ユーザーズマニュアル

Ａ－Ｌｉｎｋ　ＤＬＬ

目　次

ALink DLL の概要

本 DLL「ALink.DLL」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 Windows 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、Microsoft Visual C++、Borland C++Builder , Borland Delphi 等の開発言語から、DLL 関数をコールすることによって A-Link 入出力を処理するアプリケーションを作成することができます。

本 DLL は、上位と下位との複数から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.DLL」と A-Link マスタである PCI ボード「PCILZOO-O」等をアクセスする為のデバイスドライバ「Alink2Drv.SYS」(※１）を内部でコールする下位の「ALMstO2.DLL」(※１）等から構成されております。
ユーザーは下位 DLL およびデバイスドライバを意識する必要はありません。
（下位 DLL は使用するボードの種類によって変わり、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。また、下位 DLL「AL36Mst.DLL」、「ALPcc.DLL」、「AL36Pcc.DLL」、「MALMst.DLL」、「G4EALMst.DLL」、「G5ALMst.DLL」、「G8ALMst.DLL」、「G8ALMst64.DLL」、「AL36eMst.DLL」の設定は「AL36Mst.ini 設定マニュアル」、「ALPcc.ini 設定マニュアル」、「AL36Pcc.ini 設定マニュアル」、「MALMst.ini 設定マニュアル」、「G4EALMst.ini 設定マニュアル」、「G5ALMst.ini 設定マニュアル」、「G8ALMst.ini 設定マニュアル」、「G8ALMst64.ini 設定マニュアル」、「AL36eMst.ini 設定マニュアル」を参照して下さい。）

ユーザーは A-Link スレーブへのデータの入出力を､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。

本 DLL を使用する前に、ユーザーは、A-Link マスタボードを実行マシンの PCI スロット、PC カードスロットに挿入し、添付のインストール CD-ROM でインストール作業を行っておく必要があります。テストツールもインストールされますので実際に A-Link スレーブを接続して、テストを行って下さい。

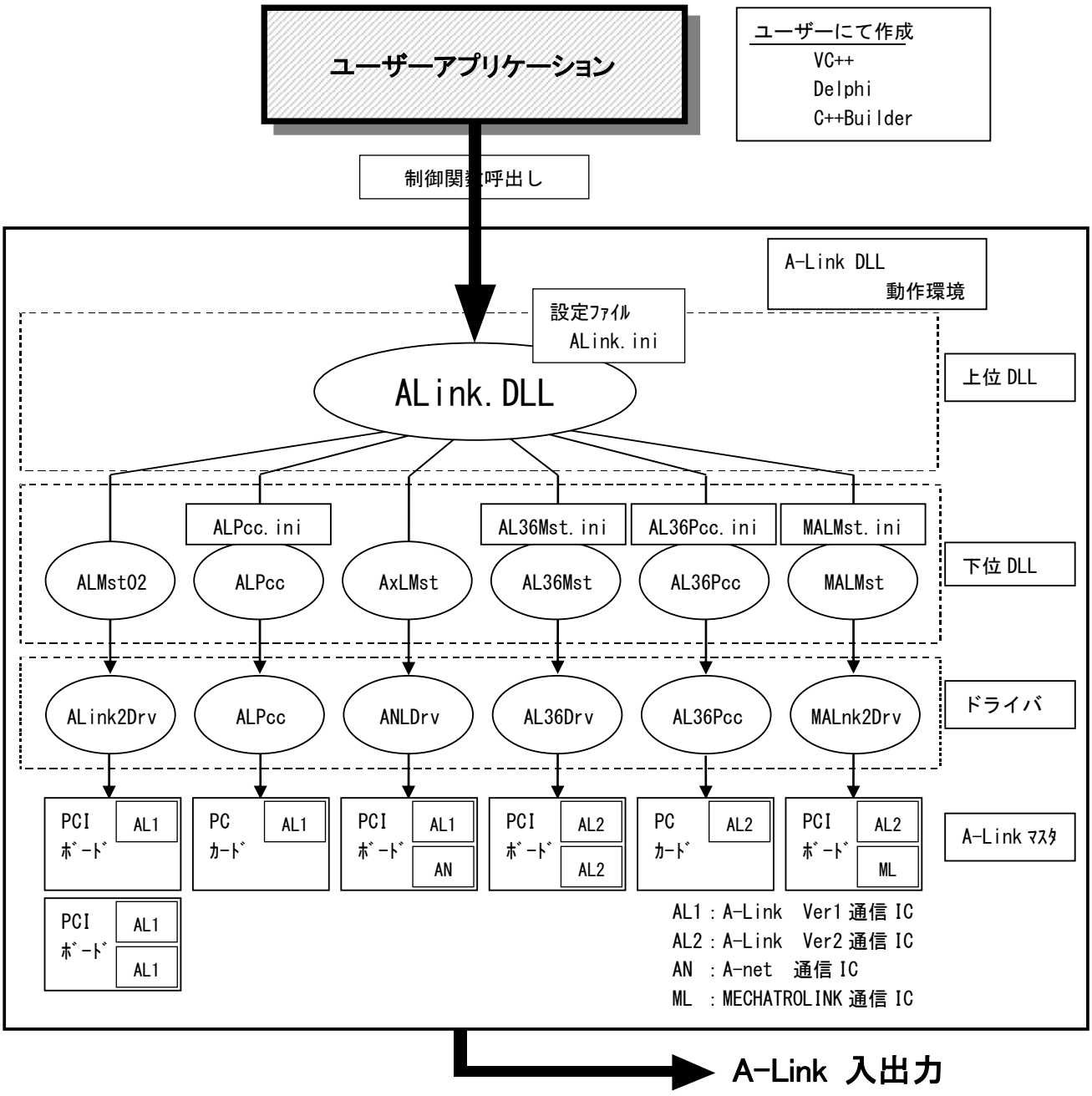
A-Link マスタボードの転送レートや全／半二重は、ハード上のディップスイッチでの設定となりますので、ハードウェアマニュアルを参照して下さい。ただし、PC カードや A-Link Ver2 通信 IC 搭載 PCI ボードの転送レートや全／半二重の設定は、下位 DLL の ini ファイル(AL36Mst.ini、ALPcc.ini 等)設定となりますので、各 ini ファイル設定マニュアルを参照して下さい。

