ＡＬＧＯ省配線システム

ユーザーズマニュアル

INtime
A-Link RSL

目　次

ALink RSLの概要

第１章 アプリケーション開発

第２章 RSL関数

ALink RSLの概要

本 RSL「ALink.RSL」は、アルゴシステム省配線システムである「A-Link」を用いたシステムの INtime アプリケーションを作成するユーザーが、容易に作成できる便宜をはかる為に提供されます。
ユーザーは、INtime 開発環境（Microsoft Visual Studio 2005/2008/2010）を使用して、アプリケーションから RSL 関数をコールすることによって A-Link 入出力を処理することができます。

本 RSL は、上位、下位の 2 層構造となっておりユーザーアプリケーションがアクセスする上位の「ALink.RSL」と、A-Link マスタである PCI ボード「PCILZ10-0」等をアクセスする下位の「AL36Mst.RSL」（※１）等から構成されております。ユーザーは下位 RSL を意識する必要はありません。
ユーザーは A-Link スレーブへのデータ入出力を、該当する関数をコールすることで簡単に行うことができます。

下位 RSL は使用するボードの種類によって変わります。使用するボードは、A-Link.ini ファイルで指定します。これにより、複数枚および複数種類の A-Link マスタボードをサポートができるようになります。
A-Link.ini ファイルについては、「ALink.ini 設定マニュアル」を参照して下さい。
下位 RSL「AL36Mst.RSL」、「G5ALMst.RSL」、「G4EALMst.RSL」、「G8ALMst.RSL」の設定は「AL36Mst.ini 設定マニュアル」、「G5ALMst.ini 設定マニュアル」、「G4EALMst.ini 設定マニュアル」、「G8ALMst.ini 設定マニュアル」を参照して下さい。

A-Link マスタボードの転送レートや全/半二重は、下位 RSL の ini ファイル(AL36Mst.ini、G5ALMst.ini、G4EALMst.ini、G8ALMst.ini 等)で設定します。各 ini ファイル設定マニュアルを参照して下さい。

産業用 PC/産業用パネル PC/オールインワンコントローラ用拡張ボード AP300EX、AP303EX を使用する場合、ALink.RSL を使用する前にマスタプロセスを起動する必要があります。(起動方法は、「INtime 省配線導入マニュアル」を参照してください。)

