AI-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 AI-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

AI-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。

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

AI-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

AI-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。

AI-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

AI1616L BSP 実行パッケージのインストール

rpm -ivh bin-ai1616l-**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- ai1616l-**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

AI1616L BSP 実行パッケージのインストール

apt install ./bin-ai1616l-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-ai1616l-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

AI1616L BSP 実行パッケージのインストール
apt install ./bin-ai1616l-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-ai1616l-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。)
AI-1616L BSP パッケージは ***/usr/local/CNC/drivers/extmem/interface/ai1616l*** 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

AI-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/ai1616l 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- ai1616l-**X.Y**_RH***x.y-z***.x86_64 *(開発パッケージの削除)*
rpm -e bin-ai1616l-**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-ai1616l-**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-ai1616l-rhx.y *(開発パッケージの削除)*
apt purge bin-ai1616l-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-ai1616l-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-ai1616l-rhx.y *(開発パッケージの削除)*
apt purge bin-ai1616l-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-ai1616l-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 ai1616l
ai1616l(3) ai1616l(3)

NAME
ai1616l - 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
ai1616l は、external memory ドライバを利用した AI1616L ボードアクセスライブラリです。

#include <sys/ai1616l.h>
gcc [options ...] file -lai1616l -lextmem ...

****
AI1616L
****

マルチファンクションボード AI1616L をアクセスする関数群
API のパラメータ変数がポインタでない場合は、unsigned long int に定義さ れていても、有効な値は 32 ビットである。

デバイスの非初期化処理
int ai1616l_reset(int fd,int port);
port は、以下のビットオアまたは、全てを示す AI1616L_ALL_PORT である
AI1616L_AI_PORT
AI1616L_DI_PORT
AI1616L_DO_PORT
AI1616L_COUNTER_PORT
AI1616L_MEMORY_PORT
AI1616L_ECU_PORT

int ai1616l_uninit(int fd); 注意：ai1616l_uninit()関数は ai1616l_reset() を呼び出す。
デバイスの初期化処理と、割り込みサービスの登録
int ai1616l_init(int fd,int option);
割り込みサービスの登録を行い、ボードを初期化する。
割り込み時のレジスタの値は、ai1616l_intr_service()API で得られ、割り込みフラグクリアは、extmem デバイスドライバによって終了している。
注意：DO は割り込まない。

option 1を指定すると以下の情報が表示される
BAR0 I/O Region addr 0x0000ccc0 offset 0x00000000 64 bytes
注意：ai1616l_init()関数は ai1616l_reset()を呼び出す。

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 ai1616l_get_ioport_long(int fd,int offset,unsigned long int *value);
int ai1616l_get_ioport_short(int fd,int offset,unsigned long int *value);
ai1616l_get_ioport_long()は、32bit、ai1616l_get_ioport_short()が 16bit で
ある。

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 ai1616l_set_ioport_long(int fd,int offset,unsigned long int *value);
int ai1616l_set_ioport_short(int fd,int offset,unsigned long int *value);
ai1616l_set_ioport_long()は、32bit、ai1616l_set_ioport_short()が 16bit で ある。

割り込みハンドラの登録と、割り込みの許可
int ai1616l_setup_signal(int fd,void (*interrupt_hadler)(int, siginfo_t *, void *),unsigned long int aimask,unsigned long int dimask,unsigned long int comask,unsigned long int memask);
各マスク値は、1 である場合に許可、0 である場合に不許可である(実際の書き出 しには、NOT の値を使用する）
この関数の呼び出しによって、ボードの IRQ 信号によって、シグナル ハ ンドラが起動されるようになる。
また、本関数は、ai1616l_enable_intrrupt()を呼び出す。

割り込みを許可する
int ai1616l_enable_intrrupt(int fd,unsigned long int aimask,unsigned long int dimask,unsigned long int comask,unsigned long int memask);
各マスク値は、1 である場合に許可、0 である場合に不許可である(実際の書き出 しには、NOT の値を使用する）

割り込みを禁止する
int ai1616l_disable_intrrupt (int fd,unsigned long int aimask,unsigned long int dimask,unsigned long int comask,unsigned long int memask);
各マスク値は、1 である場合に不許可、0 である場合に許可である(実際の書き出 にその値を使用する）

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の値を戻す
int ai1616l_intr_service(int fd,unsigned long int *iflag, unsigned long int *aiflag,unsigned long int *diflag,unsigned long int *coflag,unsigned long int *meflag,unsigned long int *pending);
通常本関数は、ai1616l_setup_signal()関数で登録したシグナルハンドラ内で呼び出される。
ファイルディスクリプタ(fd)以外の変数は、割り込み時のレジスタ値である。
iflag:ECU 機能確認ポート 0x38 の値
aiflag:AI 関連のフラグ
diflag:DI 関連のフラグ
coflag:COUNTER 関連のフラグ
meflag:MEMORY 関連のフラグ
pending:処理がペンディングされている数