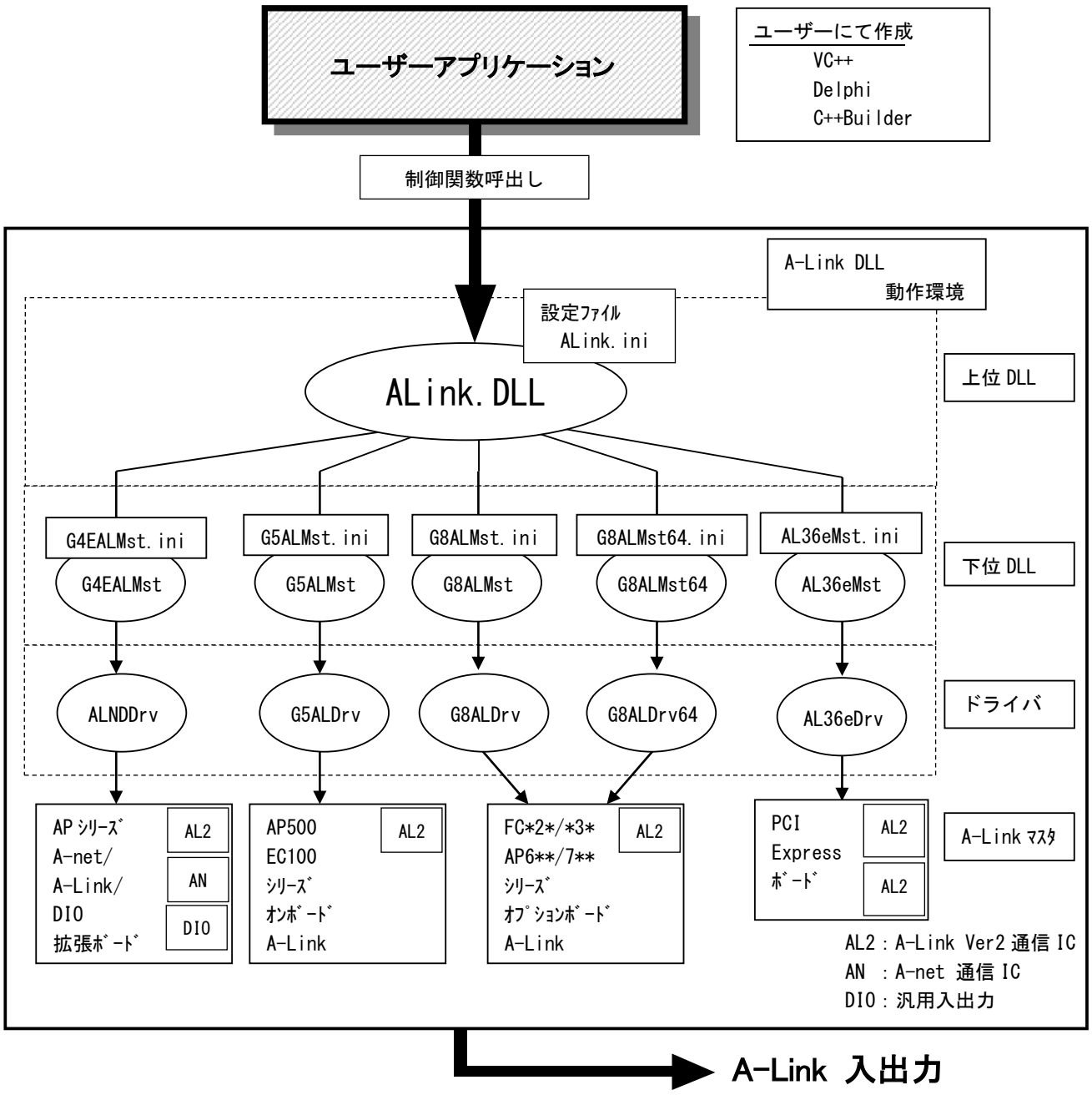
（※１）例として A-Link 用 PCI ボード用ドライバ、下位 DLL を使用しております。