（※１）例として A-Link 用 PCI ボード用ドライバ、下位 RSL を使用しております。

第１章　アプリケーション開発

１－１　A-Link動作環境

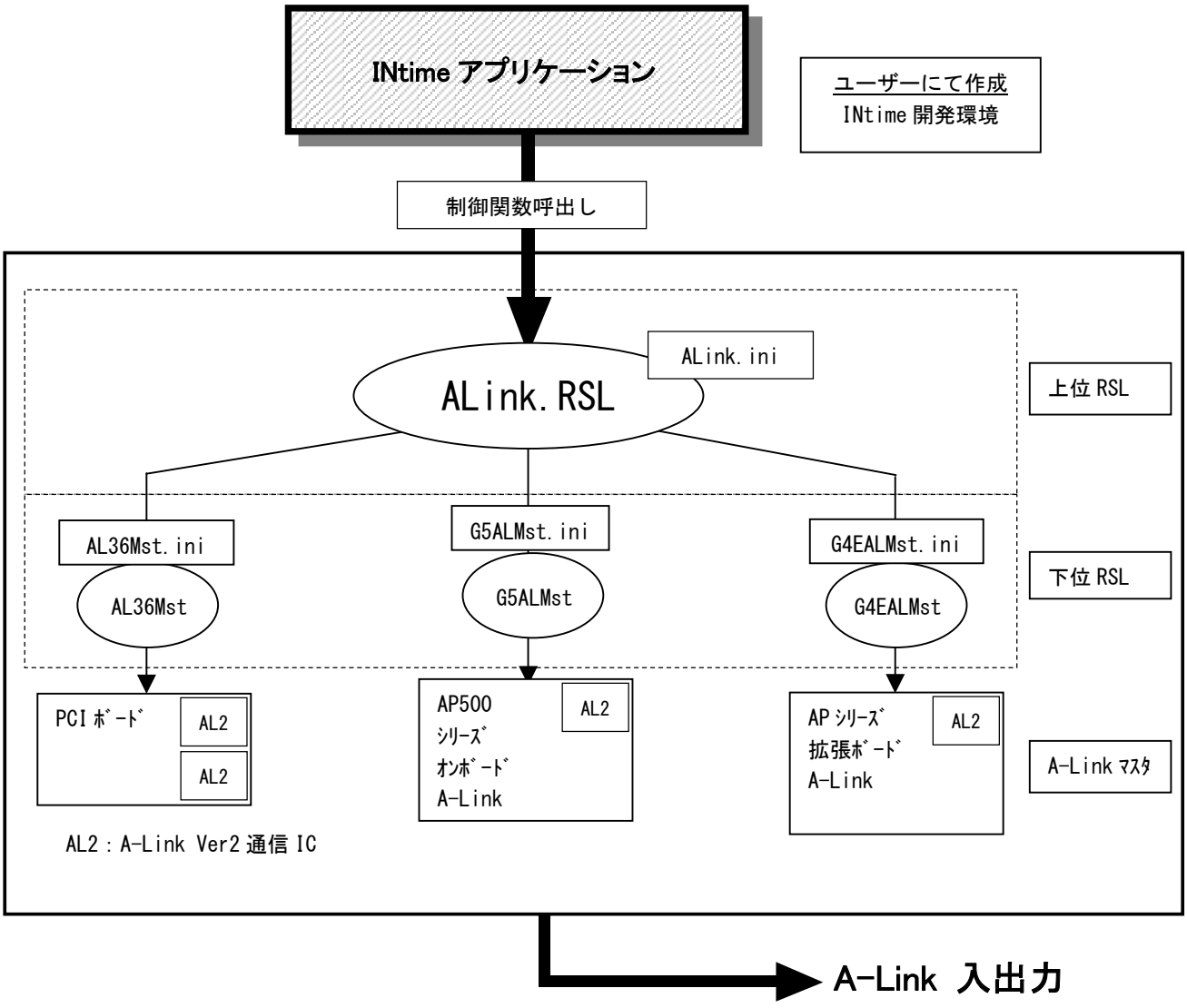
ユーザーは作成するアプリケーション内で ALink.RSL の関数をコールすることにより A-Link スレーブの入出力を処理します。

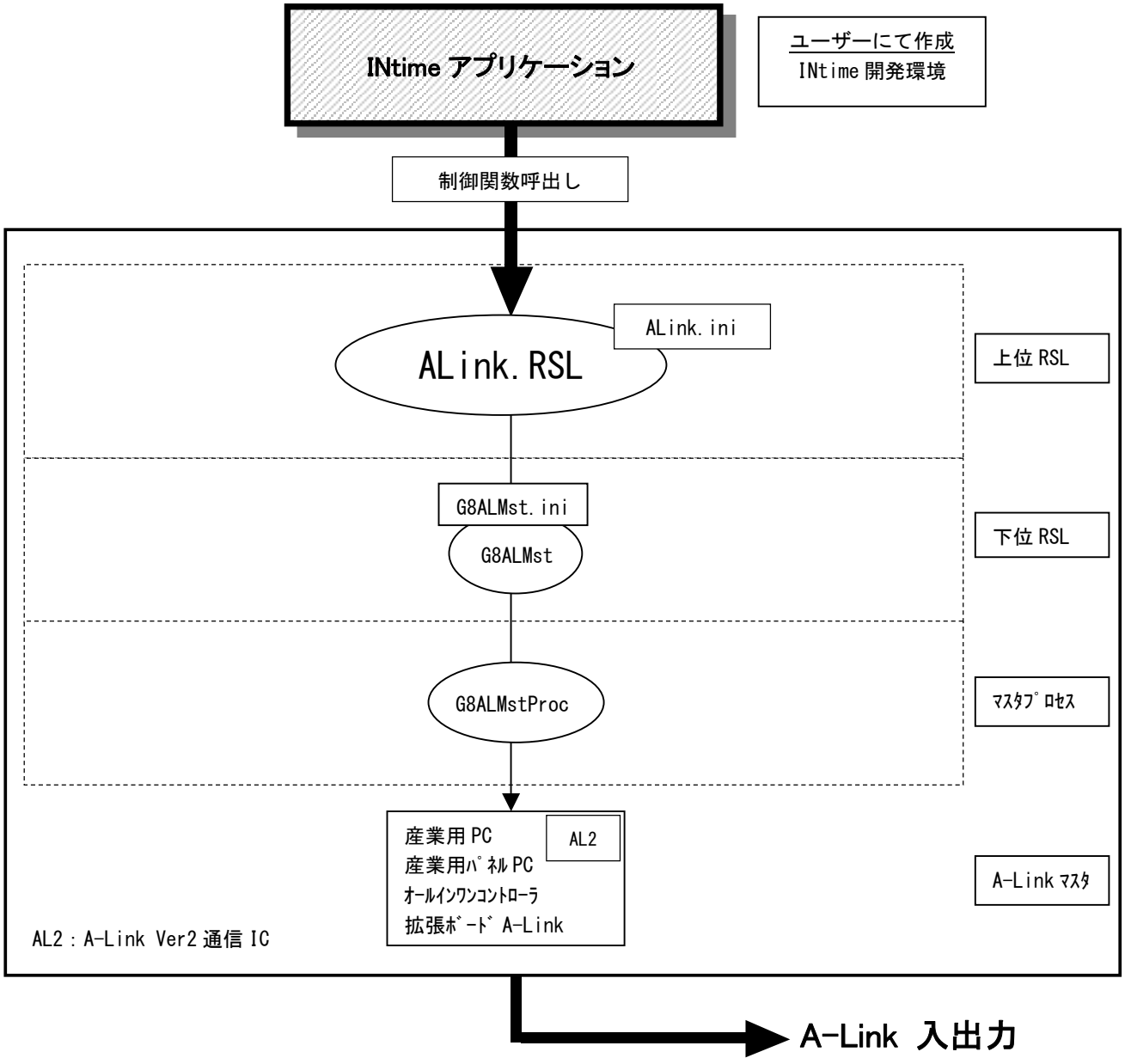
ALink.RSL は起動時にシステムの情報として ALink.ini ファイルを参照しますので､A-Link.RSL を使用したアプリケーションを動作させる前に ALink.ini を作成する必要があります。

