて ON に設定

       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 pex340416_intr_service(int fd,unsigned long int *iflag,unsigned 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
                ((PEX340416_DAC_IFR_OFFSET レジスタ) & 0xFFFF)|
                ((PEX340416_CLK_IFR_OFFSET レジスタ) & 0xFF << 16);
         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

unsigned short int pex340416_disable_dac_intrrupt(PEX340416R *dac)
unsigned char pex340416_disable_clk_intrrupt(PEX340416R *clk)
      戻り値
            割り込みフラグレジスタの値
      引数
            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 pex340416_enable_dac_intrrupt(PEX340416R *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
                  PEX340416_DAC_DACE        DA 指定件数出力割り込み
                  PEX340416_DAC_FIFO_EMPTY FIFO エンプティ割り込み
                  PEX340416_DAC_FIFO_HALF   FIFO ハーフ割り込み
                  PEX340416_DAC_EXTRG_IN    外部トリガ割り込み入力
                  PEX340416_DAC_EXINT_IN    外部割り込み入力
                  PEX340416_DAC_DASPE       DA 変換開始割り込み
                  PEX340416_DAC_DASPS       DA 変換開始割り込み
                  PEX340416_DAC_DATMR       DA 出力更新クロック割り込み
                  PEX340416_DAC_ICR_ALL     上記の全て
void pex340416_enable_clk_intrrupt(PEX340416R *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
                  PEX340416_CLK_PERR        異常入力検出での割り込み
                  PEX340416_CLK_C_B キャリー/ボローでの割り込み
                  PEX340416_CLK_EXLT        外部ラッチでの割り込み
                  PEX340416_CLK_EQ          カウンタと比較カウンター致での割り込み
                  PEX340416_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 pex340416_get_byte_mmap(PEX340416R *dev ,int offset,unsigned char *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。(0x40 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 16 ビットレジスタの値を読み出す
int pex340416_get_word_mmap(PEX340416R *dev ,int offset,unsigned short 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。(0x40 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 pex340416_get_long_mmap(PEX340416R *dev ,int offset,unsigned 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。(0x40 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 pex340416_set_byte_mmap(PEX340416R *dev ,int offset,unsigned char *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。(0x40 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 pex340416_set_word_mmap(PEX340416R *dev ,int offset,unsigned short 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。
        オフセットはバイトの位置で指定する。(0x40 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 pex340416_set_long_mmap(PEX340416R *dev ,int offset,unsigned 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。(0x40 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 制御関数
DIO を読み出す
int pex340416_dio_read(PEX340416R *dac,unsigned char *data)
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。
data   8bit 変数へのポインタ
DIO に出力する
int pex340416_dio_write(PEX340416R *dac,unsigned char *data);
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。
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 PEX340416_CLK_SELECT_OFFSET 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

使用例
```
/*************************************************************************/
/*                                                                       */
/*  test.c --  PEX-340416                        */
/*                                                                       */
/*************************************************************************/
#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 <time.h>
#include <sys/pex340416.h>

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]);
}

int main(int argc, char **argv)
{
     int  fd;
     char devname[1024];
     PEX340416R *dac;
     PEX340416R *clk;
     int shared,i;
     unsigned char value;
     unsigned char data;
     struct timespec w1u={0,1000};

     if ( argc < 2 ) {
          strcpy(devname,"/dev/pex340416/0");
     } else {
          strcpy(devname,argv[1]);
     }
        if ((fd = open(devname,O_RDWR)) == -1)
     {
          fprintf(stderr,"Device not found %s(%s)0,
          devname,strerror(errno));
                    exit(0);
     }
     printf("%s ",devname);
     //share user count
     ioctl(fd,IOCTL_EXTMEM_GET_SHARE_USERS,&shared);
     printf("share user %d0,shared);
     if (pex340416_init(fd,&dac,&clk,1)<0)
     {
          fprintf(stderr,"Device initialize error %s(%s)0,
          devname,strerror(errno));
                    exit(0);
     }
     /* デジタル入出力状態確認 */
     pex340416_get_byte_mmap(clk,PEX340416_CLK_SELECT_OFFSET,&value);

     if ((int)(value & PEX340416_CLK_SELECT_COUNTER)==PEX340416_CLK_SELECT_COUNTER)
     {
          value=PEX340416_CLK_SELECT_DIO;
          pex340416_set_byte_mmap(clk,PEX340416_CLK_SELECT_OFFSET,&value);
     }

     data =0;
     pex340416_dio_write(dac,&data);
          nanosleep(&w1u,NULL);/*待ちがないと値が反映されない*/
     pex340416_dio_read(dac,&data);
     printf("0x%02x %sB0,data,bits(data));
     for(i=0;i<8;i++)
     {
          data = (0xFF & (1<<i));
          pex340416_dio_write(dac,&data);
               nanosleep(&w1u,NULL);/*待ちがないと値が反映されない*/
          pex340416_dio_read(dac,&data);
          printf("0x%02x %sB0,data,bits(data));
     }
     data =0;
```

```
	pex340416_dio_write(dac,&data);
	     nanosleep(&w1u,NULL);/*待ちがないと値が反映されない*/
	pex340416_dio_read(dac,&data);
	printf("0x%02x %sB0,data,bits(data));

	if (pex340416_uninit(fd,dac,clk)<0)
	{
		fprintf(stderr,"%s0,strerror(errno));
	}
	close(fd);
}
```

