ＡＬＧＯ省配線シリーズ

リファレンスマニュアル

CPU 付き A-Link
PCI マスタユニット DLL

ALGO

目　次

第１章　関数一覧

第２章　関数仕様

第１章　関数一覧

１－１　関数一覧

１）マスタ制御関数

関　数	機　能
ALink_Open( )	ボードをオープンします。
ALink_Close( )	ボードをクローズします。
ALink_Start( )	通信の開始，停止を制御します。
ALink_SetCommInfo( )	通信設定を行います。
ALink_SetSystemConfig( )	システム構成情報を設定します。
ALink_GetLineStatus( )	ライン状態を取得します。
ALink_Write( )	D.P.RAM にデータを書込みます。
ALink_Read( )	D.P.RAM のデータを読込みます。

２）スレーブ制御関数

関　数	機　能
ALink_CmdWrite( )	コマンドエリアにデータを書込みます。
ALink_CmdWriteWord( )	コマンドエリアにワードデータを書込みます。
ALink_CmdRead( )	コマンドエリアからデータを読込みます。
ALink_CmdReadWord( )	コマンドエリアからワードデータを読込みます。
ALink_ResWrite( )	レスポンスエリアにデータを書込みます。
ALink_ResWriteWord( )	レスポンスエリアにワードデータを書込みます。
ALink_ResRead( )	レスポンスエリアからデータを読込みます。
ALink_ResReadWord( )	レスポンスエリアからワードデータを読込みます。
ALink_GetSlaveStatus( )	スレーブの通信状態を取得します。

３）割込み制御関数

関　数	機　能
ALink_SetInterrupt( )	割込みの設定をします。
CALINK_INTHANDLER( )	割込み処理コールバック関数書式
ALink_SetDataRenewal( )	入力変化割込みの設定を行います。

第２章　関数仕様

２－１　マスタ制御関数

ALink_Open 関数

機能　　マスタユニットをオープンします。

書式
```
int  ALink_Open(
        int  Board
);
```

引数　　Board　　：マスタユニットのボード番号 [0～3]

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_OPENDEVICE	：ユニットをオープンできません
CAL_ER_ALREADYOPEN	：すでにオープンしています
CAL_ER_CREATE	：内部オブジェクトが生成できません

説明　　ボード番号で指定した A-Link マスタユニットをオープンします。
この関数コールが正常終了すると、A-Link マスタユニットにアクセス可能となります。
A-Link マスタユニットを使用するためには、この関数をコールする必要があります。

ALink_Close 関数

機能　　マスタユニットをクローズします。

書式

```
int  ALink_Close(
        int  Board
);
```

引数

Board	：マスタユニットのボード番号 [0～3]

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　ボード番号で指定した A-Link マスタユニットをクローズします。
この関数コールが正常終了すると、A-Link マスタユニットにアクセス不可となります。
A-Link マスタユニットの使用を終了する場合、この関数をコールする必要があります。
アプリケーションで A-Link マスタユニットを使用している場合は、アプリケーションを終了する前に必ずコールするようにしてください。

ALink_Start 関数

機能　　マスタユニットの通信の開始･停止を制御します。

書式

```
int  ALink_Start(
        int  Board,
        int  Line,
        int  CtrlFlag
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
CtrlFlag	：通信制御 [0: 停止、1: 開始]

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　ボード番号、ライン番号で指定した A-Link 通信ラインの通信を制御します。
A-Link 通信の開始･停止を制御することが出来ます。

ALink_SetCommInfo 関数

機能　　通信の設定を行います。

書式

```
int  ALink_SetCommInfo(
        int  Board,
        int  Line,
        CAL_COMMINFO_REC  *pComInfoRec
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
pCommInfoRec	：通信設定情報を格納した構造体のポインタ

通信設定情報

```
typedef struct {
    USHORT      EndOfSlave;
    USHORT      Baudrate;
    USHORT      FullHalf;
    USHORT      RetryCount;
} CAL_COMMINFO_REC;
```

EndOfSlave　：スレーブ最終アドレス [1～63]
Baudrate　：通信速度設定

BIT	内容
D0-D7	通信レート [1: 3Mbps、2: 6Mbps、3: 12Mbps]
D8-D15	フレームモード [0～7] (通常使用時は 0)

FullHalf　：通信モード [0: 半二重、1: 全二重]
RetryCount　：リトライ回数 [0～7]