作成したアプリケーション A-Link.ini、ALink.RSL（＋使用する下位 RSL）は同一フォルダ（ディレクトリ）に格納してアプリケーションを動作させます。

ALink.RSL は、A-Link マスタにアクセスするために下位 RSL をロードします。下位 RSL は A-Link マスタの種類毎に存在します。ロードする下位 RSL の指定も ALink.ini ファイルで行います。

＊　AP300EX/AP303EX を使用する場合、ALink.RSL を使用する前にマスタプロセスを起動する必要があります。（起動方法は、「INtime 省配線導入マニュアル」を参照してください。）




INtime アプリケーション
ユーザーにて作成
INtime 開発環境
制御関数呼出し
ALink.ini
ALink.RSL
上位 RSL
G8ALMst.ini
G8ALMst
下位 RSL
G8ALMstProc
ﾏｽﾀﾌﾟﾛｾｽ
産業用 PC
産業用ﾊﾟﾈﾙ PC
ｵｰﾙｲﾝﾜﾝｺﾝﾄﾛｰﾗ
拡張ﾎﾞｰﾄﾞ A-Link
AL2
A-Link ﾏｽﾀ
AL2：A-Link Ver2 通信 IC
A-Link 入出力

アプリケーション開発の準備

開発アプリケーションから RSL をコールできるようにする為に、開発ユーザーは、下記の手順を実行します。

1）Microsoft Visual C++

① プロジェクトのソースファイルがあるフォルダに、AlgALink.cpp、AlgALink.h、A-LinkDef.h をコピーします。

② RSL の関数をコールするソースファイルへ、AlgALink.h をインクルードします。

③ プロジェクトへ AlgALink.cpp を追加します。

④ プログラム起動時に、次の関数をコールして RSL をロードして下さい。

```
LoadALinkRSL("ALink.rsl");
```

⑤ プログラム終了時に、次の関数をコールして RSL をアンロードして下さい。

```
UnloadALinkRSL();
```

＊ 上記で使用されるヘッダファイル等は「開発環境 CD-ROM」に含まれています。また、開発環境 CD-ROM にはサンプルソースなども含まれますので合わせてご利用下さい。

第２章　RSL関数

２－１　RSL関数概要

A-Link の RSL 関数は、各ユニットタイプ（別記号）毎に対応する制御関数とユニット制御を補助する関数が用意されています。

関数形式は、AL_[タイプ別記号]_[コマンド](引数 1, 引数 2, ・・・）になります。

ユニットタイプの詳細は「ALink.ini 設定マニュアル」を、各関数の詳細は「A-Link RSL リファレンスマニュアル」を参照して下さい。ここでは、一般的な RSL 関数について説明します。