アナログ出力制御関数
int pex340416_dac_setup(PEX340416R *dac,unsigned int clk_type, unsigned int clk_count)
アナログ出力のクロック設定を行います
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。
clk_type PEX340416_DAC_MODE_OFFSET に設定する値
以下の値の論理和 を設定する
PEX340416_DAC_MODE_INTERNAL_CK 内部クロック
PEX340416_DAC_MODE_EXTERNAL_CK 外部クロック
PEX340416_DAC_MODE_COUNTER_CK カウンタアップダウンクロック
PEX340416_DAC_MODE_RETRIGER_DISABLE リトリガ無効カウンタアップダウンクロック
PEX340416_DAC_MODE_RETRIGER_ENABLE リトリガ有効
clk_count PEX340416_DAC_DATA_UPDATE_CLOCK_OFFSET に設定する値
通常 (PEX340416_DAC_CLOCK_8MHZ / 出力速度) を設定する
なお、PEX340416_DAC_CLOCK_CHANGE_OFFSET には、
PEX340416_DAC_CLOCK_UPDATECLOCK|PEX340416_DAC_CLOCK_CHANGE_TIMER_DISABLE が設定される。
int pex340416_dac_write_continuous_stop(PEX340416R *dac);
アナログ連続出力の強制停止
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。
int pex340416_dac_write_clock_trigger(PEX340416R *dac,unsigned char value);
イネーブルの場合、DA クロック出力は DA 出力更新中に DA 変換タイミングに合わせて 1μs の Low パルスが出ます。
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。
value PEX340416_DAC_CLOCK_TRIGGER_CTRL_OFFSET に設定する値
以下の値あるいは、0 を設定する
PEX340416_DAC_CLOCK_TRIGGER_CTRL_TRG_EN DA トリガ出力設定 有効
PEX340416_DAC_CLOCK_TRIGGER_CTRL_CLK_EN DA クロック出力設定 有効
int pex340416_dac_write_external_input_control(PEX340416R *dac,unsigned char value);
外部リセット入力制御レジスタを設定します。
外部リセット入力時は DA 出力は 0V となり、かつサンプリングが終了します。
外部リセットは立ち下がりエッジで反応します。
外部リセット入力には 51ms のデジタルフィルタがついています。
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。
value PEX340416_DAC_EXT_RESET_INPUT_CTRL_OFFSET に設定する値
以下の値あるいは、0 を設定する
PEX340416_DAC_EXT_RESET_INPUT_CTRL_FIL_EN フィルタ設定有効
PEX340416_DAC_EXT_RESET_INPUT_CTRL_RESET_EN 外部リセット入力有効
int pex340416_dac_write_fifo(PEX340416R *dac, unsigned short int *array,int number);
4 チャンネル分のアナログ出力データを連続して書き込みます。
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 FIFO へ出力するデータ
FIFO は幅 16 ビット× 4 チャンネル, 深さ 256 で、4 チャンネル分まとめて書き出します。

データの並びは、 ch0-0,ch1-0,ch2-0,ch3-0,ch0-1,ch1-1,ch2-1,ch3-1,..., ch0-N,ch1-N,ch2-N,ch3-N で

number データの数

最大は 256 です。

int pex340416_dac_write_array(PEX340416R *dac, unsigned short int *array,int number);

指定件数による連続出力、全チャンネルから指定件数分の連続出力を行い、出力を終了します。

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 FIFO へ出力するデータ

FIFO は幅 16 ビット× 4 チャンネル， 深さ 256 で、4 チャンネル分まとめて書き出します。

データの並びは、 ch0-0,ch1-0,ch2-0,ch3-0,ch0-1,ch1-1,ch2-1,ch3-1,..., ch0-N,ch1-N,ch2-N,ch3-N で

number データの数

最大は 256 です。

int pex340416_dac_write_continuous(PEX340416R *dac, unsigned short int *array,int number);