イベントコントローラ入出力信号の結線を行う
int ai1616l_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 ai1616l_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だが、下記組み合わせが存在し、不可能な組み合わせではエラー(-1)になる。
destnation_signal
DST_SIGNAL_AI_STORE_ENABLE_TRIGGER AI 格納許可トリガ[0x00]
DST_SIGNAL_AI_STORE_DISABL_TRIGGER AI 不許可トリガ[0x02]
DST_SIGNAL_SAMPLING_CLOCK AI サンプリングクロック[0x04]
DST_SIGNAL_CNTEXTSTATUS00 CNT 外部ステータス[0x74]
DST_SIGNAL_COUNT_START_TRIGGER0 CNT 開 始トリガ[0x80]

DST_SIGNAL_COUNT_STOP_TRIGGER0　　CNT 停止トリガ[0x82]
DST_SIGNAL_COUNT_UP0　　CNT アップカウント[0x86]

source_signal

SRC_SIGNAL_NO_CONNECTION　　Non Connection[0x000]
SRC_SIGNAL_AI_INTERNAL_CLOCK　　AI 内部サンプリングクロック[0x004]
SRC_SIGNAL_AI_BEFORE_TRRIGER_NUM_END　　AI ビフォートリガサンプリング回数終了[0x011]
SRC_SIGNAL_DI_AI_EXTCLK_START_EDGE　　AI 外部開始エッジ[0x090]
SRC_SIGNAL_DI_AI_EXTCLK_STOP_EGDEAI 外部停止エッジ[0x091]
SRC_SIGNAL_DI_AI_EXTCLK　　AI 外部クロック[0x092]
SRC_SIGNAL_DI_CNT_EXTUCLK1　　CNT 外部アップカウントクロック[0x096]
SRC_SIGNAL_COUNT_INTERNAL0　　CNT 内部サンプリングクロック[0x110]
SRC_SIGNAL_COUNT_COUNTUP0　　CNT アップカウント一致 [0x120]
SRC_SIGNAL_COUNT_BUSY0　　CNT ビジィ[0x131]
SRC_SIGNAL_ECU_GENERAL_COMMAND0　　汎用コマンド [0x180]

			ディスティネーションシグナル						
			AI			CNT			
			格納許可トリガ	不許可トリガ	サンプリングクロック	外部ステータス	開 始トリガ	停止トリガ	アップカウント
ソースシグナル			0x00	0x02	0x04	0x74	0x80	0x82	0x86
AI	Non Connection	0x000	◎	◎	◎	◎	◎	◎	◎
	内部サンプリングクロック	0x004			◎				
	ビフォートリガサンプリング回数終了	0x011		◎					
	外部開始エッジ	0x090	◎						
	外部停止エッジ	0x091		◎					
	外部クロック	0x092			◎				
CNT	外部アップカウントクロック	0x096							◎
	内部サンプリングクロック	0x110				◎			◎
	アップカウント一致	0x120				◎			◎
	ビジィ	0x131							
汎用コマンド		0x180	◎	◎			◎	◎	

int ai1616l_ai_abort_settings(int fd,unsigned long int value)
イベントコントローラ(0x38)に、AI 異常停止設定コマンド(0x10)を出力後、ECU
設定データポート(0x3C)に value を出力する。
value は、AI1616L_ECU_GENERAL0_COMMAND_AI_CLK_ERROR(0x00000001)が AI CLK
Error で、AI1616L_ECU_GENERAL0_COMMAND_AI_OVERFLOW(0x00010000)が AI Over Flow である。

int ai1616l_ai_setcommand(int fd,unsigned long int command,unsigned long int value)
アナログ入力機能設定コマンド
アナログ入力コマンドポート(0x30)に、command を(32bit)出力する。
パラメータを伴うコマンドの場合には、アナログ入力設定データポート(0x34)
に、value(32bit)を出力する。
command,value は以下の通り

command　　: value
AI1616L_AI_RESET_COMMAND　　: AI 初期化コマンド(0x10000000)
value=0 を設定
AI1616L_AI_GATE_OPEN_COMMAND　　: AI 内部ゲートオープンコマンド(0x10000001)
value=0 を設定
AI1616L_AI_ABORT_COMMAND　　: AI 強制停止コ マ ンド(0x10000002)
value=0 を設定

