デバイスアートツールキット

制御モジュール

CM01

ソフトウェア マニュアル Rev. 1.0

2015 年 8 月

アークデバイス

目次

1. 概要

DATK（Device Art Tool Kit）の制御モジュール CM01 にはブートローダプログラム CM01_Boot, ユーザプログラムとして制御プログラム CM01_Ctrl が出荷時に書きこまれています．ブートローダは自己書き換え機能を持ちユーザプログラムの書き換えが可能です．CM01 ではまずブートローダが起動します．自己書き換えの指令がなければ，次にユーザプログラムである制御プログラム CM01_Ctrl が呼び出されます．

CM01 はシリアル通信モジュール（SC02 等）と接続し，パソコンからはシリアル通信モジュールを経由して CM01 と通信を行います．CM01 をパソコンから通信制御するソフトウェアを制作するには，制御プログラム CM01_Ctrl に対応したクラス CM01 を用います．クラス CM01 は Windows の Visual C++(Visual Studio 2008)のコンソールアプリ用です（Unicode には対応しておりません）．基本的にファームウェアの制御プログラム CM01_Ctrl とクラス CM01 は同じバージョンのものを使用します．本マニュアルの対象は Ver.2.1 および Ver.2.2 です．

CM01 のプログラム（ファームウェア・パソコン側ソフトウェア）はオープンソースになっておりますので，用途・環境に応じて適宜書き換えてご利用ください．

1.1 プログラム作成に必要なファイル

CM01 を SC02 に接続してプログラムを作成する場合，下記のファイルが必要となります．

CM01.h	クラス CM01 のヘッダファイル
CM01.cpp	クラス CM01 のソースファイル
SC02.h	クラス SC02 のヘッダファイル
SC02.cpp	クラス SC02 のソースファイル
ControlModule.h	クラス ControlModule のヘッダファイル
ftd2xx.h	FTDI 社 D2xx 用のヘッダファイル
ftd2xx.lib	FTDI 社 D2xx 用のライブラリ
ftd2xx.dll	FTDI 社 D2xx 用のダイナミックリンクライブラリ

SC02 は FTDI 社の USB チップ FT245RL を使用しているため，USB のデバイスドライバと専用ライブラリが必要となります．FTDI 社のホームページより D2XX ドライバをダウンロード・インストトールしてください．ダウンロードした D2XX ファイルにはドライバ本体のほかに，fed2xx.h，fed2xx.dll，fed2xx.lib が一括して含まれています．

DATK の各モジュール用のクラスは基底クラス ControlModule を継承しているため．クラス ControlModule 用のファイルが必要となります．

1.2 プログラム開始から終了まで

もっとも簡単な動作例としてエンコーダ入力ボードと PWM 出力ボードを取り付けた CM01 を SC02 と接続して使う例を示します．

ID＝2 のモジュールをオープンして，PWM 出力値の設定とエンコーダカウント値を入力するプログラムです．エラー処理等は省略しています．詳しくはサンプルプログラムのソースコードをご覧ください．

```
CM01 cm;                    // クラスの生成
SC02 sc;                    // クラスの生成

sc.Open( 1 );               // ID = 1 の SC02 をオープン
cm.Open( &sc, 0, 2 );       // ID = 2 の CM01 をオープン
制御ループ{
        cm.PwmBoard.sPwm1 = 100;  // PWM デューティー比の設定
        cm.Exchange(TRUE);        // データ送受信
        printf("enc %d¥n", cm.EncBoard.lEncoder);  // エンコーダ値の表示
}
cm.Close();             // クローズ
sc.Close();             // クローズ
```

また，CM01 を複数台使用する場合，送受信をまとめて行うことで通信間隔を高速化できます．例えば CM01 を 2 台（cm1,cm2）使用する場合，

```
cm1.Exchange(FALSE);        // この時点では送信しない
cm2.Exchange(TRUE);         // ここでまとめて送受信
```