出力データ追加型連続出力 、FIFO 容量が 128 件以下になると発生する割り込みを使用することで、FIFO 容量(256 件)以上の連続出力を行います。

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 FIFO へ出力するデータ

FIFO は幅 16 ビット× 4 チャンネル， 深さ 256 で、4 チャンネル分まとめて書き出します。

データの並びは、 ch0-0,ch1-0,ch2-0,ch3-0,ch0-1,ch1-1,ch2-1,ch3-1,..., ch0-N,ch1-N,ch2-N,ch3-N で

number データの数

最大は 256 です。

注意

シグナルによる割り込みの通知が必須です。

また、割り込み時のフラグが PEX340416_DAC_FIFO_EMPTY の場合、データ更新が間に合っていません。

必ずリアルタイム優先度で動作させる必要があります。

pex340416_test.c のコードを参照

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 <time.h>
#include <math.h>   /* sor sin() */
#include <signal.h>
#include <sys/pex340416.h>

#define PI     3.14159265358979323846
#define DATASIZE PEX340416_DAC_FIFO_SIZE

static PEX340416R  *pex340416_dac;
static PEX340416R  *pex340416_clk;
static int pex340416_fd;
static unsigned int dac_iflag=0;
static unsigned int clk_iflag=0;
static unsigned int pex340416_pending=0;
static unsigned short int DAC[DATASIZE][PEX340416_DAC_CHANNELS];
static float Volts[DATASIZE][PEX340416_DAC_CHANNELS];
static int RunRun=1;

static void pex340416_interrupt_handler(int signo,siginfo_t *info, void *ptr)
{
        unsigned long int iflag;
```

```
     /* 割り込みが発生すると、割り込みフラグレジスタの各ビットが ON になります。
        このフラグで発生した割り込みの要因を確認します。*/
        pex340416_intr_service(pex340416_fd,&iflag,&pex340416_pending);
        dac_iflag = (iflag    ) & 0xFFFF;
        clk_iflag = (iflag>>16) & 0xFF;
}

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 int di_test(PEX340416R *dac, PEX340416R *clk)
{
     int i;
     unsigned char value8;
     unsigned char data;
     struct timespec w1u={0,1000};

     pex340416_get_byte_mmap(clk,PEX340416_CLK_SELECT_OFFSET,&value8);
     if ((int)(value8 & PEX340416_CLK_SELECT_COUNTER)==PEX340416_CLK_SELECT_COUNTER)
     {
          value8=PEX340416_CLK_SELECT_DIO;
          pex340416_set_byte_mmap(clk,PEX340416_CLK_SELECT_OFFSET,&value8);
     }
     pex340416_dio_read(dac,&data);
     printf("0x%02x %sB0,data,bits(data));
     for(i=0;i<8;i++)
     {
          data = (0xFF & (1<<i));
          pex340416_dio_read(dac,&data);
          printf("0x%02x %sB0,data,bits(data));
          sleep(1);
     }
     return(0);
}

static int do_test(PEX340416R *dac, PEX340416R *clk)
{
     int i;
     unsigned char value8;
     unsigned char data;
     struct timespec w1u={0,1000};

     pex340416_get_byte_mmap(clk,PEX340416_CLK_SELECT_OFFSET,&value8);
     if ((int)(value8 & PEX340416_CLK_SELECT_COUNTER)==PEX340416_CLK_SELECT_COUNTER)
     {
          value8=PEX340416_CLK_SELECT_DIO;
          pex340416_set_byte_mmap(clk,PEX340416_CLK_SELECT_OFFSET,&value8);
     }
     data =0;
     pex340416_dio_write(dac,&data);
          nanosleep(&w1u,NULL);/*待ちがないと値が反映されない*/
     pex340416_dio_read(dac,&data);
     printf("0x%02x %sB0,data,bits(data));
     for(i=0;i<8;i++)
     {
          data = (0xFF & (1<<i));
          pex340416_dio_write(dac,&data);
               nanosleep(&w1u,NULL);/*待ちがないと値が反映されない*/
          pex340416_dio_read(dac,&data);
          printf("0x%02x %sB0,data,bits(data));
          sleep(1);
     }
     data =0;
     pex340416_dio_write(dac,&data);
```

```
          nanosleep(&w1u,NULL);/*待ちがないと値が反映されない*/
     return(0);
}

static void sigint(int no)
{
        RunRun=0;
}

static int ao_test1(PEX340416R *dac,int signo,int speed,int ext_clkin)
{ /* 指定件数による連続出力 */
     int i;
     unsigned long int value;
     struct timespec w1m={0,1000000};
     unsigned int clk_type;
     unsigned int clk_count;
     sigset_t imask;

        signal( SIGINT, sigint ) ;
        sigfillset(&imask);
        sigdelset(&imask,SIGINT);
        sigdelset(&imask,signo);

     if (ext_clkin==0)
          clk_type = PEX340416_DAC_MODE_INTERNAL_CK | PEX340416_DAC_MODE_RETRIGER_DISABLE;
     else
          clk_type = PEX340416_DAC_MODE_EXTERNAL_CK | PEX340416_DAC_MODE_RETRIGER_DISABLE;

