AO-1616L 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、コンテック社製 AO- 1616L 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

AO- 1616L 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。

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

AO-1616Lは、パソコンにアナログ信号の出力機能を拡張するLow Profile対応のPCI Expressバス対応ボードです。

3. インストール方法

AO-1616L 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。

AO-1616L 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

AO1616LL BSP 実行パッケージのインストール

rpm -ivh bin-ao1616ll-**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- ao1616ll-**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

AO1616LL BSP 実行パッケージのインストール

apt install ./bin-ao1616ll-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-ao1616ll-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

AO1616LL BSP 実行パッケージのインストール
apt install ./bin-ao1616ll-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-ao1616ll-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。)

AO-1616L BSP パッケージは ***/usr/local/CNC/drivers/extmem/interface/ ao1616l*** 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

AO-1616L 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/ ao1616l*** 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- ao1616ll-**X.Y**_RH***x.y-z***.x86_64 *(開発パッケージの削除)*
rpm -e bin-ao1616ll-**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-ao1616ll-**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-ao1616ll-rhx.y *(開発パッケージの削除)*
apt purge bin-ao1616ll-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-ao1616ll-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-ao1616ll-rhx.y *(開発パッケージの削除)*
apt purge bin-ao1616ll-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-ao1616ll-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 ao1616l
ao1616l(3) ao1616l(3)

NAME

ao1616l - 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

ao1616l は、external memory ドライバを利用した ao1616l ボードアクセスライブラリです。

```
#include <sys/ao1616l.h>
gcc [options ...] file -lao1616l -lextmem ...
```

****

AO-1616L-LPE をアクセスする関数群

****

デバイスの非初期化処理

```
int ao1616l_reset(int fd,int port);
```

port は、以下のビットオアまたは、全てを示す AO1616L_ALL_PORT である

```
AO1616L_AO_PORT
AO1616L_DI_PORT
AO1616L_DO_PORT
AO1616L_COUNTER_PORT
AO1616L_MEMORY_PORT
AO1616L_ECU_PORT
int ao1616l_uninit(int fd);
```

デバイスの初期化処理

```
int ao1616l_init(int fd,int option);
```

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 ao1616l_get_ioport_long(int fd,int offset,unsigned long int
*value);
int ao1616l_get_ioport_short(int fd,int offset,unsigned long int
*value);
```

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 ao1616l_set_ioport_long(int fd,int offset,unsigned long int
*value);
int ao1616l_set_ioport_short(int fd,int offset,unsigned long int
*value);
```

割り込みハンドラの登録

```
int ao1616l_setup_signal(int fd,void (*interrupt_hadler)( int, siginfo_t *, void *),unsigned
int aomask,unsigned int dimask,unsigned int comask,unsigned int
memask);
```

割り込みを許可する

```
int ao1616l_enable_intrrupt(int fd,unsigned long int aomask,unsigned
long int dimask,unsigned long int comask,unsigned long int memask);
```

割り込みを禁止する

```
int ao1616l_disable_intrrupt (int fd,unsigned int aomask,unsigned int
dimask,unsigned int comask,unsigned int memask);
```

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の値を戻す

int ao1616l_intr_service(int fd,unsigned long int *iflag, unsigned long int *aoflag,unsigned long int *diflag,unsigned long int *coflag,unsigned long int *meflag,unsigned long int *pending);

イベントコントローラ入出力信号の結線を行う
int ao1616l_set_signal_assign(int fd,unsigned long int destnation_signal,unsigned long int source_signal);