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません
CAL_ER_ALREADYSTART	：通信がスタートしています
CAL_ER_TIMEOUT	：応答タイムアウト

説明　　ボード番号、ライン番号で指定した A-Link 通信ラインの通信設定を行います。
通信を開始する前に、動作環境に合わせた通信設定を行っておく必要があります。
通信中は設定を行うことが出来ません。通信を終了させてから設定を行う必要があります。

ALink_SetSystemConfig 関数

機能　　システム構成情報を設定します。

書式

```
int  ALink_SetSystemConfig(
        int  Board,
        int  Line
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません
CAL_ER_ALREADYSTART	：通信がスタートしています
CAL_ER_TIMEOUT	：応答タイムアウト

説明　　ボード番号、ライン番号で指定した A-Link 通信ラインのシステム構成情報を設定します。システム情報はファイルから取得します。DLL と同じディレクトリ内にあるシステム情報ファイルを読み込んでマスタに設定を行います。通信中は設定を行うことが出来ません。通信を終了させてから設定を行う必要があります。

システム情報ファイル名： AlinkConfig.dat

ALink_GetLineStatus 関数

機能　　ライン状態を取得します。

書式

```
int  ALink_ GetLineStatus (
        int  Board,
        int  Line,
        CAL_LINESTATUS_REC  *pLineStatusRec
);
```

引数

Board　　：マスタユニットのボード番号 [0～3]
Line　　：A-Link 通信ライン番号 [1～]
pLineStatusRec　：ライン状態情報を格納した構造体のポインタ

ライン状態情報

```
typedef struct {
    USHORT      Master;
    USHORT      Control;
    USHORT      EndOfSlave;
    USHORT      Baudrate;
    USHORT      FullHalf;
    USHORT      RetryCount;
    USHORT      RunCounter;
} CAL_LINESTATUS_REC;
```

Master　　：マスタの状態 [0: 停止、1: 起動、2: 異常]
Control　　：コントロール状態 [0: 通信停止、1: 通信開始、2: 通信異常]
EndOfSlave　：スレーブ最終アドレス [1～63]
Baudrate　　：通信速度設定

BIT	内容
D0-D7	通信レート [1: 3Mbps、2: 6Mbps、3: 12Mbps]
D8-D15	フレームモード [0～7] (通常使用時は 0)

FullHalf　　：通信モード [0: 半二重、1: 全二重]
RetryCount　：リトライ回数 [0～7]
RunCounter　：100msec 周期でのインクリメンタルデータ（マスタ動作確認用）

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明

ボード番号、ライン番号で指定した A-Link 通信ラインのライン情報を取得します。
マスタの状態、通信状態などを確認することが出来ます。

ALink_Write 関数

機能　D.P.RAM にデータを書込みます。

書式

```
int  ALink_Write (
        int  Board,
        int  Line,
        int  Offset,
        void  *pBuffer,
        int  Length
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
Offset	：D.P.RAM 先頭からのバイトオフセット
pBuffer	：書込みデータが格納されたポインタ
Length	：書込みデータのバイト長

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　ボード番号、ライン番号で指定した A-Link 通信ライン用の D.P.RAM にデータを書込みます。データ書込みは 2 バイト単位で行われます。そのため、書込み処理の際に Length は小さい側の 2 の倍数値で処理されます。（例： Length=5 → 4）

ALink_Read 関数

機能　D.P.RAM からデータを読込みます。

書式

```
int  ALink_Read (
        int  Board,
        int  Line,
        int  Offset,
        void  *pBuffer,
        int  Length
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
Offset	：D.P.RAM 先頭からのバイトオフセット
pBuffer	：読込みデータを格納するためのポインタ
Length	：読込みデータのバイト長

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　ボード番号、ライン番号で指定した A-Link 通信ライン用の D.P.RAM にからデータを読込みます。
データ読込みは 2 バイト単位で行われます。そのため、読込み処理の際に Length は小さい側の 2 の倍数値で処理されます。（例： Length=5 → 4）

２－２　スレーブ制御関数

ALink_CmdWrite 関数

機能　　コマンドエリアにデータを書込みます。

書式

```
int  ALink_CmdWrite(
        int  Board,
        int  Line,
        int  Slave,
        CAL_ST_REC  *pStRec
);
```

引数

Board	：マスタユニットのボード番号［0～3］
Line	：A-Link 通信ライン番号［1～］
Slave	：スレーブアドレス［1～63］
pStRec	：ST 区画データを格納するポインタ

ST 区画データ