     clk_count = PEX340416_DAC_CLOCK_8MHZ / speed;
     pex340416_dac_setup(dac,clk_type,clk_count);
     /*
          全チャンネルから指定件数分の連続出力を行い、出力を終了します。
          4 チャンネル分の DA 変換データを連続して書き込みます。
          4 チャンネル設定されるとその時点で FIFO にライトされます。
          FIFO は幅 16 ビット× 4 チャンネル， 深さ 256 で、4 チャンネル分まとめて動作します。
     */
     pex340416_dac_write_array(dac,(unsigned short int*)DAC,PEX340416_DAC_FIFO_SIZE);
     while(RunRun)
     {
               sigsuspend(&imask);/* PEX340416_DAC_FIFO_HALF 割り込みシグナル(sigio)を待つ */
          if (dac_iflag & PEX340416_DAC_DACE)
          { /* DA 指定件数出力割り込み */
               fprintf(stderr,"PEX340416_DAC_DACE Exit(0x%X)0,dac_iflag);
               break;
          }
     }
     return(0);
}

static int ao_test2(PEX340416R *dac, int   signo,int speed,int ext_clkin)
{/* 出力データ追加型連続出力 */
     int i;
     float n;
     unsigned long int value;
     struct timespec w1m={0,1000000};
     sigset_t imask;
     unsigned int clk_type;
     unsigned int clk_count;

        signal( SIGINT, sigint ) ;
        sigfillset(&imask);
        sigdelset(&imask,SIGINT);
        sigdelset(&imask,signo);

     if (ext_clkin==0)
          clk_type = PEX340416_DAC_MODE_INTERNAL_CK | PEX340416_DAC_MODE_RETRIGER_DISABLE;
     else
          clk_type = PEX340416_DAC_MODE_EXTERNAL_CK | PEX340416_DAC_MODE_RETRIGER_DISABLE;
     clk_count = PEX340416_DAC_CLOCK_8MHZ / speed;
     pex340416_dac_setup(dac,clk_type,clk_count);
     /* FIFO にデータを書き出し、DA 変換を連続モードで開始する*/
     pex340416_dac_write_continuous(dac,(unsigned short int*)DAC,PEX340416_DAC_FIFO_SIZE);
     i=0;
```

```
     while(RunRun)
     {
               sigsuspend(&imask);/* PEX340416_DAC_FIFO_HALF 割り込みシグナル(sigio)を待つ */
          //fprintf(stderr,"(0x%X)0,dac_iflag);
/*
          PEX340416_DAC_FIFO_EMPTY FIFO エンプティ割り込み
          PEX340416_DAC_FIFO_HALF  FIFO ハーフ割り込み
          PEX340416_DAC_EXTRG_IN    外部トリガ割り込み入力
          PEX340416_DAC_EXINT_IN    外部割り込み入力
          PEX340416_DAC_DASPE       DA 変換開始割り込み
          PEX340416_DAC_DASPS       DA 変換開始割り込み
          PEX340416_DAC_DATMR       DA 出力更新クロック割り込み
*/
          if (dac_iflag & PEX340416_DAC_FIFO_EMPTY)
          { /* PEX340416_DAC_FIFO_EMPTY FIFO がエンプティになったらデータ更新が間に合っていない */
               fprintf(stderr,"Error FIFO is empty Exit(0x%X)0,dac_iflag);
               break;
          }
          if (dac_iflag & PEX340416_DAC_FIFO_HALF)
          {
               pex340416_dac_write_fifo(dac,(unsigned short int*)&DAC[i][0],DATASIZE/2);
               i += DATASIZE/2;
               i %= (DATASIZE);
          }
     }
     pex340416_dac_write_continuous_stop(dac);
     return(0);
}

int main(int argc, char **argv)
{
        extern int optind,errno;
        extern char *optarg;
     int  c,fd,i,j;
     unsigned int val;
     float n;
     char devname[1024];
     PEX340416R *dac;
     PEX340416R *clk;
     int shared;
     int mode=0;
     int speed = 1000;
     int signo;
     int ext_clkin=0;
     int ext_clkout=0;

        while((c = getopt(argc, argv, "hd:a:s:e:")) != EOF)
     {
               switch(c)
          {
               case 's':
                    speed = atoi(optarg);
                    if ((speed <= 0)||(speed>200000))
                    {
                         fprintf(stderr,"range error : 0< speed <= 2000000);
                         exit(0);
                    }
               break;
               case 'e':
                    if (strcmp(optarg,"i")==0 )
                    {
                          ext_clkin=1;
                    }
                    if (strcmp(optarg,"o")==0 )
                    {
                          ext_clkout=1;
                    }
               break;
               case 'a':
                    if (strcmp(optarg,"o")==0 )
                    {
                         mode|=0x4;
                    }
```

