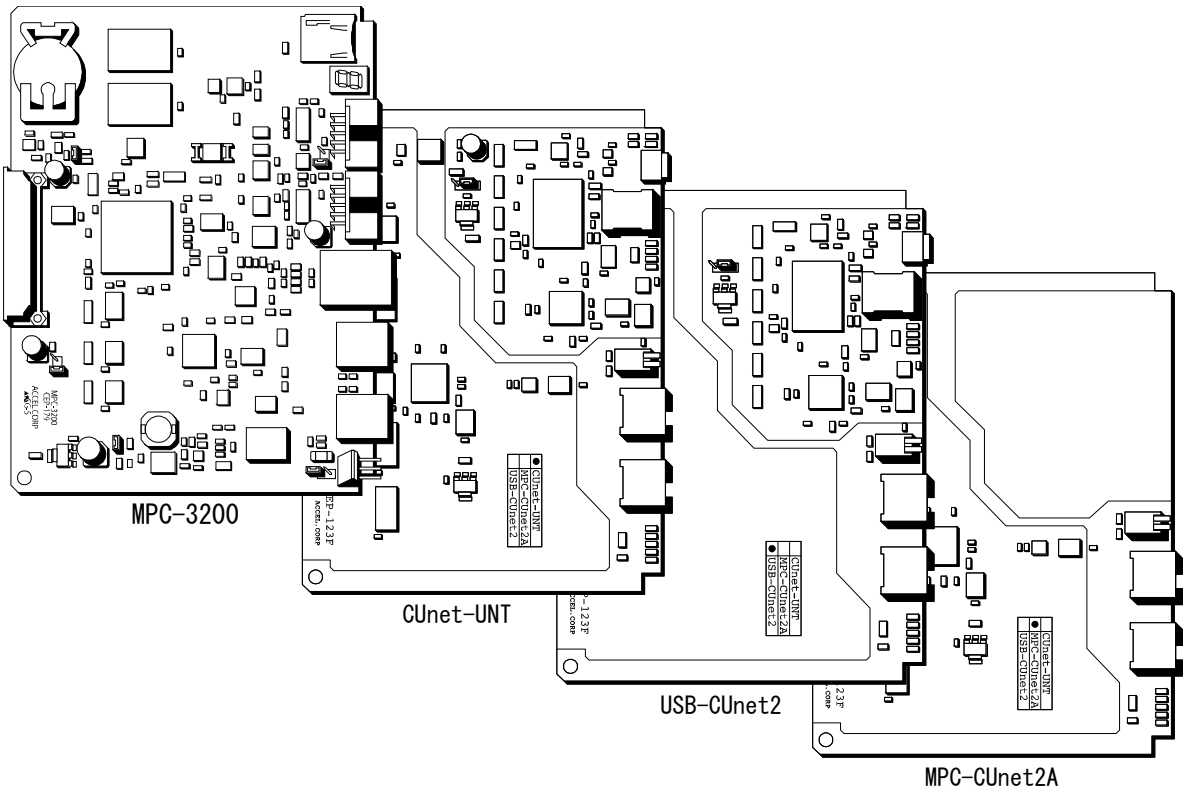
CUnet 機器 製品別マニュアル

ACCEL 240131



目次

CUnet とは

CUnet は、ステップテクニカ社のオリジナル通信方式であり、シリアル通信回線で接続された複数

の拠点間で、同一メモリ空間をリアルタイムで共有するリモートメモリ共有技術です。
　USB-CUnet は CUnet デバイス「MKY43」を搭載、パソコンと CUnet 機器間でメモリの共有を実現します。

USB-CUnet の概要

　Cunet のグローバルメモリは 512byte です。8byte を 1 ブロックとして、64 ブロックを SA0～SA63 として管理します。
　各 CUnet ステーションは、すべてのエリアの値を読むことができます。対して、書き込みエリアは、開始ブロックのステーションアドレス(sa)と占有ブロック数(ow)で指定します。usbcunet.dll の初期化コマンドでは以下のようにします。

```
cunet_init 5 4 31     'USB-CUnet 初期化
   5     sa = 開始ブロック          SA5～
   4     ow = 占有ブロック数        4 ブロック占有する。(SA8 まで)
   31    en = 共有最終ブロック      共有エリアを 31 までに限定する
```

　この例では、SA5～SA8 が書き込み可能となります。
　このエリアに対応するアドレスは、IN/OUT(※)では 2040～2071 です。このため、OUT コマンドでは、OUT dat 2043 などと使用することができます。ON/OFF では、2320～2575 が対応することになります。
　IN(),SW()はどのステーションからでも使用することができます。
　共有ブロックとは、MKY43 が常時通信により、メモリを同期化させるブロック数です。ここでは、31 としているため、SA32～SA63 は、同期しません。

　同期エリアを制限すると、全体が同一データになるまでの時間が短縮できます。
　512byte 64 ブロックすべてを共有する場合で約 2.5msec
　256byte 32 ブロックでは、1.2msec
　16byte 2 ブロックでは 102μ 秒となります。 必要に応じて調整します。

※IN,OUT,SW,ON,OFF は MPC の言語表現です。usbcunet.dll では cunet_in, cunet_out, cunet_sw, cunet_on, cunet_off が対応します。