```
typedef struct {
    USHORT      CmdRes1;
    USHORT      CmdRes2;
    USHORT      Param1;
    USHORT      Param2;
    USHORT      Param3;
    USHORT      Param4;
} CAL_ST_REC;
```

CmdRes1	：コマンド 1
CmdRes2	：コマンド 1
Param1	：パラメータ 1
Param2	：パラメータ 2
Param3	：パラメータ 3
Param4	：パラメータ 4

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　コマンドエリアの指定したスレーブに対応する ST 区画にデータを書込みます。

ALink_CmdWriteWord 関数

機能　　コマンドエリアにワードデータを書込みます。

書式

```
int  ALink_CmdWriteWord(
        int  Board,
        int  Line,
        int  Slave,
        int  DataOffset,
        USHORT  Data
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
Slave	：スレーブアドレス [1～63]
DataOffset	：ST 区画先頭からのワードオフセット
Data	：書込みデータ

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　コマンドエリアの指定したスレーブに対応する ST 区画にワードデータを書込みます。
ST 区画の先頭からのワードオフセットを DataOffset によって指定します。指定された箇所にワードデータを書込みます。

ST 区画データ：

コマンド 1	DataOffset = 0
コマンド 2	DataOffset = 1
パラメータ 1	DataOffset = 2
パラメータ 2	DataOffset = 3
パラメータ 3	DataOffset = 4
パラメータ 4	DataOffset = 5

ALink_CmdRead 関数

機能　　コマンドエリアからデータを読込みます。

書式

```
int  ALink_CmdRead(
        int  Board,
        int  Line,
        int  Slave,
        CAL_ST_REC  *pStRec
);
```

引数

Board	：マスタユニットのボード番号［0～3］
Line	：A-Link 通信ライン番号［1～］
Slave	：スレーブアドレス［1～63］
pStRec	：ST 区画データを格納するためのポインタ

ST 区画データ

```
typedef struct {
    USHORT      CmdRes1;
    USHORT      CmdRes2;
    USHORT      Param1;
    USHORT      Param2;
    USHORT      Param3;
    USHORT      Param4;
} CAL_ST_REC;
```

CmdRes1	：コマンド 1
CmdRes2	：コマンド 1
Param1	：パラメータ 1
Param2	：パラメータ 2
Param3	：パラメータ 3
Param4	：パラメータ 4

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　コマンドエリアの指定したスレーブに対応する ST 区画からデータを読込みます。

ALink_CmdReadWord 関数

機能　　コマンドエリアからワードデータを読込みます。

書式

```
int  ALink_CmdReadWord(
        int  Board,
        int  Line,
        int  Slave,
        int  DataOffset,
        USHORT  *pData
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
Slave	：スレーブアドレス [1～63]
DataOffset	：ST 区画先頭からのワードオフセット
pData	：読込みデータを格納するためのポインタ

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　コマンドエリアの指定したスレーブに対応する ST 区画からワードデータを読込みます。
ST 区画の先頭からのワードオフセットを DataOffset によって指定します。指定された箇所からワードデータを読込みます。

ST 区画データ：

コマンド 1	DataOffset = 0
コマンド 2	DataOffset = 1
パラメータ 1	DataOffset = 2
パラメータ 2	DataOffset = 3
パラメータ 3	DataOffset = 4
パラメータ 4	DataOffset = 5

ALink_ResWrite 関数

機能　　レスポンスエリアにデータを書込みます。

書式

```
int  ALink_ResWrite(
        int  Board,
        int  Line,
        int  Slave,
        CAL_ST_REC  *pStRec
);
```

引数

Board	：マスタユニットのボード番号［0～3］
Line	：A-Link 通信ライン番号［1～］
Slave	：スレーブアドレス［1～63］
pStRec	：ST 区画データを格納するポインタ

ST 区画データ

```
typedef struct {
    USHORT      CmdRes1;
    USHORT      CmdRes2;
    USHORT      Param1;
    USHORT      Param2;
    USHORT      Param3;
    USHORT      Param4;
} CAL_ST_REC;
```

CmdRes1	：レスポンス 1
CmdRes2	：レスポンス 1
Param1	：パラメータ 1
Param2	：パラメータ 2
Param3	：パラメータ 3
Param4	：パラメータ 4

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　レスポンスエリアの指定したスレーブに対応する ST 区画にデータを書込みます。

ALink_ResWriteWord 関数

機能　　レスポンスエリアにワードデータを書込みます。

書式