```
                else if (strcmp(optarg,"c")==0 )
                {
                     mode|=0x8;
                }
                else
                {
                     fprintf(stderr,"%s: invalid option -- '%s'0,argv[0],optarg);
                     fprintf(stderr,"Try '%s -h' for more information.0,argv[0]);
                     exit(-1);
                }
           break;
           case 'd':
                if (strcmp(optarg,"i")==0 )
                {
                     mode|=0x1;
                }
                else if (strcmp(optarg,"o")==0 )
                {
                     mode|=0x2;
                }
                else
                {
                     fprintf(stderr,"%s: invalid option -- '%s'0,argv[0],optarg);
                     fprintf(stderr,"Try '%s -h' for more information.0,argv[0]);
                     exit(-1);
                }
           break;
           case 'h':
           default:
                printf("-do: do test mode0);
                printf("-di: di test mode0);
                printf("-ao: ao test mode(oneshot)0);
                printf("-ac: ao test mode(continuous)0);
                printf("-s : ao speed:0< speed <= 2000000);
                printf("-ei: ao external clock in0);
                printf("-eo: ao external clock out0);
                exit(0);
           break;
     }
}
if (optind == argc)
{
           /* no more option */
     strcpy(devname,"/dev/pex340416/0");
   } else {
           strcpy(devname,argv[optind]);
   }

memset(DAC,0,sizeof(DAC));
memset(Volts,0,sizeof(Volts));
for(i=0;i<PEX340416_DAC_FIFO_SIZE;i++)
{
     n = (float)i/256.0;
     for(j=0;j<PEX340416_DAC_CHANNELS;j++)
     {
          switch(j)
          {
               case 0:
                    Volts[i][j] = 10.0 * sinf(2.0 * PI * n );
               break;
               case 1:
                    Volts[i][j] = 10.0 * cosf(2.0 * PI * n );
               break;
               case 2:
                    Volts[i][j] = -10.0 + (float)(20.0/256.0) * (float)i;
               break;
               case 3:
                    if (i&1) Volts[i][j] = 10.0 ; else Volts[i][j] = -10.0 ;
               break;
          }
          val = (unsigned int) (((Volts[i][j] +10.0) * 65536.0)/20.0);
          if (val >=0xffff) val = 0xffff;
          DAC[i][j]  =val;
     }
```

```
        }
           if ((fd = open(devname,O_RDWR)) == -1)
        {
             fprintf(stderr,"Device not found %s(%s)0,
             devname,strerror(errno));
                       exit(0);
        }
        printf("%s ",devname);
        //share user count
        ioctl(fd,IOCTL_EXTMEM_GET_SHARE_USERS,&shared);
        printf("share user %d0,shared);
        if (pex340416_init(fd,&dac,&clk,1)<0)
        {
             fprintf(stderr,"Device initialize error %s(%s)0,
             devname,strerror(errno));
                       exit(0);
        }
        pex340416_fd = fd;
        pex340416_dac = dac;
        pex340416_clk = clk;
        if (shared==1)
        {
             if (pex340416_setup_signal(fd,pex340416_interrupt_handler)<0)
             {
                       fprintf(stderr,"Interrupt initialize error %s(%s)0,
                       devname,strerror(errno));
                       exit(0);
             }
             if( ioctl(fd,IOCTL_EXTMEM_GET_SIGNAL_NUMBER,&signo)<0)
             {
                       fprintf(stderr,"extmem IOCTL_EXTMEM_GET_SIGNAL_NUMBER
fail %s0,strerror(errno));
                       exit(0);
             }
             if (ext_clkout==1)
                  pex340416_dac_write_clock_trigger(dac,PEX340416_DAC_CLOCK_TRIGGER_CTRL_CLK_EN);
           }

        if ( mode & 0x1 )
        {
             printf("Digital Input Test0);
             di_test(dac,clk);
        }
        if ( mode & 0x2 )
        {
             printf("Digital Output Test0);
             do_test(dac,clk);
        }
        if ( mode & 0x4 )
        {
             printf("Analog Output Test(oneshot %d Hz)0,speed);
             ao_test1(dac,signo,speed,ext_clkin);
        }
        if ( mode & 0x8 )
        {
             printf("Analog Output Test(continuous %d Hz)0,speed);
             ao_test2(dac,signo,speed,ext_clkin);
        }

           if (shared==1)
           {
                    pex340416_disable_clk_intrrupt(clk);
                    pex340416_disable_dac_intrrupt(dac);
           }
        if (pex340416_uninit(fd,dac,clk)<0)
        {
              fprintf(stderr,"%s0,strerror(errno));
        }
        close(fd);
   }
```

カウンタ制御関数 カウンタ読み出し 32 ビットレジスタを読み込み、32 ビットカウンタへデータをロードします。

```
int pex340416_clk_read_counter(PEX340416R *clk,unsigned int *value);
int pex340416_clk_write_counter(PEX340416R *clk,unsigned int *value);
```

戻り値
エラーなら-1 成功なら 0
引数
clk クロックカウンタ制御用メモリ領域へのポインタ。
*value 32bit 設定値へのポインタ