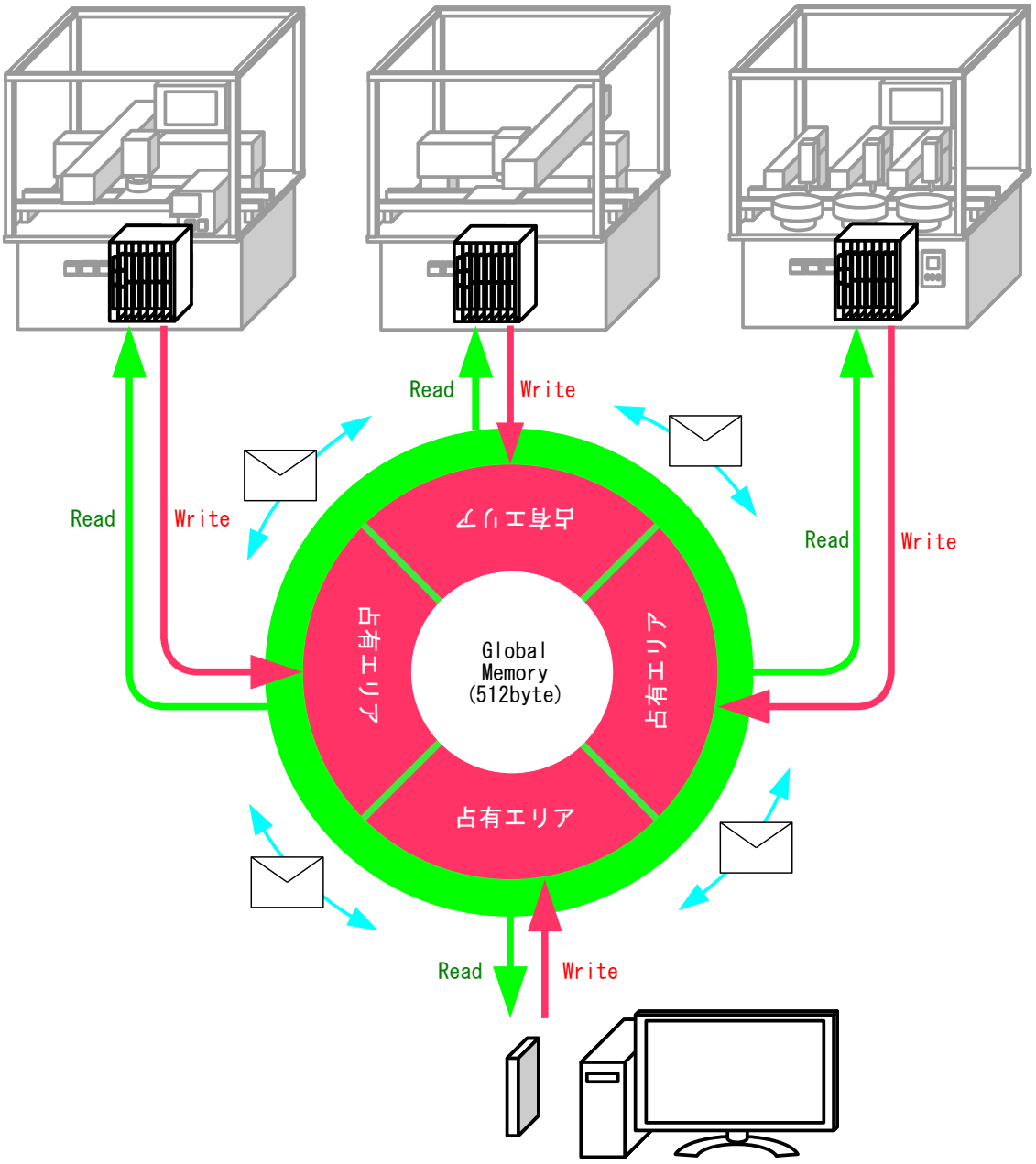
※CUnet を使って MPC のプログラムの読み込み・保存・編集はできません。

ブロックイメージ

SA	IN/OUT バンク番号	ON/OFF/SW ビット番号	例）cunet_init 5,4,31 と設定すると (MPC コマンドでは CUNET 5,4,31)
0	2000～2007	2000～2063	
1	2008～2015	2064～2127	
2	2016～2023	2128～2191	
3	2024～2031	2192～2255	
4	2032～2039	2256～2319	
5	2040～2047	2320～2383	＜占有開始　sa=5 ┐
6	2048～2055	2384～2447	占有数 ow=4
7	2056～2063	2448～2511	書込可能
8	2064～2071	2512～2575	＜ ここまで占有 ┘
9	2072～2079	2576～2639	
10	2080～2087	2640～2703	
11	2088～2095	2704～2767	
12	2096～2103	2768～2831	
13	2104～2111	2832～2895	
14	2112～2119	2896～2959	
15	2120～2127	2960～3023	読込は全ブロック可能
16	2128～2135	3024～3087	
17	2136～2143	3088～3151	
18	2144～2151	3152～3215	
19	2152～2159	3216～3279	
20	2160～2167	3280～3343	
21	2168～2175	3344～3407	
22	2176～2183	3408～3471	
23	2184～2191	3472～3535	
24	2192～2199	3536～3599	
25	2200～2207	3600～3663	
26	2208～2215	3664～3727	
27	2216～2223	3728～3791	
28	2224～2231	3792～3855	
29	2232～2239	3856～3919	
30	2240～2247	3920～3983	
31	2248～2255	3984～4047	＜ 共有最終 en=31
32	2256～2263	4048～4111	
33	2264～2271	4112～4175	
34	2272～2279	4176～4239	
35	2280～2287	4240～4303	
36	2288～2295	4304～4367	
37	2296～2303	4368～4431	
38	2304～2311	4432～4495	
39	2312～2319	4496～4559	
40	2320～2327	4560～4623	
41	2328～2335	4624～4687	
42	2336～2343	4688～4751	
43	2344～2351	4752～4815	
44	2352～2359	4816～4879	
45	2360～2367	4880～4943	
46	2368～2375	4944～5007	
47	2376～2383	5008～5071	
48	2384～2391	5072～5135	
49	2392～2399	5136～5199	
50	2400～2407	5200～5263	
51	2408～2415	5264～5327	
52	2416～2423	5328～5391	
53	2424～2431	5392～5455	
54	2432～2439	5456～5519	
55	2440～2447	5520～5583	
56	2448～2455	5584～5647	
57	2456～2463	5648～5711	
58	2464～2471	5712～5775	
59	2472～2479	5776～5839	
60	2480～2487	5840～5903	
61	2488～2495	5904～5967	
62	2496～2503	5968～6031	
63	2504～2511	6032～6095	

