ＡＬＧＯ省配線システム

ユーザーズマニュアル

CN43eMst.DLL

目　次

CN43eMst.DLLの概要

第１章 アプリケーション開発

第２章 DLL関数

第３章 iniファイルについて

CN43eMst.DLL の概要

本 DLL「CN43eMst.DLL」は、CUnet PCI Express ボードを利用する Windows アプリケーションの作成を容易にするために提供されます。本 DLL は、「Windows Embedded Standard 7 64 ビット版」「Windows 10 IoT Enterprise 64 ビット版」にて使用することができます。

ユーザーは、Microsoft Visual C++等の開発言語から、DLL をコールすることによってアプリケーションを作成することができます。32 ビット/64 ビットアプリケーション用にそれぞれ DLL を用意しています。作成するアプリケーションに合わせて DLL を選択してください。

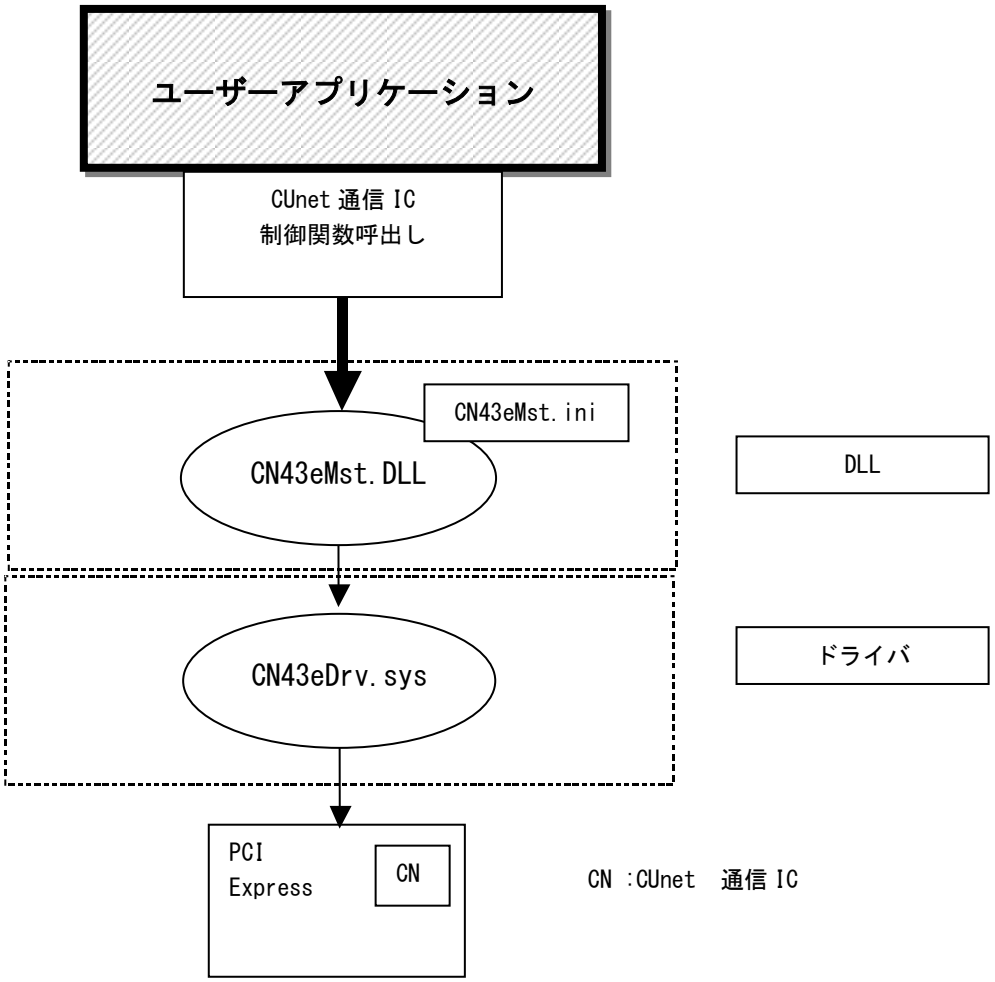
本 DLL は､CUnet 拡張基板にアクセスするためのデバイスドライバ｢CN43eDrv.sys｣を内部でコールするため、ユーザーはデバイスドライバを意識する必要はありません。