AI1616L_AI_INTCLK_COMMAND : AI 用 内 部 クロック設定(0x10000003)
value=内部クロックの パルス周期 = 周期(ms)/25 -1
AI1616L_AI_CHNUM_COMMAND : AI 格 納 チャネル数設定(0x10000005)
value=チャンネル数-1(最大 7)
AI1616L_AI_CHSEQ_COMMAND :AI チャ ネ ル 設定(0x1000000C)
value=AI1616L_AI_CHSEQ_COM-
MAND_MULTI または AI1616L_AI_CHSEQ_COMMAND_SINGLE
AI1616L_AI_BEFTRG_COMMAND : AI ビフォートリガサンプリング回数設定(0x10000009)
value = ビフォートリ ガサンプリング回数-1

パラメータを伴うコマンドの場合には、アナログ入力設定データポート(0x34)
に、value(32bit)を出力する。
int ai1616l_memory_setcommand(int fd,unsigned long int command,unsigned long int value)
メモリコマンドポート(0x30)に、command を(32bit)出力する。
command は以下の通り
command : value
AI1616L_MEMORY_RESET_COMMAND : MEM 初期化コマン ド(0x60000000)
value=0 を設定
AI1616L_MEMORY_AI_MEM_CLEAR_COMMAND : AI MEM クリ ア(0x60000007)
value=0 を設定
AI1616L_MEMORY_AI_MEM_COMPARETE_TYPE_COMMAND : AI 用 MEM 比較タイプ 設定(0x60000010)
value=0 を設定
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 、"AI 比較データ数"フラグを生成するためにあります。
A/D コンバータか らアナログ入力データが入力された場合に
"AI 比較データ数 ホールド"フラグがセットされます。
AI 機能のデータ入力ポートからアナログ入力データを読み込みによって、
データ数が減り、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。
このコマンドの次 に AI1616L_MEMORY_AI_MEM_COMPARETE_DATA_COMMAND(AI MEM 比較データ設 定)
が必要です。
リ セ ットは ECU の機能にあります。

int ai1616l_memory_getcommand(int fd,unsigned long int command,unsigned
long int *value) メモリコマンドポート(0x30)に、command を(32bit)出力し、
メモリ設定データポート(0x34)から、value(32bit)を入力する。
AI1616L_MEMORY_AI_MEM_FIFOCOUNT_COMMAND:AI FIFO カウンタ確認コマ ンド(0x6000000B)
value に AI FIFO カウンタの値
int ai1616l_memory_setcommand2(int fd,unsigned long int com-
mand,unsigned long int value1,unsigned long int value2)
メモリコマンドポート(0x30)に、command を(32bit)出力し、メモリ設定デー タ
ポート LO(0x34)に、value1(16bit)を出力後、メモリ設定データポート HI(0x36)
に、value2(16bit)を出力する。
command,value1,value2 は以下の通り
AI1616L_MEMORY_AI_MEM_COMPARETE_DATA_COMMAND:AI 用 MEM 比較データ設 定(0x60000011)
value1:比較番号
0:比較データ
上記以外無効
value2:設定データ=(比較
データ-1)
int ai1616l_di_setcommand2(int fd,int command,int value1,int value2);

デジタル入力コマンドポート(0x30)に、command を(32bit)出力し、デジタル 入力設定データポート LO(0x34)に、value1(16bit)を出力後、デジタル入力設定データポート HI(0x36)に、value2(16bit)を出力する。

command,value1,value2 は以下の通り

AI1616L_DI_RESET_COMMAND: DI 初期化コマンド(0x30000000)
value1=0,value2=0 を設定

AI1616L_DI_EDGE_DETECT_COMMAND: DI エッジ検出設定(0x30000001)
value1=以下の外部から入力 されるディジタル入力タイプを選択
AI1616L_DI_EDGE_TYPE_AI(0x1)
AI1616L_DI_EDGE_TYPE_CNT(0x3)
value2= ディジタル入力の以下 の検出ビットを
AI1616L_DI_EDGE_TYPE_NONE
AI1616L_DI_EDGE_TYPE_RISE
AI1616L_DI_EDGE_TYPE_FALL
AI1616L_DI_EDGE_TYPE_BOTH
下記のポート分シフトした値を ビットオアした値を設定
AI1616L_DI_EDGE_BIT00_SHIFT
AI1616L_DI_EDGE_BIT01_SHIFT
AI1616L_DI_EDGE_BIT02_SHIFT
例 え ば 、(AI1616L_DI_EDGE_TYPE_RISE<<AI1616L_DI_EDGE_BIT00_SHIFT)|...