１）ユニット制御関数

ユニット制御関数の関数名には、ユニットのタイプ別記号が用いられています。 複数個の同じユニットを区別する為に、関数の引数としてロジカル ID（LID）を渡す必要があります。 システム全体を通して、同じタイプのユニットは、異なるロジカル ID をもつ必要があります。 そのロジカル ID のユニットが実際にはどこに接続され、スレーブアドレスが何番であるかは、設定ファイル（ALink.ini）で指定します。（詳細は「ALink.ini 設定マニュアル」を参照して下さい。）

RSL 関数を使用するには、先ず各ユニット毎に Open 関数をコールする必要があります。正常リターンを確認してから、データ入出力関数をコールするようにして下さい。また、そのユニットのアクセスが不要になれば、Close 関数をコールするようにして下さい。

・タイプ別記号 DIO　　デジタル入出力ユニット関数

入出力として ON、OFF の 2 状態を持つ接点を扱うユニットのための関数

AL_DIO_Open(Lid)	ﾕﾆｯﾄをｵｰﾌﾟﾝします
AL_DIO_Close(Lid)	ﾕﾆｯﾄをｸﾛｰｽﾞ します
AL_DIO_InData(Lid, *InDat)	ﾕﾆｯﾄの 16ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_DIO_InDataBit(Lid, Bit, * InBit)	ﾕﾆｯﾄの指定したﾋﾞｯﾄからﾃﾞｰﾀを取得します
AL_DIO_OutData(Lid, OutDat)	ﾕﾆｯﾄに 16ﾋﾞｯﾄﾃﾞｰﾀを出力します
AL_DIO_OutDataBit(Lid, OutBit)	ﾕﾆｯﾄの指定したﾋﾞｯﾄにﾃﾞｰﾀを出力します
AL_DIO_InODdata(Lid, *InDat)	ﾕﾆｯﾄに出力したﾃﾞｰﾀを取得します
AL_DIO_InODataBit(Lid, Bit, *InBit)	ﾕﾆｯﾄに出力したﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_DIO_SetRenewData( )	DI ﾃﾞｰﾀの変化検知ｴﾘｱを 16ﾋﾞｯﾄ設定します
AL_DIO_SetRenewBit( )	DI ﾃﾞｰﾀの変化検知ｴﾘｱをﾋﾞｯﾄ設定します
AL_DIO_GetRenewData( )	DI ﾃﾞｰﾀの変化検知ｴﾘｱの 16ﾋﾞｯﾄ設定を取得します
AL_DIO_GetRenewBit( )	DI ﾃﾞｰﾀの変化検知ｴﾘｱのﾋﾞｯﾄ設定を取得します
AL_DIO_GetRenewStat( )	DI ﾃﾞｰﾀの変化検知ｴﾘｱの変化を取得します
AL_DIO_GetCondition(Lid, *Cond)	ﾕﾆｯﾄの通信状態を取得します

・タイプ別記号 ADA　　アナログ入出力ユニット関数

入出力としてアナログ値を扱うユニットのための関数

AL_ADA_Open(Lid)	ﾕﾆｯﾄをｵｰﾌﾟﾝします
AL_ADA_Close(Lid)	ﾕﾆｯﾄをｸﾛｰｽﾞします
AL_ADA_InValue(Lid, Ch, *InVal)	ﾕﾆｯﾄから 12ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_ADA_InValueFull(Lid, Ch, *InDat)	ﾕﾆｯﾄから 16ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_ADA_OutValue(Lid, Ch, OutVal)	ﾕﾆｯﾄに 12ﾋﾞｯﾄﾃﾞｰﾀを出力します
AL_ADA_OutValueFull(Lid, Ch, OutDat)	ﾕﾆｯﾄに 16ﾋﾞｯﾄﾃﾞｰﾀを出力します
AL_ADA_InOValue(Lid, Ch, *InVal)	ﾕﾆｯﾄの出力 12ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_ADA_InOValueFull(Lid, Ch, *InDat)	ﾕﾆｯﾄの出力 16ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_ADA_SetFilter(Lid, Ch, Filter, ...)	ﾕﾆｯﾄのﾌｨﾙﾀ設定を行います
AL_ADA_GetFilter(Lid, Ch, *Filter, ...)	ﾕﾆｯﾄのﾌｨﾙﾀ設定を取得します
AL_ADA_GetInRange(Lid, Ch. *Range)	ﾕﾆｯﾄの入力ﾚﾝｼﾞ設定を取得します
AL_ADA_GetOutRange(Lid, Ch, *Range)	ﾕﾆｯﾄの出力ﾚﾝｼﾞ設定を取得します
AL_ADA_GetCondition(Lid, *Cond)	ﾕﾆｯﾄの通信状態を取得します

