MDB-MIF-01

Multi Drop Bus インターフェース
シリアル通信プロトコル

１．概要

MDB-MIF-01 インターフェースは MDB / NAMA 規格バスに接続されたビルバリデータやコインメックなどのデバイス（以後、単にデバイスと呼びます）を制御するためのインターフェースであり、MDB-MIF-01 とホストの接続には２種類の方法があります。

ひとつは USB 接続の場合です。MDB-MIF-01 の USB インターフェースは、一般的な PC に接続すると、USB-Serial コンバータとして認識されるようになっています。その為、USB- Serial インターフェースを認識可能な PC に USB 接続した場合は、シリアルポートとして直接コマンドを発行することにより制御することが可能です。この場合、MDB-MIF-01 は仮想 COM ポートとして動作していますので、ボーレートなどのシリアル設定は無視されます。

もう一つは、MDB-MIF-01 の RS232C シリアルポートを使ってインターフェースする方法です。この方法は USB インターフェースを持っていないワンチップマイコンなどから機械語あるいは C 言語などで制御する場合に便利です。しかし通常の PC からこの方法を用いる場合もあります。例えば、OS によっては USB の仮想 COM ポートがサポートされておらず、BVCM-MIF-02 が認識出来ない場合があります。この場合には通常のシリアルポートあるいは市販の USB シリアルインターフェースを経由して接続することが可能です。

PC 接続の場合で、OS が Windows の場合は専用 API として DLL が用意されています。この DLL を使用すると専用 API を用いて簡単にデバイスに対しコマンドを発行することが出来ます。DLL を呼び出せる言語であればどのような言語（Visual Basic, Visual C#, Excel VBA など）からでも呼び出して使用することが可能です。（別途、Windows API リファレンス参照）

本ドキュメントでは USB による仮想 COM ポートあるいは RS232C ポート上でのシリアル通信プロトコルについて規定します。

２．シリアル通信コマンド

◇ 通信プロトコル

全てのシリアル通信は本通信プロトコルによって行われます。

a. 通信方法

- 全ての通信はホスト側（PC）からデバイス側（MDB-MIF-01）に対してコマンドを発行し、次に機器側が発行するレスポンスを受け取る方式で行われます。例外はなく、コマンドを発行すると必ずレスポンスが戻ります。

- コマンドおよびレスポンスは通信フレーム（後述）に乗せて送られるものとします。この通信フレーム内の Datagram 部分にビルバリやコインメックに対するコマンド、レスポンスが含まれます。

- 受信時に正しく受信出来なかった通信フレームは全て破棄されます。

- ホスト側が一度に発行できるコマンドは１つだけとし、レスポンス受信完了またはタイムアウト時間が経過するまで、新たなコマンドは発行出来ません。

- レスポンス受信完了してから次のコマンド発行までに、最低 20msec. のアイドルタイムが必要です。

b. シリアル通信フォーマット（USB 経由の場合、本設定は任意で良い）

通信方式： 非同期、半二重、RS485
ボーレート： 38,400 [bps]
スタートビット： 1
データビット： 8
ストップビット： 1
パリティビット： なし

c. 通信フレーム

全てのコマンド、レスポンスは本通信フレームの DATAGRAM として送受信されます。

STX	SID	RID	DATAGRAM	CRC	ETX

STX: メッセージ開始コード [02h]
SID: 送信元アドレス ID [1 Byte Binary] PC は 0x00 MDB-MIF-01 は 0x30 固定
RID: 送信先アドレス ID [1 Byte Binary] PC は 0x00 MDB-MIF-01 は 0x30 固定
DATAGRAM: データパケット（MAX: 2048Byte とする）
CRC: CRC-16-CCITT (SID から DATAGRAM まで) [2 Byte Binary HL]
ETX: メッセージ終了コード [03h]

フレーム間タイムアウト: 500[msec.]

SID から CRC までの間に STX コード 0x02 および ETX コード 0x03 が現れた場合、以下の通りエスケープします。（CRC の計算はエスケープする前に行う）

0x02 は 0x7D, 0xE2 に符号化 (STX エスケープ)
0x03 は 0x7D, 0xE3 に符号化 (ETX エスケープ)
0x7D は 0x7D, 0x5D に符号化 (制御エスケープ)

CRC-16-CCITT には各種実装がありますが、最も一般的な実装に合わせ、以下のサイトで CRC-CCITT を選択して得られるものに合わせます。（データが“123456789”の時、0x29B1）
http://www.zorc.breitbandkatze.de/crc.html