該当する関数をコールすることで、CUnet 通信での GM（グローバルメモリ）へのアクセス、メール機能、レジスタへのアクセス等を簡単に行うことができます。

第１章　アプリケーション開発

１－１　CN43eMst.DLL 動作環境

ユーザーは作成するアプリケーション内で CN43eMst.DLL の関数をコールすることにより、CUnet の入出力を処理します。
作成したアプリケーション、CN43eMst.DLL は同一フォルダ（ディレクトリ）に格納します。



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++

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

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

③プロジェクトへ、CNMstFnc.cpp を追加します。

④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。LoadCNMstDll(L“CN43eMst.dll”);

⑤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。UnLoadCNMstDll();

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。

ただし、「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、CNMstFnc.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 "CNMstFnc.h"
```

第２章　DLL 関数

２－１　DLL 関数概要

DLL 関数は、ボード名称, DLL Version 等を取得する Utility 関数、ボードのオープン,クローズ,CUnet 通信の開始等を実行する Open/Close 関数、CUnet メール機能を制御するメール関数、CUnet 通信 IC のレジスタにアクセスするレジスタアクセス関数、CUnet 通信 IC のグローバルメモリをアクセスする GM アクセス関数が用意されています。

各関数の詳細は、関数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）Open/Close 関数

DLL 関数（Utility 関数を除く）を使用するには、CN_open()関数をコールする必要があります。さらにメール関数、レジスタアクセス関数、GM アクセス関数を有効にする為には CN_start()関数にて CUnet 通信を開始する必要があります。この CN_start()関数にて自己の占有するステーションアドレス、GM エリアを指定します。

CUnet 通信を終了する際には、CN_stop()関数をコールして下さい。またボードアクセスが不要になった場合は CN_close()関数をコールして下さい。

●複数アプリケーションでの Open / Close

一つの CUnet 通信 IC に対して複数のアプリケーションがボードオープンを行う際､既に一つのアプリケーションが CN_open()関数をコールしボードのオープンを行っていたとして、以後に別のアプリケーションが CN_open()関数をコールしてもエラーは返らず正常に動作します。

CUnet 通信を開始する際、現状態が通信開始中、停止中にかかわらず CN_start()関数で指定したパラメータで通信を開始します。他のアプリケーションですでに通信が開始されている場合は注意が必要です。

CUnet 通信を停止する際､一つのアプリケーションが CN_stop()関数をコールすると他の全てのアプリケーションで CUnet 通信が停止します。

以下に関数コールの手順を示します。

・ボードオープンから CUnet 通信開始

```
CN_open(Board);
CN_start(Board, Line, StationAdr, StationSize, ByteBusSize);
```

・CUnet 通信終了から別ステーショアドレスで CUnet 通信開始

```
CN_stop(Board, Line);
CN_start(Board, Line, StationAdr, StationSize, ByteBusSize);
```

・CUnet 通信終了からボードクローズ

```
CN_stop(Board, Line);
CN_close(Board);
```

２）メール関数

メール関数を使用するにはまず、CUnet 通信状態で CN_MailOpen()関数をコールする必要があります。メール送信では CN_MailSend()関数をコールします。最大 256Byte のデータを送信することができます。前回の送信がまだ終わっていなければ、送信せずにエラーを返します。メール送信の結果取得は、CN_MailStatus()関数、もしくは CN_MailWaitObject()関数にて取得できます。メール送信完了前に CN_MailStatus()関数をコールするとステータスは 0 が読込めます。

CN_MailWaitObject()関数をコールするとメール送信完了、メール受信、メール終了(MailClose)のいずれかがシグナル状態になるまで制御を戻しません。そのため、この関数はスレッド内部でコールして下さい。

メール受信は CN_MailReceive()関数をコールします。受信したデータを取得します。メールを着信していない状態でこの関数をコールするとエラーを返します。メールの着信は CN_MailStatus()関数にて知ることができます。

●複数アプリケーションでのメール機能の使用

一つの CUnet 通信 IC に対して複数のアプリケーションがメール機能を使用することはできません。

既に一つのアプリケーションが CN_MailOpen()関数をコールしていれば、以後に別のアプリケーションが CN_MailOpen()関数をコールするとエラー[MKY43_ER_ALREADYMAILOPEN]を返します。

以下に関数コールの手順を示します。

・メールのオープン