イベントコントローラ入出力信号の結線を得る
int ao1616l_get_signal_assign(int fd,unsigned long int destnation_signal,unsigned long int source_signal,unsigned long int *destnation_signal_ret,unsigned long int *source_signal_ret);
設定できる信号は以下の通り
destnation_signal
DST_SIGNAL_AO_UPDATE_ENABLE_TRIGGER : AO 更新許可トリガ 32 20
DST_SIGNAL_AO_UPDATE_DISABLE_TRIGGER : AO 更新不許可トリガ 34 22
DST_SIGNAL_UPDATE_CLOCK : 更新クロック 36 24
DST_SIGNAL_CNTEXTSTATUS00 : CNTExtStatus00 116 74
DST_SIGNAL_COUNT_START_TRIGGER0 : カウント開始トリガ 0 128 80
DST_SIGNAL_COUNT_STOP_TRIGGER0 : カウント停止トリガ 0 130 82
DST_SIGNAL_COUNT_UP0 : カウント 0 対象/アップカウント 134 86
source_signal
SRC_SIGNAL_NO_CONNECTION : Non Connection 0 00 固定
SRC_SIGNAL_AO_INTERNAL_CLOCK : 内部更新クロック(Internal CLK for AO) 66 42 クロック
SRC_SIGNAL_AO_BEFORE_TRIGGER_NUM_END : ビフォートリガ更新回数終了 (AO Before Trigger Num End) 80 50 フラグ
SRC_SIGNAL_DI_AO_EXTCLK_START_EDGE : AO Ext Start Edge 147 93
SRC_SIGNAL_DI_AO_EXTCLK_STOP_EDGE : AO Ext Stop Edge 148 94
SRC_SIGNAL_DI_AO_EXTCLK : AO Ext CLK 149 95
SRC_SIGNAL_DI_CNT_EXTUCLK1 : CNT Ext UCLK0 150 96
SRC_SIGNAL_COUNT_INTERNAL0 : 内部標準タイマ 0 (Internal CLK For CNT0) 272 110
SRC_SIGNAL_COUNT_COUNTUP0 : 比較カウントー致 0 (CNT0 Count Up) 288 120
SRC_SIGNAL_COUNT_BUSY0 : CNT Busy0 305 131
SRC_SIGNAL_MEMORY_AO_DATA_EMPTY : AO Data Empty 352 160
SRC_SIGNAL_MEMORY_GENERAL_COMMAND0 : 汎用コマンド 0 (General Command 0) 384 180

汎用パルス出力コマンド
int ao1616l_raise_signal(int fd);
このコマンドは、汎用性のあるパルスを出力する機能です。全機能に利用で き

ます。
このコマンドには、設定データ確認/設定ポートへの I/O は不要です。
各機能の動作をソフトウェアで制御する場合に有効な機能です。
例 えば、AO 格納許可トリガ”に利用するとソフトウェアスタートが可能にな
ります。

int ao1616l_ao_abort_settings(int fd,unsigned long int value);
例えば、AO 格納許可トリガ”に利用するとソフトウェアスタートが可能に な
ります。
イベントコントローラに、AO 異常停止設定コマンドを値 value で出力する
こ のコマンドでは、AO 異常停止信号を設定します。ここで 1 と設定された信号
は AO 機能に異常
信号として入力されます。複数選択可能です。
この異常信号が入力されると AO 機能は更新の途中であっても動作を停止し ま
す。詳細の動作は AO
機能で記述します。
ただし、この機能によって生成される異常信号は IRQ 要因になりません。

int ao1616l_ao_setcommand(int fd,unsigned long int command,unsigned
long int value);
int ao1616l_ao_setcommand2(int fd,unsigned long int command,unsigned
long int value1,unsigned long int value2);
アナログ出力機能設定コマンド
command,value は以下の通り
command value AO1616L_AO_RESET_COMMAND AO 初期化コマン
ド
value=0 を設定
AO1616L_AO_GATE_OPEN_COMMAND AO 内部ゲートオープンコマンド
value=0 を設定
AO1616L_AO_ABORT_COMMAND AO 強制停止コマンド
value=0 を設定
AO1616L_AO_INTCLK_COMMAND AO 用内部クロック設定
value=内部クロックのパルス周期 = 周期(ns)/25-1
AO1616L_AO_CHNUM_COMMAND AO 更新チャネル数設定
このコマンドは、AO1616L_AO_CHNUM_SINGLE のとき
はチャネルを、AO1616L_AO_CHNUM_MULTI のときは更新するチャネル数を設定し
ます。
AO1616L_AO_CHNUM_SINGLE
·AO1616L_AO_CHNUM_MULTI は AO 入力切換えコマンドで設定します。
AO1616L_AO_CHNUM_SINGLE 時:AO 機能の更新したい
チャネルを設定します。設定データ=更新チャネル数
AO1616L_AO_CHNUM_MULTI 時 :AO 機能の更新した い
最大チャネル数を設定します。設定データ=更新チャネル数-1
AO1616L_AO_BEFTRG_COMMAND AO ビフォートリガ更新回数設定
value = ビフォートリガ更新回数-1
こ のコマンドでは、AO 更新許可トリガが入力され
た後の更新回数を設定します。
このコマンドは、"ビフォートリガ更新回数終了ト
リガ"を生成するためにあります。
主に更新回数で AO 機能を制御する場合に利用しま
す。
AO ビフォートリガ更新回数のカウント条件
カウント期間: 更新許可トリガから更新 不
許可トリガまで