```
int pex340416_clk_read_mode(PEX340416R *clk,unsigned char *value);
int pex340416_clk_write_mode(PEX340416R *clk,unsigned char *value);
```

カウンタ部モード設定レジスタ制御関数
戻り値
エラーなら-1 成功なら 0
引数
clk クロックカウンタ制御用メモリ領域へのポインタ。
*value 8bit 設定値へのポインタ(下記値の論理和を使用する)
(EQS|DIR|MDSEL[0-14])

```
PEX340416_CLK_MODE_SETTING_EQS_ENABLE   カウンタ値一致検出機能有効(1)
PEX340416_CLK_MODE_SETTING_DIR_REVERSE  カウント方向を設定リバース方向(1)
PEX340416_CLK_MODE_SETTING_DIR_FOWARD   カウント方向を設定通常方向(0)
PEX340416_CLK_MODE_SETTING_MDSEL0       ゲート付き単相パルス 1 逓倍モード
PEX340416_CLK_MODE_SETTING_MDSEL1       ゲート付き単相パルス 2 逓倍モード
PEX340416_CLK_MODE_SETTING_MDSEL4       位相差パルス 1 逓倍モード 非同期クリア
※ 1
PEX340416_CLK_MODE_SETTING_MDSEL5       位相差パルス 2 逓倍モード 非同期クリア
※ 1
PEX340416_CLK_MODE_SETTING_MDSEL6       位相差パルス 4 逓倍モード 非同期クリア
※ 1
PEX340416_CLK_MODE_SETTING_MDSEL8       2 パルスカウントモード*/
PEX340416_CLK_MODE_SETTING_MDSEL12      位相差パルス 1 逓倍モード 同期クリア ※
2
PEX340416_CLK_MODE_SETTING_MDSEL13      位相差パルス 2 逓倍モード 同期クリア ※
2
PEX340416_CLK_MODE_SETTING_MDSEL14      位相差パルス 4 逓倍モード 同期クリア ※
2
※ 1 非同期クリア:
A 相, B 相のステータスに関わりなく、 Z 相または Z 相+L1 が有効になった時、カウンタはクリアさ
れます。
※ 2 同期クリア:
B 相が“Low”で、Z 相または Z 相+L1 が有効になった時、A 相の(↑)と(↓)にてカウンタはクリアさ
れます
```

```
int pex340416_clk_read_connector(PEX340416R *clk,unsigned char *value);
```

カウンタ部端子状態・入力パルス有効/無効設定状態読み込みレジスタ 制御関数
戻り値
エラーなら-1 成功なら 0
引数
clk クロックカウンタ制御用メモリ領域へのポインタ。
*value 8bit 設定値へのポインタ(下記値の論理和を使用する)
(EN|L1|Z|B|A)

```
PEX340416_CLK_MODE_SETTING_EQS_ENABLE       カウンタ値一致検出機能有効
PEX340416_CLK_CONNECTER_EN                  入力パルス有効(0)/無効(1)
PEX340416_CLK_CONNECTER_L1                  L1 端子状態(0:Low / 1: Hi)
PEX340416_CLK_CONNECTER_Z           Z 端子状態(0:Low / 1: Hi)
PEX340416_CLK_CONNECTER_B           B 端子状態(0:Low / 1: Hi)
PEX340416_CLK_CONNECTER_A           A 端子状態(0:Low / 1: Hi)
```

```
int pex340416_clk_write_connector(PEX340416R *clk,unsigned char *value);
```

カウンタ部入力パルス有効/無効設定・カウンタ/比較レジスタ選択レジスタ 制御関数
戻り値
エラーなら-1 成功なら 0
引数
clk クロックカウンタ制御用メモリ領域へのポインタ。
*value 8bit 設定値へのポインタ(下記値の論理和を使用する)
(入力パルス|カウンタまたは、比較レジスタ)

```
PEX340416_CLK_CONNECTER_INPUT_PULSE_DISABLE  入力パルス無効(1)
PEX340416_CLK_CONNECTER_INPUT_PULSE_ENABLE   入力パルス有効(0)
PEX340416_CLK_CONNECTER_INPUT_COUNT          カウンタ(0)選択
PEX340416_CLK_CONNECTER_INPUT_COMPARE        比較レジスタ(1)選択
```