とすることで，送受信の効率を向上できます．

ただしまとめて送信できる台数には限度があります．詳細は次節で解説いたします．

1.3 送受信データサイズ

CM01 は複数枚連結して使用することを前提としていますが，一度に送受信できるデータ量には限界があります．この限界はシリアル通信モジュール（SC02）のバッファサイズによって規定されます（SC02 は送・受信ともに 256 バイト）．

2 台のモジュールを SC02 に連結した場合の送信パケットの構成を表 1 に受信パケットの構成を表 2 に示します．

CM01 の送受信データサイズは入出力ボードの構成により変化します（表３）．したがって，送受信データサイズの計算は，入出力ボードの構成に応じて行う必要があります．

表 1　SC02 送信パケット

項目	サイズ（バイト）
コマンド	1
送受信デバイス数	1
モジュール 1 のパケット	
ポート番号	1
ID	1
送信サイズ	2
受信サイズ	2
コマンド	1
データ	X
チェックサム	1
モジュール 2 のパケット	

表 2　SC02 受信パケット

項目	サイズ（バイト）
モジュール 1 のデータ	
ポート番号	1
ID	1
受信サイズ	2
データ	X
ステータス	1
チェックサム	1
モジュール 2 のパケット	
ステータス	1

表 3 各入出力ボード送受信データサイズ（単位：バイト）

入出力ボード	送信データ	受信データ
エンコーダ入力ボード	0	14
センサ入力ボード	0	16
PWM 出力ボード	4	0
DA 出力ボード	2	0
PIO ボード A・B	4	2

・計算例

2 台の CM01（PIO ボード A＋PWM 出力ボート）を SC02 に連結し Exchange コマンドを実行した場合の送受信パケットサイズを計算します．表 1 ～3 より，送信データ以外の要素は 8 バイト，PIO ボード A の送信データは 4 バイト，PWM 出力ボードは 0 バイトであるため，1 台の CM01 の送信パケットサイズ TxSize は，

TxSize = 8 バイト+ 4 バイト+ 4 バイト= 16 バイト

となります．同様に CM01 の受信パケットサイズ RxSize は，

RxSize = 6 バイト+ 0 バイト+ 2 バイト= 8 バイト

となります．

SC02 の送信データ以外の要素は 2 バイトなので SC02 の送信パケットサイズ ScTxSize は，

ScTxSize = 2 バイト+ 16 バイト×2 = 34 バイト

となります．同様に SC02 の受信パケットサイズ ScRxSize は，

ScRxSize = 1 バイト+ 8 バイト×2 = 17 バイト

となります．

2. 入出力データ構造体

2.1 CM01_INPUT_ENC 型構造体

定義

```
typedef struct _CM01_INPUT_ENC
{
    long lEncoder;          エンコーダのカウント値.
    WORD wSensor[5];        センサ入力のA/D変換値.
} CM01_INPUT_ENC;
```

概要

エンコーダ入力ボード用のデータ構造体

2.2 CM01_INPUT_SENSOR 型構造体

定義

```
typedef struct _CM01_INPUT_SENSOR
{
    WORD wSensor[8];        センサ入力のA/D変換値.
} CM01_INPUT_SENSOR;
```

概要

センサ入力ボード用のデータ構造体

2.3 CM01_OUTPUT_PWM 型構造体

定義

```
typedef struct _CM01_OUTPUT_PWM
{
    short sPwm1;    PWM出力デューティ比.
    short sPwm2;    PWM出力デューティ比.

} CM01_OUTPUT_PWM;
```

概要

PWM 出力ボード用のデータ構造体. デューティ比の設定値によって, PWM 出力基板の出力端子の状態は以下のようになります.

sPwm	DIR	EN	PWM
0	0	1	0%
1024	1	1	100%
-1024	0	1	100%
1024 以上	0	0	0%

2.4 CM01_OUTPUT_DA 型構造体

定義

```
typedef struct _CM01_OUTPUT_DA
{
    short sDA;      DA出力値.
} CM01_OUTPUT_DA;
```

概要

DA 出力ボード用のデータ構造体. DA 出力値により DA 出力基板の出力端子の状態は以下のようになります.