```
CN_MailOpen(Board, Line);
```

・メールの送信

CN_MailSend()後の送信確認は別スレッドから CN_MailWaitObject()（結果が分かるまで待機状態）を呼出しチェックする方法

```
CN_MailSend(Board, Line, StationAdr, pMail, Len);
```

<別スレッド>

```
while(fEnd){
    dwRet = CN_MailWaitObject(m_Board, m_Line, Status);
    if (dwRet == MKY43_MAIL_STATUS_SENDEND ){
        // 送信完了処理
        if ((Status & 0x00ff) == 0x0001){
            // メール送信完了（正常）
        }
        if ((Status & 0x0002) == 0x0002){
            // 相手先が BUSY
        }
        if ((Status & 0x0004) == 0x0004){
            // 相手先が不在
        }
    }else if (dwRet == MKY43_MAIL_STATUS_RECEIVE){
        // 受信処理
    }
    Sleep(StaTimer);
}
```

尚、別方法としてステータスのポーリング

```
Status = 0;
while(Status == 0){
    CN_MailStatus(Board, Line, Status);
}
if ((Status & 0x00ff) == 0x0001){
    // メール送信完了（正常）
}
if ((Status & 0x0002) == 0x0002){
    // 相手先が BUSY
}
if ((Status & 0x0004) == 0x0004){
    // 相手先が不在
}
```

・メールの受信

別スレッドで CN_MailWaitObject()（結果が分かるまで待機状態）を
呼出しメール着信を知り CN_MailReceive()を発行して受信する。

〈別スレッド〉

```
while(fEnd){
    dwRet = CN_MailWaitObject(m_Board, m_Line, Status);
    if (dwRet == MKY43_MAIL_STATUS_SENDEND){
        // 送信完了処理
    }else if (dwRet == MKY43_MAIL_STATUS_RECEIVE){
        // 受信処理
        CN_MailReceive(Board, Line, pMail, Len, pStationAdr);
    }
    Sleep(StaTimer);
}
```

尚、別方法としてステータスのポーリング

```
Status = 0;
while(Status == 0){
    CN_MailStatus(Board, Line, Status);
}
if ((Status & 0xff00) == 0x0100){
    // メール着信
    CN_MailReceive(Board, Line, pMail, Len, pStationAdr);
}
```

・メールのクローズ