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.DLL 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.DLL 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.DLL 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.DLL（＋使用する下位 DLL）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。




ユーザーアプリケーション
ユーザーにて作成
VC++
Delphi
C++Builder
制御関数呼出し
A-Link DLL
動作環境
設定ﾌｧｲﾙ
ALink.ini
ALink.DLL
上位 DLL
G4EALMst.ini
G5ALMst.ini
G8ALMst.ini
G8ALMst64.ini
AL36eMst.ini
G4EALMst
G5ALMst
G8ALMst
G8ALMst64
AL36eMst
下位 DLL
ALNDDrv
G5ALDrv
G8ALDrv
G8ALDrv64
AL36eDrv
ドライバ
APｼﾘｰｽﾞ
A-net/
A-Link/
DIO
拡張ﾎﾞｰﾄﾞ
AL2
AN
DIO
AP500
EC100
ｼﾘｰｽﾞ
ｵﾝﾎﾞｰﾄﾞ
A-Link
AL2
FC*2*/*3*
AP6**/7**
ｼﾘｰｽﾞ
ｵﾌﾟｼｮﾝﾎﾞｰﾄﾞ
A-Link
AL2
PCI
Express
ﾎﾞｰﾄﾞ
AL2
AL2
A-Link ﾏｽﾀ
AL2 : A-Link Ver2 通信 IC
AN : A-net 通信 IC
DIO : 汎用入出力
A-Link 入出力

１－２　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 DLL 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++
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。
DLL 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。
プロジェクトへ AlgALink.CPP を追加します。
プログラム起動時に走る部分で、次の関数をコールして下さい。LoadALinkDll(“ALink.DLL”);
プログラム終了時に走る部分で、次の関数をコールして下さい。UnLoadALinkDll();

上記以外に C++のコンパイル設定で「プリコンパイルヘッダを使用しない」を指定して下さい。但し、「プリコンパイルヘッダを使用する」を指定する場合（ヘッダ指定が　stdafx.h　の時、つまりスイッチが /Yu"stdafx.h" の時）は、AlgALink.cpp の上部 ヘッダ指定に次の１行追加して下さい。

```
例：　　#include <windows.h>
　　#include "stdafx.h"　　＜－－追加行
　　#include "AlgALink.h"
```