```
int  ALink_ResWriteWord(
        int  Board,
        int  Line,
        int  Slave,
        int  DataOffset,
        USHORT  Data
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
Slave	：スレーブアドレス [1～63]
DataOffset	：ST 区画先頭からのワードオフセット
Data	：書込みデータ

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　レスポンスエリアの指定したスレーブに対応する ST 区画にワードデータを書込みます。
ST 区画の先頭からのワードオフセットを DataOffset によって指定します。指定された箇所にワードデータを書込みます。

ST 区画データ：

コマンド 1	DataOffset = 0
コマンド 2	DataOffset = 1
パラメータ 1	DataOffset = 2
パラメータ 2	DataOffset = 3
パラメータ 3	DataOffset = 4
パラメータ 4	DataOffset = 5

ALink_ResRead 関数

機能　　コマンドエリアからデータを読込みます。

書式

```
int  ALink_ResRead(
        int  Board,
        int  Line,
        int  Slave,
        CAL_ST_REC  *pStRec
);
```

引数

Board	：マスタユニットのボード番号［0～3］
Line	：A-Link 通信ライン番号［1～］
Slave	：スレーブアドレス［1～63］
pStRec	：ST 区画データを格納するためのポインタ

ST 区画データ

```
typedef struct {
    USHORT      CmdRes1;
    USHORT      CmdRes2;
    USHORT      Param1;
    USHORT      Param2;
    USHORT      Param3;
    USHORT      Param4;
} CAL_ST_REC;
```

CmdRes1	：レスポンス 1
CmdRes2	：レスポンス 1
Param1	：パラメータ 1
Param2	：パラメータ 2
Param3	：パラメータ 3
Param4	：パラメータ 4

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　レスポンスエリアの指定したスレーブに対応する ST 区画からデータを読込みます。

ALink_ResReadWord 関数

機能　　レスポンスエリアからワードデータを読込みます。

書式

```
int  ALink_ResReadWord(
        int  Board,
        int  Line,
        int  Slave,
        int  DataOffset,
        USHORT  *pData
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
Slave	：スレーブアドレス [1～63]
DataOffset	：ST 区画先頭からのワードオフセット
pData	：読込みデータを格納するためのポインタ

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　レスポンスエリアの指定したスレーブに対応する ST 区画からワードデータを読込みます。ST 区画の先頭からのワードオフセットを DataOffset によって指定します。指定された箇所からワードデータを読込みます。

ST 区画データ：

コマンド 1	DataOffset = 0
コマンド 2	DataOffset = 1
パラメータ 1	DataOffset = 2
パラメータ 2	DataOffset = 3
パラメータ 3	DataOffset = 4
パラメータ 4	DataOffset = 5

ALink_GetSlaveStatus 関数

機能　　スレーブの通信状態を取得します。

書式

```
int  ALink_GetSlaveStatus(
        int  Board,
        int  Line,
        int  Slave,
        USHORT  *pStatus
);
```

引数

Board	：マスタユニットのボード番号［0～3］
Line	：A-Link 通信ライン番号［1～］
Slave	：スレーブアドレス［1～63］
pStatus	：スレーブ通信状態［0: 正常、1: 異常］

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　指定したスレーブの通信状態を取得します。

２－３　割込み制御関数

ALink_SetInterrupt 関数

機能　　割込みの設定をします。

書式

```
int  ALink_SetInterrupt(
        int  Board,
        int  Line,
        USHORT  IntMask,
        CALINK_INTHANDLER  IntHandler,
        void  *pArg
);
```

引数

Board　　：マスタユニットのボード番号［0～3］
Line　　：A-Link 通信ライン番号［1～］
IntMask　　：割込みマスク設定（0 で割込み設定解除）
IntHandler　：割込み処理ユーザー関数（IntMask が 0 のときは無視されます）
pArg　　：ユーザーデータのポインタ（IntMask が 0 のときは無視されます）

割込みマスク設定

BIT	内容
D0	SCANR（SCAN Read タイミング）
D1	CHK2（CHK2 の発生）
D2～D7	なし
D8	Data Renewal 1
D9	Data Renewal 2
D10	Data Renewal 3
D11	Data Renewal 4
D12	Data Renewal 5
D13	Data Renewal 6
D14	Data Renewal 7
D15	Data Renewal 8

割込み処理ユーザー関数