AI1616L_DI_DIGITAL_FILTER_COMMAND: DI Digital Filter 設 定(0x30000002)
value1=以下のディジタルフ ィルタを入れる信号名を選択
AI1616L_DI_DIGITAL_FITER_DI DI Digital Filter
AI1616L_DI_DIGITAL_FILTER_AI AI Digital Filter
AI1616L_DI_DIGITAL_FILTER_CNT CNTDigital Filter
value2=以下のディジタルフ ィルタ時間を選択
AI1616L_DI_DIGITAL_FILTER_OFFDigital Filter なし
AI1616L_DI_DIGITAL_FILTER_1U Digital Filter 1 マイクロ秒

int ai1616l_do_setcommand(int fd,int command);
デジタル出力コマンドポート(0x30)に、AI1616L_DO_RESET_COMMAND を(32bit)出力する
command は以下の通り

AI1616L_DO_RESET_COMMAND DO 初期化コマンド (0x40000000)
本ボードは設定が固定になっています。設定の変更はできません。

int ai1616l_cnt_setcommand(int fd,int command,int value);
カ ウンタコマンドポート(0x30)に、command を(32bit)出力し、カウンタ設定データポート(0x34)に、value(32bit)を出力する。

AI1616L_COUNTER_RESET_COMMAND: CNT 初期化コマンド(0x50000000)
value=0 を設定

AI1616L_COUNTER_GATE_OPEN_COMMAND:CNT 内部ゲートオープ ン コ マ ン ド(0x50000001)
value=0 or 1:内部ゲートオープンするかしないかを設定

AI1616L_COUNTER_ABORT_COMMAND: CNT 強制停止コマンド(0x50000002)
value=0 or 1:強制停止するかしないかを設定

int ai1616l_get_counter(int fd,int *value);
カ ウンタコマンドポート(0x30) に 、AI1616L_COUNTER_INTCLK_COMMAND(0x50000003)を(32bit)出力し、カウンタ設定データポート LO(0x34)に、内部クロックの値を読み出し*value に設定する。
int ai1616l_set_counter(int fd,int ch,int value);
カ ウンタコマン ドポート(0x30) に 、AI1616L_COUNTER_INTCLK_COMMAND(0x50000003)を(32bit)出力し、カウンタ設定データポート LO(0x34)に、value の値を設定する。
int ai1616l_set_counter_comparete(int fd,int value);
カ ウンタコマン ド ポート(0x30) に 、AI1616L_COUNTER_COMPARETE_COMMAND(0x50000007)を(32bit)出力し、カウ ンタ 設定データポート(0x34)に、value(32bit)を出力し、比較データを設定する。
int ai1616l_ai_read(int fd,unsigned long int *data,int n);
AD 変換器から n データを読み出す 内部で、ai1616l_ai_start()を最初に呼び出す
注意:この関数で無限サンプリングを実現することはできません。
無限サンプリングを行うためのレジスタ設定例は ai1616l_samples/intr.c を 参考にしてください。

int ai1616l_ai_read_data(int fd,unsigned long int *data);
AD 変換器から1データだけ読み出す
int ai1616l_ai_read_csr(int fd,unsigned short int *data);
AD 変換器の CSR(0x04)から1データだけ読み出す
int ai1616l_ai_calibration(int fd,unsigned int value);
AI のキャリブレーションを行う
value:固定値のため以下の値を設定する
AI1616L_AI_CAL_RANGE_M10VtoP10V -10 to +10
返り値:-2:タイ ム ア ウ ト(5 秒)
-1:それ以外 errno
参照
int ai1616l_ai_start(int fd);
以下の手順で AD 変換をスタートする
(1) アナ ログ入力コマンドポート(0x30) に 、AI1616L_AI_GATE_OPEN_COMMAND(0x10000001)を(32bit)出力する。
(2) ア ナ ロ グ 入 カ ス テ ー タ ス セ ン ス ポ ート(0x04)をスキャンし、AI1616L_AI_START_DISABLE(0x00000010)ビットが真である間、フラグスキャ ンループを行う。
(3)ECU コマンドポート(0x38)に、AI1616L_ECU_GENERAL0_COMMAND(0x00000005) を出力し、AI をスタートする。
int ai1616l_di_setmask(int fd,int value);
DI のマスク(0x14)を設定する
int ai1616l_di_getvalue(int fd,int *value);
DI の値(0x10)を読み出す
int ai1616l_do_setmask(int fd,int value);
DO のマスク(0x1C)を設定する
int ai1616l_do_setvalue(int fd,int value);
DO(0x18)に値を設定する

SEE ALSO
/usr/local/CNC/drivers/extmem/contec/ai1616l 下のプログラム

AUTHORS