２）Borland C++Builder
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.HA-LinkDef.H をコピーします。
DLL 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。
プロジェクトへ、AlgALink.CPP を追加します。
プログラム起動時に走る部分で、次の関数をコールして下さい。LoadALinkDll(“ALink.DLL”);
プログラム終了時に走る部分で、次の関数をコールして下さい。UnLoadALinkDll();

３）Borland Delphi
プロジェクトのソースファイルがあるフォルダに、ALinkLib.Pas と ALinkConst.Pas をコピーします。
プロジェクトへ A-LinkLib.Pas と ALinkConst.Pas を追加します。
DLL の関数をコールするソースファイルの Use 節に A-LinkLib.Pas と ALinkConst.Pas を追加して下さい。

＊　上記で使用されるヘッダファイル等は「A-Link 開発基本ソフト」をインストールすることによって得ることができます。また、インストーラ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　DLL 関数

２－１　DLL 関数概要

A-Link の DLL 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 DLL 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 DLL 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。）

DLL 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(*1)
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(*1)
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(*1)
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(*1)
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(*1)
AL_DIO_GetCondition(Lid, *Cond)	ﾕﾆｯﾄの通信状態を取得します

（＊１）A-Link　Ver2 通信 IC 搭載ボード使用時のみサポート（PCILZ10-0～PCILZ13-0、CRDLZ02-0）

・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。

・Windows システムタイマ精度変更関数

これらの関数によって Windows システムタイマ精度を変更することによって、制御を高速なものとすることができます。
位置決めユニット制御関数においてはハンドシェイクの時間が短縮され、より速い処理が可能となります。
システムタイマの変更は Windows の全体に及び、他のアプリケーション上でもタイマの精度が変更されるため注意が必要です。
（この関数を使用しない WinNT デフォルトのシステムタイマの精度は 10[ms]です）

AL_BeginPeriod( )	Windows ｼｽﾃﾑﾀｲﾏの精度を最小の 1[ms]とします。
AL_EndPeriod( )	Windows ｼｽﾃﾑﾀｲﾏの精度を標準のものに戻します。