※巻末に白表があります。I/O 割付時などにご利用ください。

メモリー共有イメージ



512 バイトのグローバルメモリを共有します。各ステーションは全エリアを読むことができますが、書き込みは自分が占有したエリアだけです。

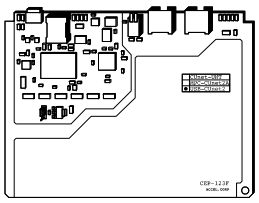
グローバルメモリとは別に送受信 256 バイトのメールバッファが有り、任意のステーション間でメール転送ができます。

ハードウェア

■ ボードの種類

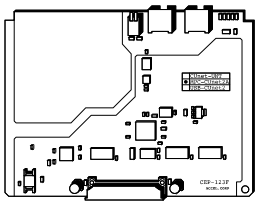
ベース基板は同じですが、実装の異なる 3 形態があります。

- USB-CUnet2



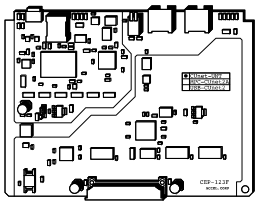
PC との I/F に用います。
従来の USB-CUnet 相当です。
CASE-1S に収納できます。

- MPC-CUnet2A



ラックに挿入して、他のステーションとリンクします。
従来の MPC-CUnet2 相当です。

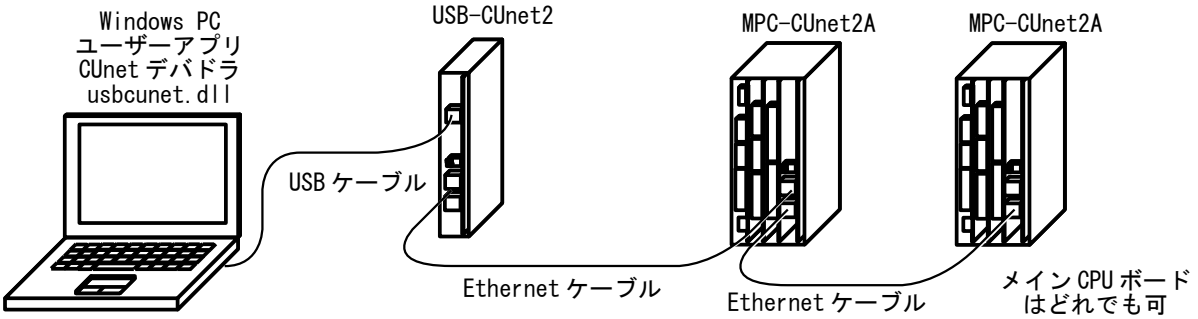
- CUnet-UNT



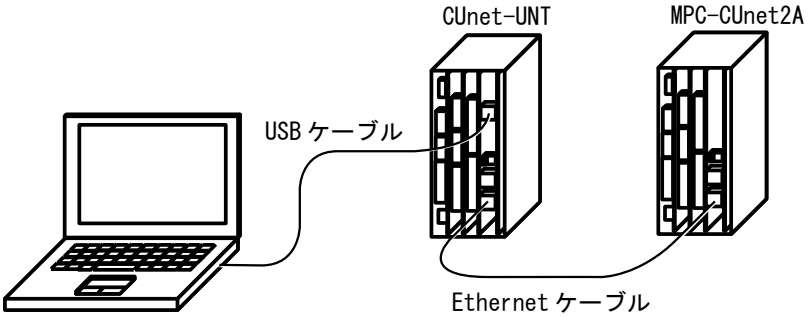
ラックに挿入します。
USB-CUnet2 と MPC-CUnet2A の両方の機能を持ちます。

■ パソコンとリンクする場合の組み合わせ例

- PC と MPC が離れている場合は USB-CUnet2 を PC の近くに置き Ethernet ケーブルで延長します。従来の USB-CUnet を使用した構成と同等です。