・タイプ別記号 AXSA　　位置決めユニット 関数

モータの制御を扱うユニットのための関数

AL_AXSA_Open(Lid)	ﾕﾆｯﾄをｵｰﾌﾟﾝします
AL_AXSA_Close(Lid)	ﾕﾆｯﾄをｸﾛｰｽﾞします
AL_AXSA_PutCmd(Lid, JNo, CmdAll, ...)	ﾕﾆｯﾄへの各種ｺﾏﾝﾄﾞをｾｯﾄします
AL_AXSA_Answer(Lid, JNo, *Status, ...)	ｺﾏﾝﾄﾞの応答(ｽﾃｰﾀｽ含)を取得します
AL_AXSA_AnsWith (Lid, JNo, *Status, ...)	ｺﾏﾝﾄﾞの応答(ｽﾃｰﾀｽ含)と現在位置(拡張)ﾃﾞｰﾀを取得します
AL_AXSA_GetStatus(Lid, JNo, ...)	常時ｱｯﾌﾟﾛｰﾄﾞ時にｽﾃｰﾀｽと現在位置ﾃﾞｰﾀを取得します
AL_AXSA_GetCondition(Lid, *Cond)	ﾕﾆｯﾄの通信状態を取得します

・タイプ別記号 ENC　　エンコーダ・カウンタユニット関数

エンコーダ・カウンタ値を扱うユニットのための関数

AL_ENC_Open(Lid)	ﾕﾆｯﾄをｵｰﾌﾟﾝします
AL_ENC_Close(Lid)	ﾕﾆｯﾄをｸﾛｰｽﾞします
AL_ENC_PuCmd(Lid, Ch, Cmd, Param)	ﾕﾆｯﾄへの各種ｺﾏﾝﾄﾞをｾｯﾄします
AL_ENC_GetCondition(Lid, *Cond)	ﾕﾆｯﾄの通信状態を取得します

・タイプ別記号 SIO　　シリアルユニット関数

シリアル通信データを扱うユニットのための関数

AL_SIO_Open(Lid)	ﾕﾆｯﾄをｵｰﾌﾟﾝします
AL_SIO_Close(Lid)	ﾕﾆｯﾄをｸﾛｰｽﾞします
AL_SIO_Config(Lid, Ch, Cmd, Param)	ｼﾘｱﾙ通信の通信設定を行います
AL_SIO_PutData(Lid, Ch, *Buffer, …)	ﾃﾞｰﾀを送信します
AL_SIO_GetData(Lid, Ch, *Buffer, …)	ﾃﾞｰﾀを受信します
AL_SIO_ClearError(Lid, Ch)	通信ｴﾗｰをｸﾘｱします
AL_SIO_CheckBuffer(Lid, Ch, *Size)	受信ﾊﾞｯﾌｧに取得されているﾃﾞｰﾀ数を取得します
AL_SIO_ClearBuffer(Lid, Ch)	受信ﾊﾞｯﾌｧをｸﾘｱします
AL_SIO_GetCondition(Lid, *Cond)	ﾕﾆｯﾄの通信状態を取得します

・タイプ別記号 ADAC　　ちび丸君アナログ入出力ユニット関数

ちび丸君シリーズのアナログ入出力を扱うユニットのための関数

AL_ADAC_Open(Lid)	ﾕﾆｯﾄをｵｰﾌﾟﾝします
AL_ADAC_Close(Lid)	ﾕﾆｯﾄをｸﾛｰｽﾞします
AL_ADAC_InValue(Lid, Ch, *InVal)	ﾕﾆｯﾄから 12ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_ADAC_OutValue(Lid, Ch, OutVal)	ﾕﾆｯﾄに 12ﾋﾞｯﾄﾃﾞｰﾀを出力します
AL_ADAC_SetFilter(Lid, Ch, Filter)	ﾕﾆｯﾄのｱﾅﾛｸﾞ入力のﾌｨﾙﾀ設定を行います
AL_ADAC_GetCondition(Lid, *Cond)	ﾕﾆｯﾄの通信状態を取得します