sDA	REF	CTRL1
0	0V	1
2047	10V	1
-2047	-10V	1
2048 以上	0V	0

2.5 CM01_PIO 型共用体

定義

```
typedef union _CM01_PIO
{
  struct{
        WORD wData;         入出力データのワード値
        WORD wDir;          入出力方向設定値のワード値
  }Word;
  struct{
        unsigned P0: 1;     入出力データのビット値
              ⋮
        unsigned P15: 1;
        unsigned DIR0:  1;  入出力方向設定値のビット値
                    ⋮
        unsigned DIR15: 1;
  } Bit
} CM01_PIO;
```

概要

PIO 基板用のデータと入出力方向設定構造体．各ポートごとに入出力方向を設定できる．データ，方向設定値ともに PIOA では 8bit，PIOB では 10bit まで有効．

3. クラス CM01

クラス CM01 はクラス ControlModule を継承しています．このためクラス ControlModule の変数や関数がメンバとなりますが，実際には使用されず無効となる変数や機能もあります．

3.1 メンバ変数

クラスConrolModuleからの継承

BYTE bPort;	CM01の接続されているポート（無効）．
BYTE bID;	CM01のID.
BYTE bStatus;	CM01本体及びパソコンのステータス．
BYTE bVersion;	ファームウェアバージョン．
char Error[256];	エラー発生時にエラーメッセージが書き込まれます．

クラスCM01独自のもの

BYTE bBoardID_PA;	PA側に接続されている入出力基板のID
BYTE bBoardID_PB;	PB側に接続されている入出力基板のID
short sInputDataSize;	入力データのサイズ
short sOutputDataSize;	出力データのサイズ
CM01_INPUT_ENC EncBoard;	エンコーダ入力基板用データ構造体
CM01_INPUT_SENSOR SensorBoard;	センサ入力基板データ構造体
CM01_OUTPUT_PWM PwmBoard;	PWM出力基板データ構造体
CM01_OUTPUT_DA DaBoard;	DA出力基板データ構造体
CM01_PIO PioBoardA;	PIOA基板データ・入出力方向構造体
CM01_PIO PioBoardB;	PIOB 基板データ・入出力方向構造体

3.2 CM01::Open

書式

BOOL Open(ControlModule* pParent, BYTE bParentPort, BYTE bModuleID);

引数

ControlModule* pParent	親モジュールのポインタ
BYTE bParentPort	接続している親モジュールのポート番号
BYTE bModuleID	CM01 の ID（0～255）

戻値

成功：TRUE

失敗：FALSE

動作

CM01 をオープンします．bModuleID に CM_MASTER_ID を指定すると，すべての ID の制御モジュールが応答します．ID が分からない場合や，制御モジュールが 1 枚しか使用しない場合に CM_MASTER_ID を設定します．

オープンの時点で接続されている入出力ボードの情報が bBoardID_PA と bBoardID_PB に入力されます．

3.3 CM01::Close

書式

BOOL Close(void);

引数

なし

戻値

成功：TRUE，失敗：FALSE

動作

CM01 をクローズします.

3.4 CM01::Reset

書式

BOOL Reset(void);

引数

なし

戻値

成功：TRUE，失敗：FALSE

動作

CM01 をリセットします. CM01 内部でソフトウェアリセットがかかります.

3.5 CM01::Ping

書式

BOOL Ping(void);

引数

なし

戻値

成功：TRUE，失敗：FALSE

動作

動作チェック用のコマンドです. パソコン側から送信した 1 バイトデータと同じデータが返ってくるかを確認します.

3.6 CM01::SetID

書式

BOOL SetID(BYTE bNewID);

引数

BYTE bNewID　　新しい ID（0～254）

戻値

成功：TRUE，失敗：FALSE

動作

CM01 に新しい ID を設定します. 新しい ID は CM01 の EEPROM に書き込まれるため，次回起動時も有効になります. ID=255(CM_MASTER_ID)は指定できません.

3.7 CM01::GetID

書式

BOOL GetID(void);

引数

なし

戻値

成功：TRUE，失敗：FALSE

動作

CM01 に設定されている ID を取得し bID に書き込みます.