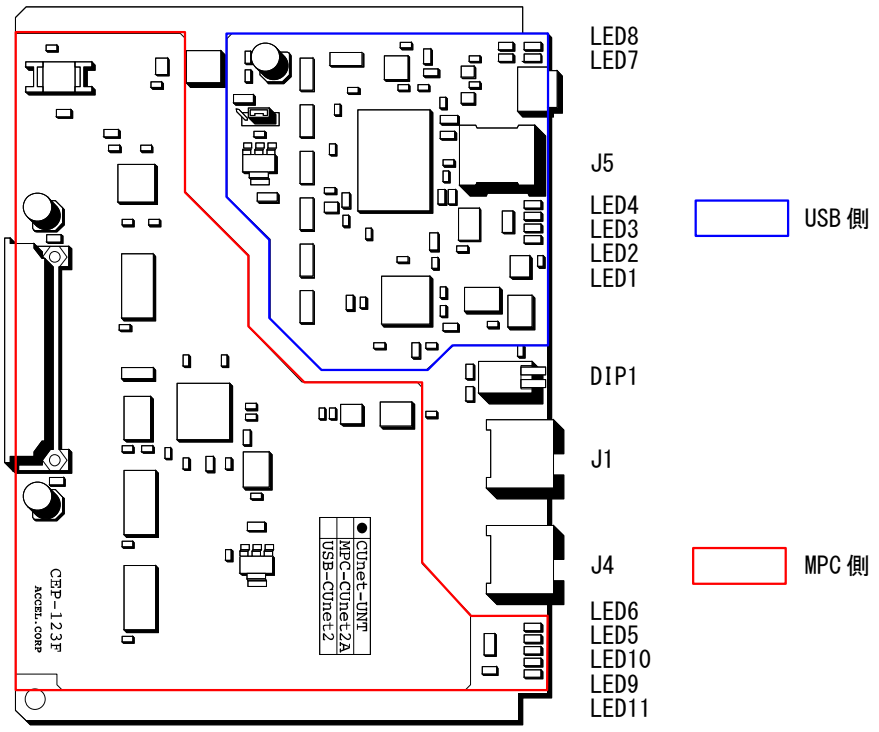


- PC と MPC が近い場合は CUNET-UNT を用いて直結することができます。



どちらの場合も USB ケーブルは Max5m で品質、ノイズの混入、引き回しに配慮してください。

■ 外観



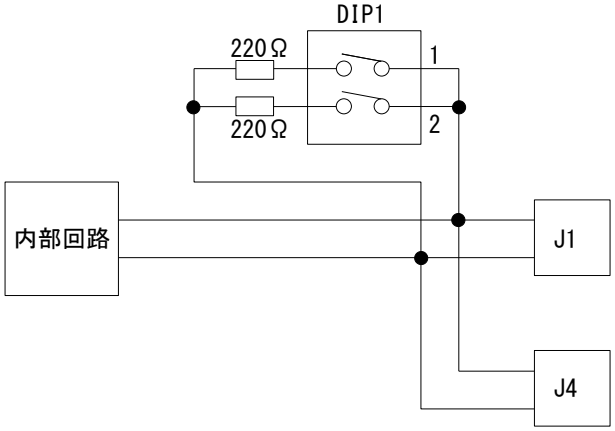
図は CUnet-UNT です。
USB-CUnet2 には MPC 側の実装がありません。
MPC-CUnet2A には USB 側の実装がありません。

コネクタ、終端抵抗

J1, J4	モジュラーコネクタ	RJ-45 8 極。他局間 CUnet 接続
J5	USB コネクタ	USB-B タイプ 4 芯メス。パソコン間 USB 接続
DIP1	ディップスイッチ	終端抵抗設定。抵抗無し/200Ω/100Ω
USB 側 LED		
LED1 (R)	メンバー減少表示 (MKY43 #MCARE)	メンバステータスが減少したときに、約 50ms 点灯。1 サイクルが 50ms 以内の場合、数サイクルに渡って 1 つのパルスが出力されることになります。
		点灯の可能性： 外来ノイズによる通信障害。
LED2 (R)	リンク切れ表示 (MKY43 #LCARE)	一旦リンクが成立していた装置が、リンク不成立になったことを検出した時に、約 50ms 点灯。1 サイクルが 50ms 以内の場合、数サイクルに渡って 1 つのパルスが出力されることになります。
		点灯の可能性：(正常に共有動作を継続しているステーションで点灯) 通信ケーブルの断線、コネクタの外れ、装置の離脱、装置の電源断、ドライバ/レシーバ部品の故障。 初期化(CUNET, init_cunet)の不適合。
LED3 (G)	サイクル開始通知 (MKY43 UTY1)	サイクルの先頭で毎回パルスで通知します。 (※パルスが短くて LED では認識できません)
LED4 (G)	リンク成立表示 (MKY43 #MON)	メンバステータスを持ったステーション(装置)が存在している時に点灯。(他装置とのメモリ共有がコンスタントにできていることを示す)
LED7 (G)	CPU(CY7C68013)動作	USB-CUnet ファームウェア動作中点滅(約 0.3 秒の一定周期)
LED8 (G)	USB アクセス	PC が USB アクセス中点灯
MPC 側 LED		
LED5 (G)	サイクル開始通知 (MKY43 UTY1)	(表示内容は上記と同じ)
LED6 (G)	リンク成立表示 (MKY43 #MON)	
LED9 (R)	メンバー減少表示 (MKY43 #MCARE)	
LED10 (R)	リンク切れ表示 (MKY43 #LCARE)	
LED11 (G)	電源	内部動作電源(DC3V)

正常に動作していれば LED7:点滅、LED8:アクセス中点灯、LED4:点灯、LED6:点灯、LED11:点灯

■ 終端抵抗



両端のステーションは終端抵抗を設定して下さい(出荷時 DIP1 は 1,2 とも OFF(抵抗無し))
稼動中 LED1,2,9,10 が点灯する場合は通信異常です。終端抵抗の設定で正常化する場合があります。

※参考：「CUnet テクニカルガイド（ネットワーク用）」
・通信ケーブルにおける終端処理の原理
・終端抵抗の具体的な接続
など

■ 通信仕様

- CUnet ステーション間
  RS-485(SN75LBC176D、パルストランスアイソレート)
  転送レート 12Mbps
  Ethernet 通信ケーブル　10BASE-T、Cat3 以上、かつ一括シールドの通信ケーブルを推奨

  ※参考：「CUnet テクニカルガイド（ネットワーク用）」
  ・通信ケーブルの選択
  など

- PC/USB-Cunet 間
  USB2.0

■ CUnet のパケット受信時の検定

Cunet 専用 IC には、パケットを受信する際に以下の 3 つの検定を重畳して実行することが義務付けられています。これにより一般的な通信によって生じがちなデータ化けの現象は、全く発生しません。