```
CN_MailClose(Board, Line);
```

３）レジスタアクセス関数
レジスタアクセス関数を使用するには CUnet 通信状態でなければなりません。

以下に関数コールの手順を示します。

メンバーフラグレジスタ（MFR）を取得
```
CN_GetReg64(Board, Line, 0x0310, pReg);
```

４）GM アクセス関数

GM アクセス関数を使用するには CUnet 通信状態でなければなりません。

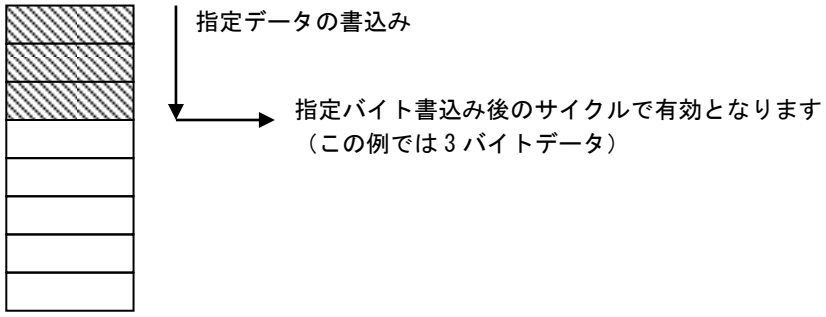
以下に関数コールの手順を示します。

ステーションアドレス 0 の先頭から 2Byte のデータを 16 ビットアクセスで書込み
```
CN_SetMemShort(Board, Line, pBuf, 0, 0, 1);
```

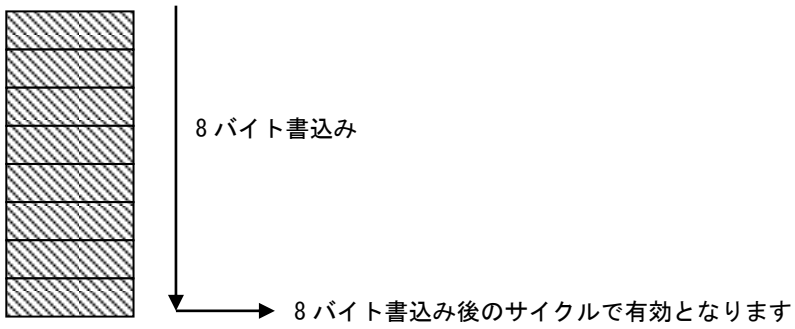
ステーションアドレス 0 の先頭から 2Byte のデータを 16 ビットアクセスで読込み
```
CN_GetMemShort(Board, Line, pBuf, 0, 0, 1);
```

●GM の書き込みについて
GM 書き込み関数ではステーションデータサイズである 8 バイトデータブロックを不整合なく GM に書込むための処理が含まれています。以下にその処理の例を示します。

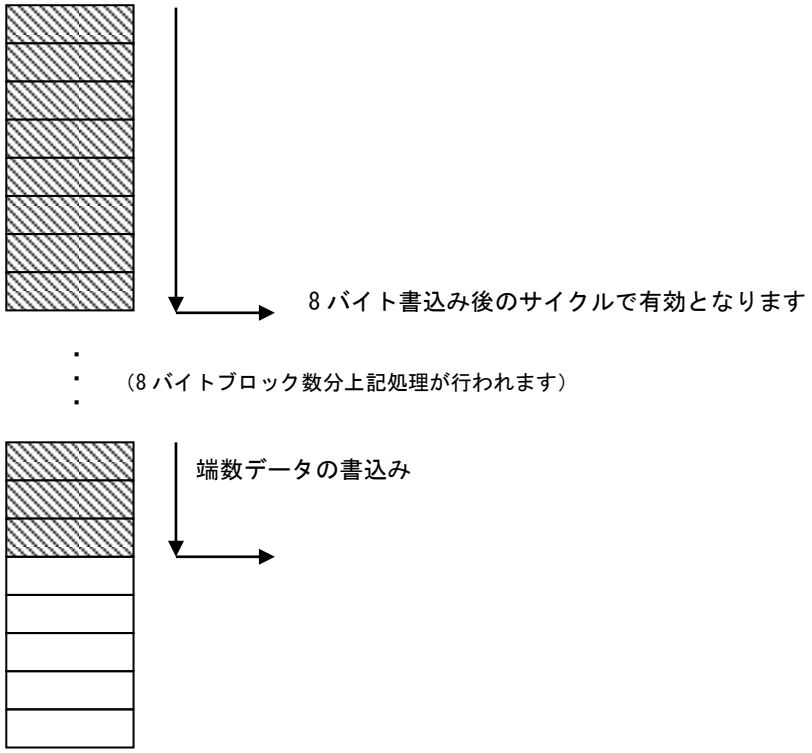
・8 バイト以下のデータの場合



・8 バイトデータの場合



・8 バイト以上のデータの場合



第３章　ini ファイルについて

３－１　ini ファイルの構成

ini ファイルは以下のように構成されます。

```
[BOARD_0]
;; 3:3Mbps, 6:6Mbps, 12:12Mbps
Baudrate_0=12
```

それぞれの項目は以下のように設定します。

[BOARD_0]
　ボード番号を指定します。
　拡張基板を使用する場合、0 のみ有効です。

Baudrate_0
　ボーレートの数値を指定します。
　　3： 3Mbps
　　6： 6Mbps
　　12：12Mbps

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。

７６ＤＬＨ００６６Ａ　　２０２２年　８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/