・PCI ボード状態の取得

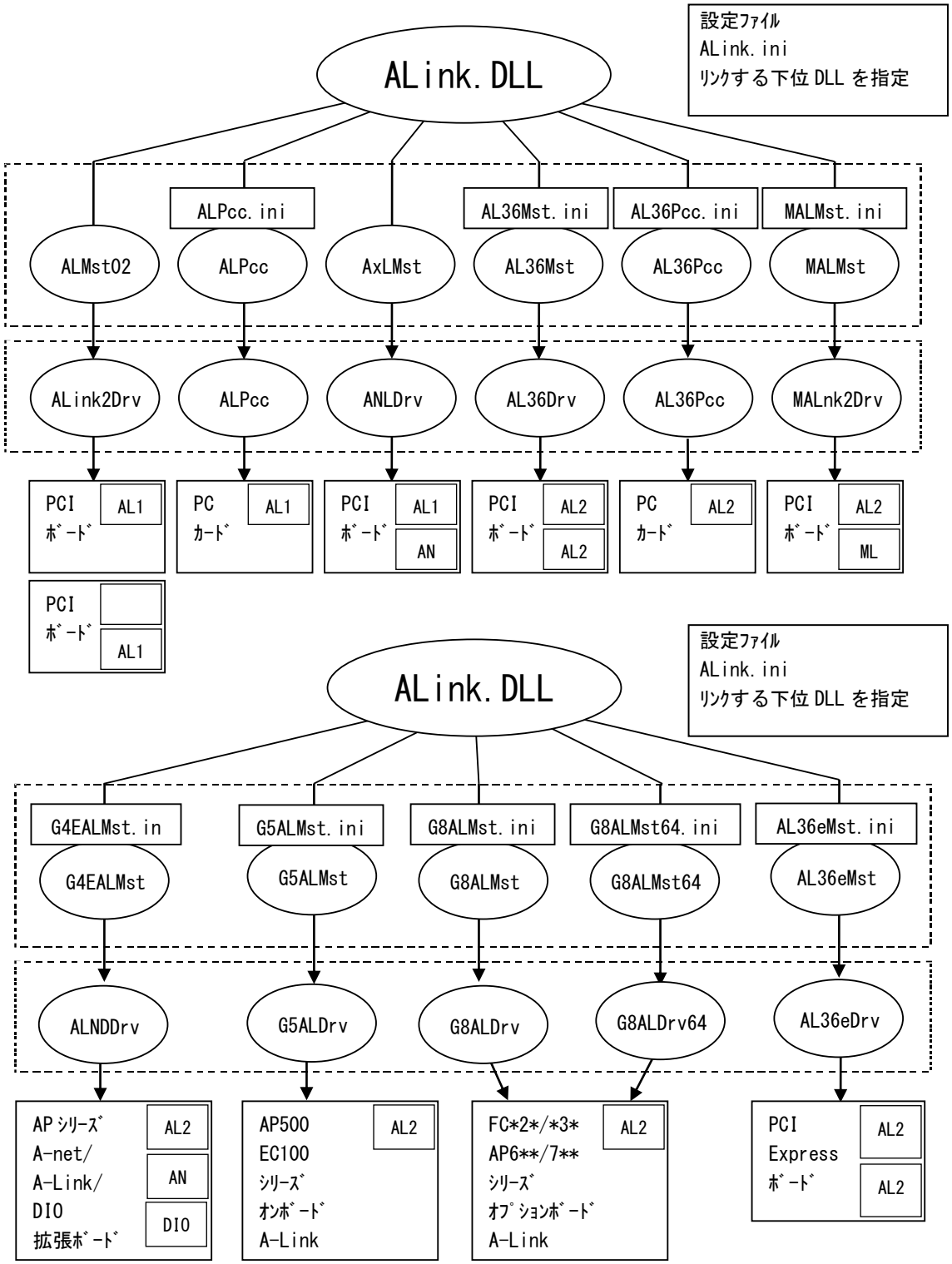
PCI ボードの状態を調べるための関数

AL_CheckBoard(Board)	PCI ﾎﾞｰﾄﾞのｵｰﾌﾟﾝ状態を調べます。
AL_CheckStatusLED(Board, Line, *Status)	PCI ﾎﾞｰﾄﾞの LED の状態を調べます。
AL_ClearStatusLED(Board, Line)	PCI ﾎﾞｰﾄﾞの LED を消灯させます。
AL_GetErrorCount(Board, Line, …)	PCI ﾎﾞｰﾄﾞのｴﾗｰを取得します。(*1)

（＊１）A-Link　Ver2 通信 IC 搭載ボード使用時のみサポート（PCILZ10-0～PCILZ13-0、CRDLZ02-0）

２－２　ドライバ・下位 DLL

A-Link DLL を正しく動作させるには A-Link マスタに対応したドライバが必要となります。
各々のドライバには対応した下位 DLL が存在します A-Link.DLL はこの下位 DLL 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 DLL は以下のようになります。



ﾎﾞｰﾄﾞ	A-Link[Ver1] PCI ﾎﾞｰﾄﾞ	A-Link[Ver1] PC ｶｰﾄﾞ	A-net/ A-Link[Ver1] PCI ﾎﾞｰﾄﾞ	A-Link[Ver2] PCI ﾎﾞｰﾄﾞ	A-Link[Ver2] PC ｶｰﾄﾞ	MECHATROLINK A-Link[Ver2] PCI ﾎﾞｰﾄﾞ
型式	PCILZ00-0 PCILZ01-0 PCILZ02-0 PCILZ03-0	CRDLZ00-0 CRDLZ01-0	PCISZ02-0	PCILZ10-0 PCILZ11-0 PCILZ12-0 PCILZ13-0	CRDLZ02-0	PCILM00-0
Vender ID	134FH	---	134FH	134FH	---	
Device ID	0002H	---	9903H	0302H	---	
ﾄﾞﾗｲﾊﾞ	ALink2Drv.sys	ALPcc.sys	ANLDrv.sys	AL36Drv.sys	AL36Pcc.sys	MALnkdrv.sys
下位 DLL	ALMst02.DLL	ALPcc.DLL	AxNMst.DLL	AL36Mst.DLL	AL36Pcc.DLL	MALMst.DLL