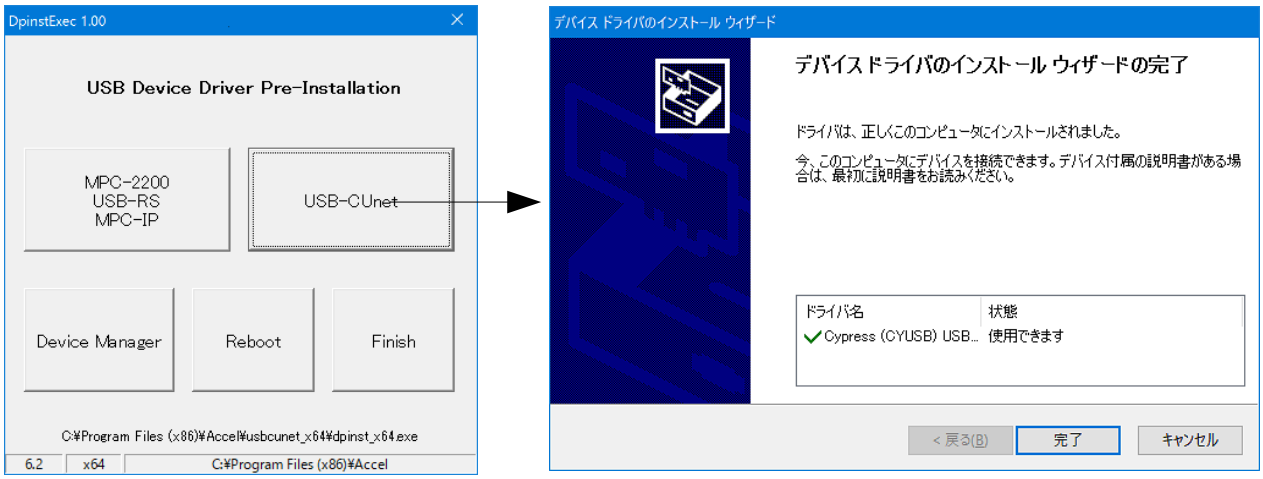
- フォーマット（構成形式）検定：パケットを受信し終えた瞬間に、特定のスタートパターンから始まるパケットの構成形式が CUnet プロトコルの規約に適合しているかを検定します。
- CRC-16 検定：一般にイーサネット LAN や大容量通信においては、数千～数万ビットの配列に対して 1 つの CRC-12 ブロックチェックコード（BCC）が利用されています。これに対して CUnet プロトコルは、1 つのパケット（数百～数千ビットの配列）に付き CRC-12 よりも検定率の高い CRC-16 ブロックチェックコード（BCC）を採用しています。このブロックチェックコードは、パケットを受信し終えた瞬間に検定されます。
- RZ 信号形式の正当性検定（RZ 検定）：シリアル通信信号のパケットは、“4.1 時間の基準単位”に記述された RZ（Return to Zero：“マンチェスタ符号”とも呼ばれる）のパルス配列から構成されています。CUnet プロトコルにおいては、受信するパケットの信号が RZ の形式を保っているかを 1 ビット単位毎に検定します。

  ※参考：「CUnet 導入ガイド　（CUnet プロトコル基本解説）」
  ・データの品質保証
  など

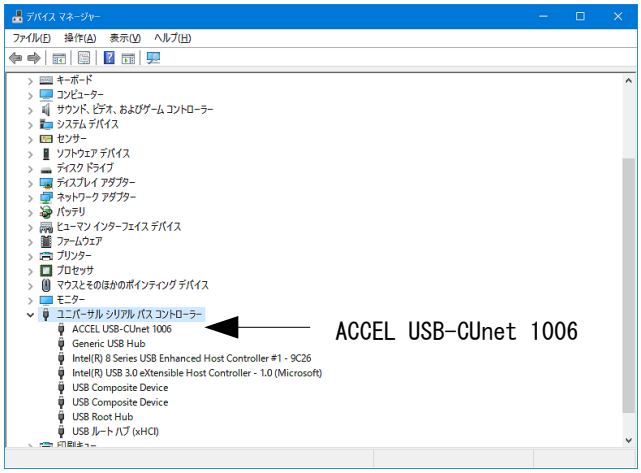
PC ソフトウェア

■ PC のデバイスドライバ インストール

デバイスドライバ、Dll、ツールは「開発ツールダウンロード」のインストーラ Accel_Setup_jpn.msi でセットアップされます。Accel_Setup_jpn.msi 終了時に現れる「USB Device Driver Pre-Installation」の USB-CUnet ボタンをクリックしてデバイスドライバをプリインストールして下さい。



その後、USB-CUnet2 を接続すると PC がドライバをセットアップします。
正常にセットアップが完了するとデバイスマネージャに「ACCEL USB-CUnet 1006」が現れます。



※デバイスドライバのプリインストールはいつでもできます。ACCEL フォルダの DpinstExec.exe を実行して下さい。

■ 関係ファイル

インストーラにより C:\Program Files (x86)\Accel または C:\Program Files\Accel に下記のサブフォルダが作成されます。

usbcunet_x64	64 ビット Windows7 以降用 デバイスドライバ設定フォルダ
usbcunet_x86_wlh	32 ビット Windows7 以降用 デバイスドライバ設定フォルダ
usbcunet_x86_wxp	32 ビット WindowsXP 用 デバイスドライバ設定フォルダ

各フォルダには次のファイルが入ってします。これらのファイルはインストール時に所定のフォルダに配置されるので何もする必要はありませんが、usbcunet.dll は実行アプリケーション(EXE)と同じフォルダにコピーすることによりアプリケーションからアクセスできます。

Cyload1.spt	ファームウェアスクリプトファイル
cyusb.inf	設定ファイル
CYUSB.SYS	デバイスドライバ
usbcunet.dll	インターフェースライブラリ(全機種共通)

usbcunet.dll 関数

※usbcunet.dll は 32 ビットアプリケーションです。

■ 初期化、各種情報

- **cunet_usb_open()**

機能

USB オープン

引数

無し

戻値

0:失敗(デバイス認識不能)
1:成功
2:失敗(HighSpeed 接続不能(EHCI では無いと思われる))

使用例

```
If cunet_usb_open() <> 1 Then
   MsgBox("open error")
   Exit Sub
End If
```