```
int pex340416_clk_read_status(PEX340416R *clk,unsigned char *value);
```

カウンタ部ステータスレジスタ 制御関数
戻り値

エラーなら-1 成功なら 0

引数

clk　クロックカウンタ制御用メモリ領域へのポインタ。

*value　8bit 設定値へのポインタ(下記値の論理和を使用する)

(PERR|EQF|EXLTS|EQ|CBF|UD)

PEX340416_CLK_STATUS_PERR　位相差パルスカウントモードでの異常入力検出です。値をリードするとクリアされます。

PEX340416_CLK_STATUS_EQF　カウンタ値一致。値をリードするとクリアされます

PEX340416_CLK_STATUS_EXLTS　外部ラッチ。値をリードするとクリアされます

PEX340416_CLK_STATUS_EQ　カウンタと比較カウンタが一致しているかどうかを表します。

PEX340416_CLK_STATUS_CBF　キャリー/ボロー。値をリードするとクリアされます

PEX340416_CLK_STATUS_UD　現在のカウンタの方向を表します。 0:ダウンカウント 1:アップカウント

int pex340416_clk_write_status(PEX340416R *clk,unsigned char *value);

カウンタ部ソフトウェアラッチクリアレジスタ 制御関数

戻り値

エラーなら-1 成功なら 0

引数

clk　クロックカウンタ制御用メモリ領域へのポインタ。

*value　8bit 設定値へのポインタ(下記値の何れかを使用する)

PEX340416_CLK_STATUS_SOFT_LATCH　カウンタ値をリードし、レジスタに移動する。(ソフトウェアラッチ)

PEX340416_CLK_STATUS_SOFT_CLEAR　カウンタ,リードレジスタをクリアする。 (ソフトウェアクリア)

int pex340416_clk_write_lpzpltscls(PEX340416R *clk,unsigned char *value);

カウンタ部外部ラッチ, クリア, Z 相論理設定レジスタ制御関数

戻り値

エラーなら-1 成功なら 0

引数

clk　クロックカウンタ制御用メモリ領域へのポインタ。

*value　8bit 設定値へのポインタ(下記値の論理和を使用する)

(LP|ZP|LTS[012]|CLS[012])

PEX340416_CLK_LPZPLTSCLS_LP_RISE　L 論理設定 L の立ち上がりを有効(“High”で有効)

PEX340416_CLK_LPZPLTSCLS_LP_FALL　L 論理設定 L の立ち下がりを有効(“Low”で有効)

PEX340416_CLK_LPZPLTSCLS_ZP_HIGH　Z 相論理設定 Z 相反転(“Low”で有効)

PEX340416_CLK_LPZPLTSCLS_ZP_LOW　Z 相論理設定 Z 相通常(“High”で有効)

PEX340416_CLK_LPZPLTSCLS_LTS0　外部ラッチ設定 カウンタの外部ラッチを行わない。

PEX340416_CLK_LPZPLTSCLS_LTS1　外部ラッチ設定 L のみでカウンタをラッチする。

PEX340416_CLK_LPZPLTSCLS_LTS2　外部ラッチ設定 L と Z 相が有効の時、カウンタをラッチする。

PEX340416_CLK_LPZPLTSCLS_CLS0　外部クリア設定 カウンタの外部クリアを行わない。

PEX340416_CLK_LPZPLTSCLS_CLS1　外部クリア設定 Z 相のみでカウンタをクリアする。

PEX340416_CLK_LPZPLTSCLS_CLS2　外部クリア設定 L と Z 相が有効の時、カウンタをクリアする。

int pex340416_clk_read_filter(PEX340416R *clk,unsigned char *value);

int pex340416_clk_write_filter(PEX340416R *clk,unsigned char *value);

カウンタ部デジタルフィルタ設定レジスタ制御関数

戻り値

エラーなら-1 成功なら 0

引数

clk　クロックカウンタ制御用メモリ領域へのポインタ。

*value　8bit 設定値へのポインタ(下記値の論理和を使用する)

PEX340416_CLK_MODE_SETTING_EQS_ENABLE　カウンタ値一致検出機能有効

*value = (フィルタ基準クロック 2bit|フィルタカウンタ 6bit)

フィルタ基準クロック

PEX340416_CLK_DFIL_STOP　下位ビットは、デジタルフィルタのカウンタ

PEX340416_CLK_DFIL_1US　下位ビットは、デジタルフィルタのカウンタ

PEX340416_CLK_DFIL_10US　下位ビットは、デジタルフィルタのカウンタ

PEX340416_CLK_DFIL_100US　下位ビットは、デジタルフィルタのカウンタ

使用例

カウンタ値の取得方法

3 モードパルスカウンタインタフェースモジュールのカウンタ値を読み込みます。
(1)カウントモードの設定を行います。
カウンタ/デジタル入出力選択
カウンタ機能有効に設定
```
value8=PEX340416_CLK_SELECT_COUNTER;
pex340416_set_byte_mmap(clk,PEX340416_CLK_SELECT_OFFSET,&value8);
```
カウンタ部モード設定
位相差パルス，4 逓倍モード，非同期クリアに設定
```
value8=PEX340416_CLK_MODE_SETTING_MDSEL6;
pex340416_clk_write_mode(clk,&value8);
```
カウンタ部ソフトウェアラッチ/クリア
カウンタクリアコマンドを発行
```
value8=PEX340416_CLK_STATUS_SOFT_CLEAR;
pex340416_clk_write_status(clk,&value8);
```
(2)パルスの入力を開始します。
(3)カウンタ値を取得したいタイミングでカウンタラッチコマンドを発行します。
カウンタラッチコマンドを発行
```
value=PEX340416_CLK_STATUS_SOFT_LATCH;
pex340416_clk_write_status(clk,&value);
```
(4)カウンタ値の取得を行います。
```
pex340416_clk_read_counter(clk,&value32);
```
(5)取得したカウンタ値の表示を行います。
```
printf("0x08X0,value32);
```

カウンタ値の設定方法
3 モードパルスカウンタインタフェースモジュールへカウンタ値を書き込みます。
(1)カウントモードの設定を行います。
カウンタ機能有効に設定
```
value8=PEX340416_CLK_SELECT_COUNTER;
pex340416_set_byte_mmap(clk,PEX340416_CLK_SELECT_OFFSET,&value8);
```
カウンタ部モード設定
位相差パルス，4 逓倍モード，非同期クリアに設定
```
value8=PEX340416_CLK_MODE_SETTING_MDSEL6;
pex340416_clk_write_mode(clk,&value8);
```
(2)書き込みを行うカウンタを選択します。
書き込み先にカウンタプリレジスタを指定
```
value=PEX340416_CLK_CONNECTER_INPUT_COUNT;
pex340416_clk_write_connector(clk,&value);
```
(3)カウンタ値を設定します。
```
value32 = カウンタ値;
pex340416_clk_write_counter(clk,&value);
```

比較カウンタ設定方法
3 モードパルスカウンタインタフェースモジュールへカウンタ値を書き込みます。
(1)カウントモードの設定を行います。
カウンタ機能有効に設定
```
value8=PEX340416_CLK_SELECT_COUNTER;
pex340416_set_byte_mmap(clk,PEX340416_CLK_SELECT_OFFSET,&value8);
```
カウンタ部モード設定
位相差パルス，4 逓倍モード，非同期クリアに設定
```
value8=PEX340416_CLK_MODE_SETTING_MDSEL6;
pex340416_clk_write_mode(clk,&value8);
```
(2)書き込みを行うカウンタを選択します。
書き込み先に比較レジスタを指定
```
value=PEX340416_CLK_CONNECTER_INPUT_COMPARE;
pex340416_clk_write_connector(clk,&value);
```
(3)比較カウンタ値を設定します。
```
value32 = 比較カウンタ値;
pex340416_clk_write_counter(clk,&value);
```

割り込み処理設定方法
割り込み処理を行う場合、以下の処理を行う必要があります。
シグナルハンドラーの登録
```
pex340416_setup_signal(fd,pex340416_interrupt_handler);
```

シグナルハンドラー定義
```
static void pex340416_interrupt_handler(int signo,siginfo_t *info, void *ptr)
{
    unsigned long int iflag;
```