```
typedef void (* CALINK_INTHANDLER)(
    int  Board,
    int  Line,
    USHORT  IntStatus,
    USHORT  *pRenewalData,
    void  *pArg
);
```

*** 詳細は、CALINK_INTHANDLER 関数を参照してください。**

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません
CAL_ER_ALREADYSTART	：通信がスタートしています
CAL_ER_CREATE	：内部オブジェクトが生成できません (他のプロセスなどで割込み設定されている)

説明

指定した A-Link 通信ラインの割込み設定を行います。
割込み設定は通信開始とともに有効となります。通信通信中は設定を行うことが出来ません。通信を終了させてから設定を行う必要があります。
割込み設定が正常に行われると、割込み発生時に割込み処理ユーザー関数が呼び出される様になります。
1 つのラインに対する割込みは、OS 上で 1 つしか設定できません。他のプロセスなどですでに割込みが設定されてる場合は設定できません。
割込み処理を必要としなくなった場合、または、ボードをクローズする場合は、IntMask=0 で関数をコールして、割り込み設定を解除してください。

CALINK_INTHANDLER 関数

機能　　割込み処理ユーザー関数

書式

```
void  CALINK_INTHANDLER(
        int  Board,
        int  Line,
        USHORT  IntStatus,
        USHORT  *pRenewalData,
        void  *pArg
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
IntStatus	：割込みステータス
pRenewalData	：割込み発生時の Data Renewal スレーブデータを格納したポインタ
pArg	：ユーザーデータを格納したポインタ

割込みマスク設定

BIT	内容
D0	SCANR (SCAN Read タイミング)
D1	CHK2 (CHK2 の発生)
D2～D7	なし
D8	Data Renewal 1
D9	Data Renewal 2
D10	Data Renewal 3
D11	Data Renewal 4
D12	Data Renewal 5
D13	Data Renewal 6
D14	Data Renewal 7
D15	Data Renewal 8

Data Renewal スレーブデータ

Data Renewal 1 データ
Data Renewal 2 データ
Data Renewal 3 データ
Data Renewal 4 データ
Data Renewal 5 データ
Data Renewal 6 データ
Data Renewal 7 データ
Data Renewal 8 データ

戻り値　　なし

説明　　割込み発生時にコールされる関数。
ユーザーはこの書式に従って、割込み処理ユーザー関数を作成します。

ALink_SetDataRenewal 関数

機能　　入力変化割込みデータの設定を行います。

書式

```
int  ALink_SetDataRenewal(
        int  Board,
        int  Line,
        int  RenewalNo,
        CAL_RENEWALINFO_REC  *pRenewalInfo
);
```

引数

Board	：マスタユニットのボード番号 [0～3]
Line	：A-Link 通信ライン番号 [1～]
RenewalNo	：Data Renewal 番号 [1～8]
pRenewalInfo	：Data Renewal 設定データを格納するポインタ

Data Renewal 設定データ

```
typedef struct {
    USHORT      Slave;
    USHORT      Param;
    USHORT      Renewal;
} CAL_RENEWALINFO_REC;
```

Slave	：スレーブアドレス
Param	：パラメータ番号 [1～4]
Renewal	：入力変化設定（変化監視対象とする Bit を 1 にする）

戻り値

CAL_ER_OK	：正常
CAL_ER_INVALIDPARAM	：無効な引数です
CAL_ER_NOTOPEN	：オープンされていません

説明　　入力変化割込みデータの設定を行います。
入力変化割込みデータを有効にするには、設定した Data Renewal 番号の割込みを ALink_SetInterrupt 関数を用いて有効にする必要があります。
この関数で入力変化割込みデータを設定した後、ALink_SetInterrupt 関数をコールする様にしてください。

このリファレンス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。

７６ＤＬＨ００１９Ａ

ALGO 株式会社アルゴシステム

本社
〒５８７―００２１　大阪府南河内郡美原町小平尾６５６　　ＴＥＬ（０７２）３６２－５０６７
ＦＡＸ（０７２）３６２－４８５６

大阪営業所
〒５４２－００８１　大阪市中央区南船場１－１２－３　船場グランドビル３Ｆ　　ＴＥＬ（０６）６２６３－９５７５
ＦＡＸ（０６）６２６３－９５７６

東京営業所
〒１０４－００６１　東京都中央区銀座７－１５－８　銀座堀ビル２Ｆ　　ＴＥＬ（０３）３５４１－７１７０
ＦＡＸ（０３）３５４１－７１７５

ホームページ　http://www.algosystem.co.jp/