- **cunet_init(sa, ow, en)**

機能

初期化

引数

sa:占有するブロックの先頭 SA(Station Address)番号。
例) SA0～SA3 を占有するなら sa=0,SA12～SA15 なら sa=12,sa=255 とすると MKY43 Reset
ow:占有するブロック数。
例) SA0～SA3 を占有するなら ow=4,SA12～SA15 も ow=4
en:共有する最終ブロック。
例) SA0～SA15 を共有するなら en=15

戻値

無し

使用例

```
cunet_init(255, 0, 0)    'USB-CUnet Reset (これはグローバルメモリをクリアしません)
Sleep(500)
cunet_init(0, 4, 7)      'USB-CUnet Initialize (これでグローバルメモリをクリアします)
```

・USB-CUnet 通信レートの変更方法
MPC-CUnet2～USB-CUnet の通信レートは 12Mbps です。上記の設定で 12Mbps に設定されますが、MPC 以外の機器で 3Mbps または 6Mbps に変更するには次のように行います。

```
cunet_init(&H80 + sa, ow, en)    ' 6Mbps
cunet_init(&H40 + sa, ow, en)    ' 3Mbps
```

使用例

```
cunet_init(255, 0, 0)
Sleep(500)
cunet_init(&H80 + 4, 4, 7)       ' rate=6M,SA=4,ow=4,en=7
```

- **cunet_fw_vid()**

機能

VID 取得

引数

無し

戻値

VID 番号

使用例

```
Label_fw_vid.Text = "FW VID: " + Hex$(cunet_fw_vid())
```

- **cunet_fw_pid()**

機能

PID 取得

引数

無し

戻値

PID 番号

使用例

```
Label_fw_pid.Text = "FW PID: " + Hex$(cunet_fw_pid())
```

- **cunet_fw_ver()**

機能
　ファームウェアのバージョン番号取得
引数
　無し
戻値
　バージョン番号
使用例
```
Label_fw_ver.Text = "FW VER: " + CStr(cunet_fw_ver())
```

- **cunet_dll_ver()**

機能
　DLL のバージョン番号取得
引数
　無し
戻値
　バージョン番号
使用例
```
Label_dll_ver.Text = "DLL VER: " + CStr(cunet_dll_ver())
```

■ グローバルメモリ Read/Write

- **cunet_in(adr, siz)**

機能
　パラレル読み込み(全てのブロックを参照可能)
引数
　adr:バンク番号(2000~)
　siz:読み込みサイズ(1:1byte 符号無、 2:2Byte 符号付、 4:2Byte 符号無、 8:4Byte 符号付)
戻値
　パラレル入力値
使用例
```
Label_in.Text = cunet_in(TextBox_in.Text, 1)
```

- **cunet_out(dat, adr, siz)**

機能
　パラレル書き込み(占有したブロックのみ)
引数
　dat:書き込みデータ
　adr:バンク番号(2000~)
　siz:書き込みサイズ(1:1byte 符号無、 2:2Byte 符号付、 4:2Byte 符号無、 8:4Byte 符号付)
戻値
　無し
使用例
```
cunet_out(TextBox_out_dt.Text, TextBox_out_bnk.Text, 1)
```

- **cunet_sw(adr)**

機能
　ビット読み込み(全てのブロックを参照可能)
引数
　adr:ビット番号(2000~)
戻値
　0:オフ
　1:オン
使用例
```
Label_sw.Text = cunet_sw(TextBox_sw.Text)
```

- **cunet_on(adr)**

機能
　ビットオン(占有したブロックのみ)
引数
　adr:ビット番号(2000~)
戻値
　無し
使用例
```
cunet_on(TextBox_on_off.Text)
```

- **cunet_off(adr)**

機能
　ビットオフ(占有したブロックのみ)
引数
　adr:ビット番号(2000~)
戻値
　無し
使用例
```
cunet_off(TextBox_on_off.Text)
```

- **cunet_in_blk(stadr, length, ar)**

機能
　グローバルメモリ バルク読み込み(全てのブロックを参照可能)
引数
　stadr:読み込み開始バンク番号(2000~)
　length:読み込み長(バイト)
　ar:格納配列
戻値
　無し
使用例
```
Dim ar(512) As Integer
cunet_in_blk(2000, 512, ar(0))
```

- **cunet_out_blk(stadr, length, ar)**

機能
　グローバルメモリ バルク書き込み(占有したブロックのみ)
引数
　stadr:書き込み開始バンク番号(2000~)
　length:書き込み長(バイト)
　ar:書込む配列
戻値
　無し
使用例
```
Dim ar(512) As Integer
cunet_out_blk(2000, 512, ar(0))
```

■ MPC間メール転送

CUnetのメール機能を使いPC～MPC間で点・MBK・I/Oエリアのデータ転送を行いなます。MPCはCU_POSTコマンドを実行します。

- **cunet_post_pnt(dest_sa, ar_top, send_ar)**

機能
　MPC-CUnetステーションへ点データを送信
引数
　dest_sa:送信先SA
　ar_top :送信先のP()先頭番号
　send_ar:送信する配列
戻値
　-1:MSCレジスタSENDが下がらない
　0以上:MESR(Mail Error Status Register)の内容(0なら正常)
使用例
```
Dim ar(60) As Integer
res = cunet_post_pnt(4, 1000, ar(0))
```
注意 MPCの点データにはXYZUの順番で入ります
```
PC側:            ar(0)    ar(1)    ar(3)    ar(2)
                   ↓        ↓        ↓        ↓
MPC側:  P(1000) X=20000 Y=20001 U=20003 Z=20002
```

- **cunet_post_mbk(dest_sa, ar_top, send_ar)**

機能
　MPC-CUnetステーションへMBKデータを送信
引数
　dest_sa:送信先SA
　ar_top :送信先のMBK()先頭
　send_ar:送信する配列
戻値

-1:MSC レジスタ SEND が下がらない
0 以上:MESR(Mail Error Status Register)の内容(0 なら正常)