ﾎﾞｰﾄﾞ	AP ｼﾘｰｽﾞ A-net/A-Link/ DIO 拡張ﾎﾞｰﾄﾞ	AP500 EC100 ｼﾘｰｽﾞ ｵﾝﾎﾞｰﾄﾞ A-Link	FC*2*/*3*, AP6**/7** ｼﾘｰｽﾞ ｵﾌﾟｼｮﾝﾎﾞｰﾄﾞ A-Link	A-Link[Ver2] PCI Express ﾎﾞｰﾄﾞ
型式	---	---	---	PCIE-ALM01 PCIE-ALM02
Vender ID	134FH	134FH	134FH	134FH
Device ID	0901H	1001H	120AH	2102H 2103H
ﾄﾞﾗｲﾊﾞ	ANLDDrv.sys	G5ALDrv.sys	G8ALDrv.sys (Win7 32bit) G8ALDrv64.sys (Win7 64bit)	AL36eDrv.sys (Win10 64bit)
下位 DLL	G4EALMst.DLL	G5ALMst.DLL	G8ALMst.DLL (Win7 32bit) G8ALMst64.DLL (Win7 64bit)	AL36eMst.DLL (Win10 64bit)

第３章　付録

３－１　サンプルソース

１）Delphi 用　デジタル入出力ユニットサンプル
DLL とのリンク部分とユニット制御関連のオープン部とユニットの入出力、ビット単位での入出力、エラーメッセージのサンプルを次に示します。
（このソースはセットアップでインストールされる Delphi のサンプルから抜粋しています）
（DLL とのリンク宣言等はサンプルプログラムの A-LinkLib.pas にて行っています）

```
//-----------------------------------------------------
//      デジタル入出力ユニットオープン
//-----------------------------------------------------
procedure TMainForm.ButtonOpenClick(Sender: TObject);
var
  lid       : Integer;
  ret       : Integer;
begin
  lid := UpDownLid.Position;                              // ユーザ設定 ロジカル ID

  ret := AL_DIO_Open(lid);                                //オープン

  if (ret = AL_ER_OK) or (fOpen[lid] = True) then
  begin
        fOpen[lid] := True;    //  正常
        PanelOpen.Color := clGreen;
  end else
  begin
        fOpen[lid] := False;
        PanelOpen.Color := clBtnFace;
  end;

  ErrDsp(ret, 'AL_DIO_Open');
end;

//-----------------------------------------------------------------------
//      デジタル入出力ユニット　クローズ
//-----------------------------------------------------------------------
procedure TMainForm.ButtonCloseClick(Sender: TObject);
var
  lid       : Integer;
  ret       : Integer;
begin
  lid := UpDownLid.Position;

  ret := AL_DIO_Close(lid);                               //クローズ

  if (ret = AL_ER_OK) or (fOpen[lid] = False) then
  begin
```

```
        fOpen[lid] := False;
        PanelOpen.Color := clBtnFace;
  end else
  begin
        fOpen[lid] := True;                //正常
        PanelOpen.Color := clGreen;
  end;

  ErrDsp(ret, 'AL_DIO_Close');
end;

//---------------------------------------------------------------
//      ユニット　ビット単位　入力
//---------------------------------------------------------------
procedure TMainForm.ButtonInDataBitClick(Sender: TObject);
var
  lid           : Integer;
  bit_no        : Integer;
  inbit         : Integer;
  ret           : Integer;
begin
  lid := UpDownLid.Position;
  bit_no := ComboBoxBitNum.ItemIndex;

  ret := AL_DIO_InDataBit(lid, bit_no, @inbit);          // bit 入力

  if inbit = ALDIO_ON then PanelInDataBit.Color := clBlue
  else          PanelInDataBit.Color := clBtnFace;

  ErrDsp(ret, 'AL_DIO_InDataBit');
end;

//---------------------------------------------------------------
//      ユニット　ビット単位　出力
//---------------------------------------------------------------
procedure TMainForm.ButtonOutDataBitClick(Sender: TObject);
var
  lid           : Integer;
  bit_no        : Integer;
  outbit        : Integer;
  ret           : Integer;
begin
  lid := UpDownLid.Position;
  bit_no := ComboBoxBitNum.ItemIndex;

  if CheckBoxOutDataBit.Checked = True then outbit := ALDIO_ON
  else