3.8 CM01::GetVersion

書式

BOOL GetVersion(void);

引数

なし

戻値

成功：TRUE，失敗：FALSE

動作

CM01 のプログラムバージョンを取得し bVersion に書き込みます.
プログラムバージョンが 2.1 の場合 bVersion = 0x21 となります.

3.9 CM01::Blink

書式

BOOL Blink(BYTE bNum);

引数

BYTE bNum　　点滅回数

戻値

成功：TRUE，失敗：FALSE

動作

bNum に設定した回数 LED が点滅します．チェック時などに使用します．

3.10 CM01::SetParam

書式

BOOL SetParam(PCM_PARAM pParam);

引数

PCM_PARAM pParam　　制御パラメータ構造体のポインタ

戻値

成功：TRUE，失敗：FALSE

動作

CM01 に制御パラメータを設定します．CM01 では制御タイムアウトは使用されていませんので fCtrlTimeout の値は無効となります．

3.11 CM01::GetParam

書式

BOOL GetParam(PCM_PARAM pParam);

引数

PCM_PARAM pParam　　制御パラメータ構造体のポインタ

戻値

成功：TRUE，失敗：FALSE

動作

CM01 に設定されている制御パラメータを取得します．

3.12 CM01::Exchange

書式 1

```
BOOL Exchange( void* pOutputData, void* pInputData, BOOL RightNow=TRUE );
```

引数

void* pOutputData	出力データバッファのポインタ
void* pInputData	入力データバッファのポインタ
BOOL RightNow	送受信タイミングの指定

戻値

成功：TRUE，失敗：FALSE

動作

CM01 とデータの送受信を行います．pOutputData に出力データを格納したバッファのポインタを，pInputData に入力データバッファのポインタを設定します．送受信されるデータサイズは sInputDataSize と sOutputDataSize の値となります．引数 RightNow を省略するか TRUE を指定するとすぐに送受信が行われ，FALSE を指定すると送受信予約だけが行われます．

書式 2

```
BOOL Exchange( BOOL RightNow=TRUE );
```

引数

BOOL RightNow	送受信タイミングの指定

戻値

成功：TRUE，失敗：FALSE

動作

CM01 とデータの送受信を行います．接続されている入出力基板に合わせたデータが送受信されます．例えば，エンコーダ入力ボードと PWM 出力ボードが接続されている場合，メンバ変数の PwmBoard 内の値が出力されエンコーダのカウント値と AD 変換値が EncBoard 内の変数に入力されます．引数 RightNow を省略するか TRUE を指定するとすぐに送受信が行われ，FALSE を指定すると送受信予約だけが行われます．

3.13 CM01::ClearData

書式

```
BOOL ClearData( void );
```

引数

なし

戻値

成功：TRUE，失敗：FALSE

動作

CM01 内部のデータをクリアします．現状ではエンコーダのカウント値が 0 にクリアされます．

*ファームウェア v2.0 は未対応です．

付録

・CM_PARAM 型構造体

メンバ変数

float dt;	制御周期をミリ秒単位で指定します．デフォルトは1.0ミリ秒です．
float fCtrlTimeout;	制御タイムアウトをミリ秒単位で指定します．デフォルトは1000ミリ秒です．0でタイムアウトは無効になります．
float fCommTimeout;	パソコン側，モジュール側双方の送受信通信タイムアウトをミリ秒単位で指定します．デフォルトは1000ミリ秒です．0でタイムアウトは無効になります．

注意事項

・本ソフトウェアは，ソースコードをオープンにしておりますが，商業利用・無断転載はおやめください．また，本ソフトウェアはお客様が自由に改変して御使用いただけますが，お客様のソフトウェア改変に伴う事故や故障等に関して当方では，一切の責任を負いません．
・本ソフトウェアにより生じるお客様の製品に起因して発生したいかなる損害に対しても当方では，一切の責任を負いません．
・本ソフトウェアの仕様は改良のため予告なく変更することがあります．

改版履歴

2015 年 8 月　初版

お問い合わせはメールにてお願いいたします．

アークデバイス

E-mail:info@arcdevice.com

URL: http://www.arcdevice.com/