使用例

```
Dim ar(120) As Integer
res = cunet_post_mbk(4, 5000, ar(0))
```

- **cunet_post_mbk_lng(dest_sa, ar_top, send_ar)**

機能

MPC-CUnet ステーションへ MBK データをロング(4byte 長)で送信

引数

dest_sa:送信先 SA
ar_top :送信先の MBK()先頭
send_ar:送信する配列

戻値

-1:MSC レジスタ SEND が下がらない
0 以上:MESR(Mail Error Status Register)の内容(0 なら正常)

備考

タッチパネルのデータ長をロングにした場合に使用

使用例

```
Dim ar(60) As Integer
res = cunet_post_mbk_lng(4, 5000, ar(0))    → 対応する MPC の関数は MBK(5000~Lng) など
```

- **cunet_post_io(dest_sa, ar_num, send_data)**

機能

MPC-CUnet ステーションへ I/O データを送信

引数

dest_sa :送信先 SA
ar_num   :送信先の I/O バンク番号
send_data:送信する変数

戻値

-1:MSC レジスタ SEND が下がらない
0 以上:MESR(Mail Error Status Register)の内容(0 なら正常)

備考

ar_num の 1Byte に書き込みます。負の数値を入れるとメモリー I/O へ書き込みます。

使用例

```
res = cunet_post_io(4, -1, dt)
```

- **cunet_post_io_blk(dest_sa, ar_top, send_ar)**

機能

MPC-CUnet ステーションへ I/O データを送信

引数

dest_sa:送信先 SA
ar_top :送信先の I/O 先頭バンク番号
send_ar:送信する配列

戻値

-1:MSC レジスタ SEND が下がらない
0 以上:MESR(Mail Error Status Register)の内容(0 なら正常)

備考

ar_top から 240Byte 連続で書き込みます。負の数値を入れるとメモリー I/O に書き込みます。

使用例

```
Dim ar(0 To 239) As Integer
res = cunet_post_io_blk(4, -1, ar(0))
```

- **cunet_post_pnt_str(dest_sa, ar_top, send_str)**

機能

MPC-CUnet ステーションへ PNT 文字列を送信

引数

dest_sa :送信先 SA
ar_top :送信先の P()先頭
send_str:送信する文字列(max239 キャラクタ)

戻値

-1:MSC レジスタ SEND が下がらない
0 以上:MESR(Mail Error Status Register)の内容(0 なら正常)

使用例

```
Dim s As String
Dim res As Integer
s = TimeOfDay() + vbNullChar
res = cunet_post_pnt_str(4, 2000, s)
If res <> 0 Then
```

```
        MsgBox("error res=" + CStr(res))
    End If
MPC プログラム例
    S$=P$(2000)             /* P(2000)～ 文字列読み込み
```

● cunet_post_mbk_str(dest_sa, ar_top, send_str)

```
機能
    MPC-CUnet ステーションへ MBK 文字列を送信
引数
    dest_sa :送信先 SA
    ar_top  :送信先の MBK()先頭
    send_str:送信する文字列(max239 キャラクタ)
戻値
    -1:MSC レジスタ SEND が下がらない
    0 以上:MESR(Mail Error Status Register)の内容(0 なら正常)
使用例
    Dim s As String
    Dim res As Integer
    s = TimeOfDay() + vbNullChar
    res = cunet_post_mbk_str(4, 2000, s)
    If res <> 0 Then
        MsgBox("error res=" + CStr(res))
    End If
MPC プログラム例
    S$=MBK$(2000,18)        /* MBK(2000)～ 文字列読み込み,18 文字
```

● cunet_req_pnt(req_sa, ar_top, rcv_ar)

```
機能
    MPC-CUnet ステーションへ点データを要求
引数
    req_sa:要求する SA
    ar_top:要求する P()先頭
    rcv_ar:格納する配列
戻値
    -1:MSC レジスタ SEND が下がらない。
    -2:MRC0 レジスタ RDY が立たない。
    0 以上:MESR(Mail Error Status Register)の内容(0 なら正常)
使用例
    Dim ar(60) As Integer
    res = cunet_req_pnt(4, 2000, ar(0))
注意 配列には XYZU の順番で格納されます
    MPC 側: P(2000) X=10000 Y=10001 U=10002 Z=10003
                    ↓       ↓       ↓       ↓
    PC 側:          ar(0)   ar(1)   ar(3)   ar(2)
```

● cunet_req_mbk(req_sa, ar_top, rcv_ar)

```
機能
    MPC-CUnet ステーションへ MBK データを要求
引数
    req_sa:要求する SA
    ar_top:要求する MBK()先頭
    rcv_ar:格納する配列
戻値
    -1:MSC レジスタ SEND が下がらない。
    -2:MRC0 レジスタ RDY が立たない。
    0 以上:MESR(Mail Error Status Register)の内容(0 なら正常)
使用例
    Dim ar(120) As Integer
    res = cunet_req_mbk(4, 6000, ar(0))
```

● cunet_req_mbk_lng(req_sa, ar_top, rcv_ar)

```
機能
    MPC-CUnet ステーションへ MBK データをロング(4byte 長)で要求
引数
    req_sa:要求する SA
    ar_top:要求する MBK()先頭
    rcv_ar:格納する配列
戻値
    -1:MSC レジスタ SEND が下がらない。
    -2:MRC0 レジスタ RDY が立たない。
```

0以上:MESR(Mail Error Status Register)の内容(0なら正常)

備考

タッチパネルのデータ長をロングにした場合に使用

使用例

```
Dim ar(60) As Integer
res = cunet_req_mbk_lng(4, 6000, ar(0)) ←対応するMPCのコマンドは S_MBK data 6000~Lng
```