カウント条件: 更新クロック入力時
カウントロード: コマンド設定時、AO 強制停止コマンド出力時、異常停止入力時、更新許可トリガ
入力時、ビフォートリガサンプ リング回数終了時
注 意 : "ビフォートリガサンプリング回数終了”フラグについて
・カレント更新回数がこのコマンドで設 定した 回数に 達した場合 にフラグセンスポートに
セットされます。
・リセットは ECU 機能にあります。
・Destination として使用しなくても、このフラグは ECU 機能へ出力されます。
注意 : 無限更新について
このコマンドで無限更新を実現することはできません。その場合は更新不許可トリガを"汎
用コマンド"にしてください。
注意 : 設定回数の制限
ハードの搭載バッファメモリの容量は最大256WORD(0x100h)です。 それ以上の更新回数を設
定する場合では、256WORD 分のアナログ 出力データがバッファメモリに書き込まれる前に
外部(DA コンバータ)に出力される必要があります。
AO1616L_AO_SWITCH_COMMAND AO 切り換え設定
単数更 新(AO1616L_AO_CHNUM_SINGLE) と 複 数 更新(AO1616L_AO_CHNUM_MULTI)の切り換えを設定します。
注 意 このコマンドは必ず AO 更新チャネル数設定コマンドを設定する前に設定してください。
value=AO1616L_AO_CHNUM_SINGLE or value=AO1616L_AO_CHNUM_MULTI
AO1616L_AO_CAL_COMMAND AO キャリブレーション条件設定
以 下の値のオアをとったもの(ajdust(data)|ch|range|select)を設定する
select
AO1616L_AO_CAL_OFFSET
AO1616L_AO_CAL_GAIN
range
AO1616L_AO_CAL_RANGE_M10VtoP10V(-10 to +10)
ch:
AO1616L_AO_CAL_CH(ch)
adjust
AO1616L_AO_CAL_ADJUST(data)

int ao1616l_ao_calibration(int fd,unsigned long int ch);
AO-1616L-LPE テクニカルリファレンスマニュアル 31/64 を参照してください
int ao1616l_memory_setcommand(int fd,unsigned long int command);
command,value は以下の通り
command value
AO1616L_MEMORY_RESET_COMMAND MEM 初期化コマンド
AO1616L_MEMORY_AO_MEM_CLEAR_COMMAND AO MEM クリア
このコマンドでは、以下の内容を行います。このコマンドには、設定データ確認/設定ポートへの I/O

は不要です。
FIFO カウンタをリセットします。
AO 用 MEM 比較関連コマンドを再度設定する場合や新たに更新動作する場合では、 このコマンドを出力してください。
注意 : 更新途中(AO 機能動作中)にこのコマンドを動作させると、MEM 機能関連のフラグ等が正常に動作しない場合があります。
int ao1616l_memory_getcommand(int fd,unsigned long int command,long int *value);
command,value は以下の通り
AO1616L_MEMORY_AO_FIFO_COMMAND AO FIFO カウンタ確認コマンド
このポートでは、AO 用バッファメモ リの FIFO カウンタの現在値を確認します。
I/O アドレス"+30h"ポートを出力したときに現在の FIFO カウンタ値がラッチされます。
FIFO カウンタデータ AO 機能のデー タポートから出力して、
まだ DA コンバータへ出力されていないデータ数(バッファメモリ内に蓄積された残データ数)
int ao1616l_memory_setcommand2(int fd,unsigned long int command,unsigned long int value1,unsigned long int value2);
command,value は以下の通り
AO1616L_MEMORY_AO_MEM_COMPARETE_TYPE_COMMAND AO 用 MEM 比較タイプ設定
このコマンドでは、バッファメモリに格納される任意のデータ数・ポインタでフラグをセットする条件のうち比較タイプを設定します。
このコマンドでは、"AO 比較データ数ホールドフラグ"もしくは"AO 転送数フラグ"を生成するためにあります。
value1
AO1616L_MEMORY_AO_MEM_COMPARETE_TYPE_DATA: データ数

AO1616L_MEMORY_AO_MEM_COMPARETE_TYPE_TRAS: 転送数
value2
AO1616L_MEMORY_AO_MEM_COMPARETE_TYPE_IGNORE: 無効
AO1616L_MEMORY_AO_MEM_COMPARETE_TYPE_UNDER: 下方一致
AO1616L_MEMORY_AO_MEM_COMPARETE_TYPE_EVERY: 設定転送数毎
詳細:
"下方一致"の場合:設定したデータ数からさらに読み込まれた(アナログ出力データが D/A コンバータ へ出力された)場合に"AO 比較データ数ホールド"フラグがセットされます。
AO 機能のアナログ出力ポートからア ナログ出力データを追加の書き込みによって、データ数が増え、設定したデータ数を通過した場合は、このフラグはセットされません。
"設定転送数毎"の場合:ポインタの移動量が設定した値に等しくなった場合に"AO 転送数"フラグ