                                        outbit := ALDIO_OFF;

  ret := AL_DIO_OutDataBit(lid, bit_no, outbit);           // bit 出力
ErrDsp(ret, 'AL_DIO_OutDataBit');
```

```
end;


//------------------------------------------------------------
//      ユニット　入力
//------------------------------------------------------------
procedure TMainForm.ButtonInDataClick(Sender: TObject);
var
  lid           : Integer;
  ret           : Integer;
begin
  lid := UpDownLid.Position;
  ret := InDataDisp(lid);

  ErrDsp(ret, 'AL_DIO_InData');
end;

//------------------------------------------------------------------------------
function TMainForm.InDataDisp(Lid: Integer): Integer;
var
  I             : Integer;
  indata        :Word;
begin
  indata := 0;

  Result := AL_DIO_InData(Lid, @indata);                    // 入力

  if Result <> AL_ER_OK then Exit;

                                                            // ビット表示
  for i := 0 to 15 do
  begin
    if (indata and ($0001 shl i)) = $0000 then
                pnWordIB[i].Color := clBtnFace
    else        pnWordIB[i].Color := clBlue;
  end;

  PanelInData.Caption := '0x' + IntToHex(indata, 4);
end;

//------------------------------------------------------------------------------
//         ユニット　出力
//------------------------------------------------------------------------------
procedure TMainForm.ButtonOutDataClick(Sender: TObject);
var
  lid           : Integer;
  ret           : Integer;
begin
  lid := UpDownLid.Position;
  ret := OutDataDisp(lid);
```

```
  ErrDsp(ret, 'AL_DIO_OutData');
end;

//---------------------------------------------------------------------------
function TMainForm.OutDataDisp(Lid: Integer): Integer;
var
  i             : Integer;
  outdata       :Word;
begin
  outdata := 0;

                                                  // outdata create
  for i := 0 to 15 do
  begin
        if cbWordOB[i].Checked then outdata := outdata or ($0001 shl i);
  end;

  Result := AL_DIO_OutData(Lid, outdata);                    // 出力
end;

//---------------------------------------------------------------------------
//      エラー表示
//---------------------------------------------------------------------------
procedure TMainForm.ErrDsp(Ret: Integer; DspStr: String);
var
  msg       : String;
begin

  case Ret of -1:
                          msg := 'ユニットの指定が間違っています';
  AL_ER_OK:               msg := DspStr + ': 正常';
  AL_ER_ALREADYOPEN:      msg := DspStr + ': 既にオープンしています';
  AL_ER_NOTOPEN:          msg := DspStr + ': オープンされていません';
  AL_ER_INVALIDPARAM:     msg := DspStr + ': 無効な引数';
  AL_ER_OPENDEVICE:       msg := DspStr + ': デバイスが使用不可です';
  AL_ER_CREATETHREAD:     msg := DspStr + ': スレッド作成失敗';
  AL_ER_CREATESEMAPH:     msg := DspStr + ': セマフォ作成失敗';
  AL_ER_CREATEMAIL:       msg := DspStr + ': メイルスロット作成失敗';
  AL_ER_CREATEMAP:        msg := DspStr + ': マップトファイル作成失敗';
  AL_ER_INICONFIG:        msg := DspStr + ': INI ファイルの設定が間違っています';
  AL_ER_ALREADYSTART:     msg := DspStr + ': 既にスタートしているかストップされていない'
  AL_ER_NOTSTART:         msg := DspStr + ': スタートされていません'

else                    msg := DspStr+':エラー';
  end;

  StatusBarInfo.SimpleText := msg;
end;
```