CRC-16-CCITT`の実装例（C 言語および Visual Basic による）を次頁に示します。

CRC-16-CCITT（C 言語による実装例）

```
#ifdef ROM_TABLE
/*****************************************************************
*   ROM Table
*****************************************************************/
        const unsigned short CrcTable[256] =
{
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
    0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
    0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,
    0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
    0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
    0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
    0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
    0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
    0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
    0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
    0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,
    0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
    0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,
    0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,
    0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
    0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,
    0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
    0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,
    0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
    0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,
```

```
    0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
    0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,
    0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0 };
#else
    unsigned short CrcTable[256];
#endif

/*****************************************************************
* Function: void crc_create_table(void);
* Overview: Create CRC-16 CCITT Table
* Input: none
* Output: none
*****************************************************************/
void crc_create_table(void)
{
#ifndef ROM_TABLE
#define polynomial 0x1021

    unsigned int i, j;
                unsigned short a;

    for (i = 0; i < 256; i++)
    {
        a = (i << 8);
        for (j = 0; j < 8; j++)
        {
            if (0 == (a & 0x8000)) a = (a << 1);
            else a = polynomial ^ (a << 1);
        }
        CrcTable[i] = a;
    }
#endif
}
```

```
/*********************************************************************
* Function: unsigned short crc_1byte(unsigned char in, unsigned short crc);
* Overview: Calc CRC (1 byte)
* Input: in - data
*          crc - Initial Value
* Output: crc value
*********************************************************************/
            unsigned short crc_1byte(unsigned char in, unsigned short crc)
{
    return CrcTable[((crc >> 8) ^ in) & 0xFF] ^ (crc << 8);
}

/*********************************************************************
* Function: unsigned short crc_block(unsigned char buff[], unsigned short size, unsigned crc);
* Overview: Calc CRC (block)
* Input: in - data
*          crc - Initial Value  （初期値は通常、0xFFFF を設定して呼び出します。）
* Output: crc value
*********************************************************************/
            unsigned short crc_block(unsigned char buff[], unsigned short size, unsigned short crc)
{
                unsigned short pos;

    for(pos=0; pos!=size; pos++)
    {
        crc = crc_1byte(buff[pos], crc);
    }

    return crc;
}
```

CRC-16-CCITT（Visual Basic による実装例）

```
Private CRC_CCITT_TABLE(255) As Integer

Private Sub CreateCRCTable()
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer
    Dim Value As Integer
    For i = 1 To 256
        j = i * &HFF
        Value = &H0
        For k = 0 To 7
            If ((Value Xor j) And &H8000) = &H8000 Then
                Value = ((Value * 2) Xor &H1021) And &HFFFF
            Else
                Value = (Value * 2) And &HFFFF
            End If
            j = j * 2
            CRC_CCITT_TABLE(i - 1) = Value
        Next
    Next
End Sub

Private Function CRC16_Calculate(ByRef bytes() As Byte, nmbr As Integer) As Integer
    Dim CRC_Value As Integer = &HFFFF
    Dim i As Integer
    For i = 0 To nmbr - 1
        Dim Value As Integer = (CRC_Value >> 8) Xor bytes(i)
        CRC_Value = ((CRC_Value << 8) And &HFFFF) Xor CRC_CCITT_TABLE(Value)
    Next
    Return CRC_Value
End Function
```

◇ コマンド、レスポンス

a. コマンド形式

＜command＞＜data＞

＜command＞は1バイトのMDBアドレス（コマンド）に対応します。（詳細はMDB/ICP規格仕様書参照）ただし0x00－0x07まではMDB-MIF-01専用のコマンドとなり以下が定義されています。

0x01: バージョン情報要求

0x02: バスリセット

バスリセットに対するレスポンスはありません。また本コマンドを発行後、実際にデバイスがリセットされるまでには数秒間かかります。この間は他のコマンドを受け付けません。この間に送られる他のコマンドに対してはレスポンス（ＢＵＳＹ）が返ります。

＜data＞はMDBアドレス（コマンド）に引き続き送出されるデータ列で、コマンドに依存する複数バイトのデータです。MDB-MIF-01はコマンドに続き渡される＜data＞部分をそのままMDBバスに送出します。

b. レスポンス形式

MDB-MIF-01はコマンドをMDBバスに送出すると、次にデバイスから返ってくるレスポンスを本レスポンス形式で返します。

＜response＞

＜response＞ デバイスからのレスポンス（複数バイトのデータ）です。MDB-MDB-01はデバイスから受信したデータフレームをそのまま＜response＞として返します。

c. 例外動作

i. MDB-MIF-01 は、CRC 計算が合わないなど通信フレームに異常があった場合は直ちにエラーを返します。

ii. MDB-MIF-01 はデバイスとの通信に異常があった場合、自動的にリトライ処理を行い、データ転送を完了させます。

iii. MDB-MIF-01 はデバイスとの通信試行で応答がなかった場合、デバイスタイムアウトを返します。