がセットされます。
注 意："AO 比較データ数ホールド"フラグ、"AO 転送数"フラグについて
このコマンドの他に AO 用 MEM 比較データ設定の設定が必要です。
リセットは ECU の機能にあります。
注意：リセット
サンプリング終了後に FIFO カウンタをリセットする場合は以下のコマンドを使ってくださ
い。MEM 初期化コマンド
注意：コマンド設定項目制限
このコマンドは比較番号が"データ数"の場合は比較タイプが"無効"か"下方一致”しか選
択できません。また、比較番号が"転送数"の場合は比較タイプが"無効"か"設定転送数毎"
しか選択できません。
AO1616L_MEMORY_AO_MEM_COMPARETE_DATA_COMMAND: //AO 用 MEM 比較データ設定
このコマンドでは、バッファメモリ に格納される任意のデータ数・ポインタでフラグをセットする条
件のうちポインタ/データ数を設定します。
このコマンドでは、"AO 比較データ 数ホールド"フラグもしくは"AO 転送数"フラグを生成するた
めにあります。
value1
AO1616L_MEMORY_AO_MEM_COMPARETE_TYPE_DATA: データ数
AO1616L_MEMORY_AO_MEM_COMPARETE_TYPE_TRAS: 転送数
value2
設定データ ＝ 比較データ数(転送数) +1
設定例

設定データ	データ数("AO 比較データ数ホールド"フラグがセットされる条件)	転送数("AO 転送数"フラグがセットされる条件)
0	動作せず	動作せず
1	アナログ出力データ数が 1 個でさらにアナログ出力データが 読み込まれた場合。	バッファメモリから DA コンバータへアナログ読み込まれた場合。
101(65h)	アナログ出力データ数が 101 個でさらにアナログ出力データが 読み込まれた場合。	バッファメモリから DA コンバータへ 101 個の読み込まれた場合。

|アナログ出力データが出力される毎に。 |
| |つまり残 デ ー タ 数 が 100 個 に な っ た 場 合 。
| |
+---------------+--+--+
| 256(100h) |アナログ出力データ数が 256 個でさらにアナログ出力デ ー
タが |バッファメモリから DA コンバータへ 101 個の |
| | 読 み 込 ま れ た 場 合 。
|アナログ出力データが出力される毎に。 |
| |つまり残 デ ー タ 数 が 255 個 に な っ た 場 合 。
| |
+---------------+--+--+

int ao1616l_di_setcommand2(int fd,unsigned long int command,unsigned
long int value1,unsigned long int value2);
command,value1,value2 は以下の通り
AO1616L_DI_RESET_COMMAND DI 初期化コマンド
value1=0,value2=0 を設定
AO1616L_DI_EDGE_DETECT_COMMAND DI エッジ検出設定
value1: AO1616L_DI_EDGE_TYPE_AO or
AO1616L_DI_EDGE_TYPE_CNT
value2:以下の値をシフトしたもの
AO1616L_DI_EDGE_TYPE_NONE
AO1616L_DI_EDGE_TYPE_RISE
AO1616L_DI_EDGE_TYPE_FALL
AO1616L_DI_EDGE_BIT00_SHIFT:AO
Ext Start
AO1616L_DI_EDGE_BIT01_SHIFT:AO
Ext Stop
AO1616L_DI_EDGE_BIT02_SHIFT:AO
Ext CLK
AO1616L_DI_DIGITAL_FILTER_COMMAND DI Digital Filter 設 定
value1:
AO1616L_DI_DIGITAL_FILTER_AO
AO1616L_DI_DIGITAL_FILTER_CNT
AO1616L_DI_DIGITAL_FILTER_OFF
value2: 0 or AO1616L_DI_DIGITAL_FILTER_1U

int ao1616l_di_getvalue(int fd,unsigned long int *value);
DI の値を読み出す
int ao1616l_di_setmask(int fd,unsigned long int value);
DI のマスクを設定する

int ao1616l_do_setmask(int fd,unsigned long int value);
DO のマスクを設定する
int ao1616l_do_setvalue(int fd,unsigned long int value);
DO に値を設定する
int ao1616l_cnt_getcommand(int fd,unsigned long int command,long int
*value);
int ao1616l_cnt_setcommand(int fd,unsigned long int command,unsigned
long int value);
AO1616L_COUNTER_DATA_COMMAND CNT カレントカウントデータ設定/
確認
int ao1616l_get_system_timer(int fd,unsigned long int *value);
value:カレントカウントデータ設定/確認
int ao1616l_ao_start(int fd);
AO をスタートする
int ao1616l_ao_write(int fd,unsigned long int *addr,int nch);

配列 addr のデータを nch 分 AO に書き出す

SEE ALSO

/usr/local/CNC/drivers/extmem/contec/ao1616l 下のプログラム

AUTHORS