```
                /*割り込みが発生すると、割り込みフラグレジスタの各ビットが ON になります。このフラグで発生した割り込みの要因
を確認します。*/
                pex340416_intr_service(pex340416_fd,&iflag,&pex340416_pending);
                dac_iflag = (iflag    ) & 0xFFFF;
                clk_iflag = (iflag>>16) & 0xFF;
            }
       シグナル番号の確認、ディフォルトでは SIGIO。
            ioctl(fd,IOCTL_EXTMEM_GET_SIGNAL_NUMBER,&signo);
       シグナルマスクの設定
               sigfillset(&imask);
               sigdelset(&imask,SIGINT);
               sigdelset(&imask,signo);
       割り込みの許可
            pex340416_enable_clk_intrrupt(clk,PEX340416_CLK_IFR_ALL);
       PEX340416_CLK_IFR_ALL 割り込みシグナルの待ちと、要因別処理
            sigsuspend(&imask);
               if (clk_iflag & PEX340416_CLK_PERR)
                 /* 異常入力検出での割り込み 0:割り込み未検出 1:割り込み検出*/
               if (clk_iflag & PEX340416_CLK_C_B)
                 /* キャリー/ボローでの割り込み 0:割り込み未検出 1:割り込み検出*/
               if (clk_iflag & PEX340416_CLK_EXLT)
                 /* 外部ラッチでの割り込み 0:割り込み未検出 1:割り込み検出*/
               if (clk_iflag & PEX340416_CLK_EQ)
                 /* カウンタと比較カウンター致での割り込み 0:割り込み未検出 1:割り込み検出*/
       割り込みの禁止
               pex340416_disable_clk_intrrupt(clk);

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

AUTHORS
       Copyright (C) 1995-2023 Concurrent Real Time Inc.

                   30 Jan 2023                                    pex361116(3)
```