２）C++ 用　デジタル入出力ユニットサンプル

DLL とのリンク部分とユニット制御関連のオープン部と実際の入出力部分のみのサンプルを次に示します。
（このソースはセットアップでインストールされる C++ Builder5 のサンプルから抜粋しています。）

```
//---------------------------------------------------------------------------
//       DLL リンク
//---------------------------------------------------------------------------
void __fastcall TFormMain::FormCreate(TObject *Sender)
{
  char fname[100];

  //  DLL ファイル名取得
  strcpy(&fname[0] , ExtractFilePath(Application->ExeName).c_str());
  strcat(&fname[0], "ALink.dll");
  // DLL リンク(この関数はサンプルファイル AlgALink.cpp/h に存在します)
  LoadALinkDll(&fname[0]);                        // DLL リンク

}

//---------------------------------------------------------------------------
//      DLL リンク開放
//---------------------------------------------------------------------------
void __fastcall TFormMain::FormClose(TObject *Sender, TCloseAction &Action)
{
  UnLoadALinkDll();
}

//---------------------------------------------------------------------------
//      デジタル入出力ユニット オープン・クローズ
//---------------------------------------------------------------------------
void __fastcall TFormMain::ButtonOpenClick(TObject *Sender)
{
  int ret;
  int Lid;
  char ms[256];

  Lid = StrToIntDef(ComboBoxL->Text, 1);                // ユーザ指定のロジカル ID


switch (((TButton *)Sender)->Tag) {
  case 1:
        ret = AL_DIO_Open(Lid);                         // オープン
        if (ret) {
                wsprintf(&ms[0], "LID=%02d OPEN 失敗 RET=%d", Lid, ret);
        }
        else {
                // 正常終了
                wsprintf(&ms[0], "LID=%02d OPEN 完了", Lid);
                fOpen[(Lid - 1)] = TRUE;
        }
        break;
```

```
  case 2:
        ret = AL_DIO_Close(Lid);                        // クローズ
        if (ret) {
                wsprintf(&ms[0], "Lid=%02d CLOSE 失敗 RET=%d", Lid, ret);
        }
        else {
                // 正常終了
                wsprintf(&ms[0], "Lid=%02d CLOSE 完了", Lid);
                fOpen[(Lid - 1)] = 0;
        }
        break;
  }
                                                // メッセージ出力
  FormMain->ListBoxMsg->Items->Insert(FormMain->ListBoxMsg->Items->Count, &ms[0]);
}

//---------------------------------------------------------------------------
//       ユニット入力          引数：ロジカル ID
//---------------------------------------------------------------------------
WORD __fastcall IOMonitor::input(int Lid)
{
  WORD ret;

  AL_DIO_InData(Lid, &ret);

  return ret;
}

//---------------------------------------------------------------------------
//       ユニット出力          引数：ロジカル ID, 出力データ
//---------------------------------------------------------------------------
void __fastcall IOMonitor::output(int Lid, WORD odata)
{
  AL_DIO_OutData(Lid, odata);
}
```

このマニュアルについて

改訂履歴

日時	バージョン	変更点
2002.04.01	Rev 1.00	初版
2003.03.25	Rev 1.01	PC カード対応
2003.12.08	Rev 1.02	A-Link Ver2 通信 IC 対応
2003.12.24	Rev 1.03	下位 DLL（A-Link Ver2 通信 IC）関数追加
2004.07.29	Rev 1.04	ちび丸君シリーズアナログ入出力ユニット対応
2004.08.20	Rev 1.05	PC カード（A-Link Ver2 通信 IC）対応
2004.12.16	Rev 1.06	MECHATROLINK/A-Link PCI ボード対応、ボード型式変更
2005.03.03	Rev 1.10	全面改訂
2006.07.24	Rev 1.11	住所変更(本社)
2007.10.25	Rev 1.12	住所変更
2008.11.12	Rev 1.13	ALD シリーズアナログ入出力ユニット対応　ロゴ変更
2010.10.29	Rev 1.14	AP シリーズ、AP500 シリーズ向け A-Link 対応
2012.08.07	Rev 1.15	FC*2*/*3*、AP6**/7**シリーズ向け A-Link 対応
2013.02.14	Rev 1.16	EC100 シリーズ向け A-Link 対応
2013.11.15	Rev 1.17	FC*2*/*3*、AP6**/7**シリーズ向け A-Link Win7 64bit 用対応
2014.02.26	Rev 1.18	住所表記変更
2022.08.05	Rev 1.19	A-Link Ver2 PCI Express 対応

76DLH0001Q　　2022年　8月　第17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/