・タイプ別記号 ADAD　　ALD アナログ入出力ユニット関数

入出力としてアナログ入出力を扱うユニットのための関数

AL_ADAD_Open(Lid)	ﾕﾆｯﾄをｵｰﾌﾟﾝします
AL_ADAD_Close(Lid)	ﾕﾆｯﾄをｸﾛｰｽﾞします
AL_ADAD_InValue(Lid, Ch, *InVal, Mode)	ﾕﾆｯﾄから 13ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_ADAD_InValueFull(Lid, Ch, *InDat)	ﾕﾆｯﾄから 16ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_ADAD_OutValue(Lid, Ch, OutVal, Mode)	ﾕﾆｯﾄに 13ﾋﾞｯﾄﾃﾞｰﾀを出力します
AL_ADAD_OutValueFull(Lid, Ch, OutDat)	ﾕﾆｯﾄに 16ﾋﾞｯﾄﾃﾞｰﾀを出力します
AL_ADAD_InOValue(Lid, Ch, *InVal)	ﾕﾆｯﾄの出力 13ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_ADAD_InOValueFull(Lid, Ch, *InDat)	ﾕﾆｯﾄの出力 16ﾋﾞｯﾄﾃﾞｰﾀを取得します
AL_ADAD_SetFilter(Lid, Ch, Filter, ...)	ﾕﾆｯﾄのﾌｨﾙﾀ設定を行います
AL_ADAD_GetFilter(Lid, Ch, *Filter, ...)	ﾕﾆｯﾄのﾌｨﾙﾀ設定を取得します
AL_ADAD_GetCondition(Lid, *Cond, Mode)	ﾕﾆｯﾄの通信状態を取得します
AL_ADAD_GetVersion(Lid, *Version)	ﾕﾆｯﾄのﾊﾞｰｼﾞｮﾝを取得します

２）制御補助関数

制御補助関数はユニット制御関数と共に用いることにより制御を補助します。

・PCI ボード状態の取得

PCI ボードの状態を調べるための関数

AL_CheckBoard(Board)	PCI ﾎﾞｰﾄﾞのｵｰﾌﾟﾝ状態を調べます。
AL_GetErrorCount(Board, Line, …)	PCI ﾎﾞｰﾄﾞのｴﾗｰを取得します。

２－２　下位RSL

A-Link RSL を正しく動作させるには A-Link マスタに対応した下位 RSL が必要となります。A-Link.RSL は、この下位RSL をリンクすることにより目的のボードを制御します。A-Link マスタに対応するドライバ、下位 RSL は以下のようになります。

ﾎﾞｰﾄﾞ	A-Link[Ver2] PCI ﾎﾞｰﾄﾞ	AP500 ｼﾘｰｽﾞ ｵﾝﾎﾞｰﾄﾞ A-Link	AP ｼﾘｰｽﾞ A-Link 拡張ﾎﾞｰﾄﾞ	産業用 PC 産業用ﾊﾟﾈﾙ PC ｵｰﾙｲﾝﾜﾝｺﾝﾄﾛｰﾗ ｼﾘｰｽﾞ 拡張ﾎﾞｰﾄﾞ A-Link
型式	PCILZ10-0 PCILZ11-0 PCILZ12-0 PCILZ13-0	---	AP140EX-0	AP300EX AP303EX
下位 RSL	AL36Mst.RSL	G5ALMst.RSL	G4EALMst.RSL	G8ALMst.RSL

このマニュアルについて

（1）本書の内容の一部または全部を当社からの事前の承諾を得ることなく、無断で複写、複製、掲載することは固くお断りします。
（2）本書の内容に関しては、製品改良のためお断りなく、仕様などを変更することがありますのでご了承下さい。
（3）本書の内容に関しては万全を期しておりますが、万一ご不審な点や誤りなどお気付きのことがございましたらお手数ですが巻末記載の弊社もしくは、営業所までご連絡下さい。その際、巻末記載の書籍番号も併せてお知らせ下さい。

７７ＩＴ１０００２Ｅ　２０１４年　１０月　第５版
７７ＩＴ１０００２Ａ　２０１１年　７月　初版

株式会社アルゴシステム

本社
〒５８７―００２１　大阪府堺市美原区小平尾６５６番地
ＴＥＬ（０７２）３６２－５０６７
ＦＡＸ（０７２）３６２－４８５６

ホームページ　http://www.algosystem.co.jp