等

- **cunet_req_io(req_sa, ar_num, rcv_data)**

機能

MPC-CUnetステーションへI/Oデータを要求

引数

```
req_sa  :要求するSA
ar_num  :要求するI/Oバンク番号
rcv_data:格納する変数
```

戻値

```
-1:MSCレジスタSENDが下がらない。
-2:MRCOレジスタRDYが立たない。
```

0以上:MESR(Mail Error Status Register)の内容(0なら正常)

備考

ar_numの1Byteを読み込みます。負の数値を入れるとメモリーI/Oを読み込みます。

使用例

```
res = cunet_req_io(4, -1, dt)
```

- **cunet_req_io_blk(req_sa, ar_top, rcv_ar)**

機能

MPC-CUnetステーションへI/Oデータを要求

引数

```
req_sa:要求するSA
ar_top:要求するI/O先頭バンク番号
rcv_ar:格納する配列
```

戻値

```
-1:MSCレジスタSENDが下がらない。
-2:MRCOレジスタRDYが立たない。
```

0以上:MESR(Mail Error Status Register)の内容(0なら正常)

備考

ar_topから240Byte連続で読み込みます。負の数値を入れるとメモリーI/Oを読み込みます。

使用例

```
Dim ar(0 To 239) As Integer
res = cunet_req_io_blk(4, -1, ar(0))
```

- **cunet_req_pnt_str(req_sa, ar_top, rcv_str, str_len)**

機能

MPC-CUnetステーションへPNT文字列を要求

引数

```
req_sa :要求するSA
ar_top :要求するP()先頭
rcv_str:格納する文字列変数(max239キャラクタ)
str_len:返された文字数
```

戻値

```
-1:MSCレジスタSENDが下がらない
```

0以上:MESR(Mail Error Status Register)の内容(0なら正常)

使用例

```
Dim s As String = Space(240)
Dim res, l As Integer
res = cunet_req_pnt_str(4, 1000, s, l)
If res <> 0 Then
    MsgBox("error res=" + CStr(res))
Else
    s = Mid(s, 1, l)
    TextBox_res.Text = s + "<eol>" + "len=" + CStr(l)
End If
```

MPCプログラム例

```
TM$="now "+HEX$(TIME(0))
SETP 1000 TM$                  /* P(1000)～  文字列書き込み
```

- **cunet_req_mbk_str(req_sa, ar_top, rcv_str, str_len)**

機能
MPC-CUnet ステーションへ MBK 文字列を要求
引数
req_sa :要求する SA
ar_top :要求する MBK()先頭
rcv_str:格納する文字列変数(max239 キャラクタ)
str_len:返された文字数
戻値
-1:MSC レジスタ SEND が下がらない
0 以上:MESR(Mail Error Status Register)の内容(0 なら正常)
使用例

```
Dim s As String = Space(240)
Dim res, l As Integer
res = cunet_req_mbk_str(4, 1000, s, l)
If res <> 0 Then
    MsgBox("error res=" + CStr(res))
Else
    s = Mid(s, 1, l)
    TextBox_res.Text = s + "<eol>" + "len=" + CStr(l)
End If
```

MPC プログラム例

```
TM$="now "+HEX$(TIME(0))
S_MBK TM$ 1000 12              /* MBK(1000)～ 文字列書き込み,12 文字
```

■ MKY43 レジスタ Read/Write

- **cunet_peek(adr)**

機能
MKY43 レジスタ読み込み
引数
adr:MKY43 レジスタアドレス
戻値
adr で指定されたレジスタの値(1byte)

- **cunet_peek_Lng(adr)**

機能
MKY43 レジスタ読み込み
引数
adr:MKY43 アドレス
戻値
adr で指定されたレジスタの値(4byte)

- **cunet_poke(dat, adr)**

機能
MKY43 レジスタ書き込み
引数
dat:レジスタへ書き込むデータ(1byte)
adr:MKY43 レジスタアドレス
戻値
無し

- **cunet_poke_Lng(dat, adr)**

機能
MKY43 レジスタ書き込み
引数
dat:レジスタへ書き込むデータ(4byte)
adr:MKY43 アドレス
戻値
無し

- **cunet_poke_msb(ar)**

機能
MKY43 Mail Send Buffer に 256 バイト書込み
引数
ar:書込む配列
戻値
無し

- **cunet_peek_mrb0(ar)**

機能
MKY43 Mail Receive Buffer0 から256バイト読み込み
引数
ar:格納配列
戻値
無し

- **cunet_peek_mrb1(ar)**

機能
MKY43 Mail Receive Buffer1 から256バイト読み込み
引数
ar:格納配列
戻値
無し

- **cunet_chk_mfr(sa)**

機能
MFR(Member Flag Register)のSAビットチェック
引数
sa:SA番号 0～63
戻値
1:SA有り(リンク成立)
0:SA無し(リンク不成立)
備考
MFRにはSA0～SA63まで個々のメンバの状態が格納されています。
使用例
```
If cunet_chk_mfr(4) = 0 Then      'SA4の有無を確認
    MsgBox("相手SAがない")
End If
```

- **cunet_chk_run()**

機能
SCR(System Control Register)のRUN phaseビットチェック
引数
無し
戻値
1:RUNフェーズである
0:RUNフェーズでない
備考
ボード上のLED4(緑)と連動します。
使用例
```
If cunet_chk_run = 0 Then
    MsgBox("RUNフェーズではない")
End If
```

- **cunet_chk_mcare()**

機能
CCTR(Care CounTer Register)のMCC0～7(Member Care Counter)をリード&セット
引数
無し
戻値
MCC値(最高0xff)
備考
MCARE発生時はボード上のLED1(赤)が約50ms点灯します。
使用例
```
c = cunet_chk_mcare             ' 読み込み後カウンタを0にする
If c <> 0 Then
    MsgBox("MCARE発生" + CStr(c))
End If
```

- **cunet_chk_lcare()**

機能

CCTR(Care CounTer Register)のLCC0～7(Link Care Counter)をリード&セット

引数

無し

戻値

LCC値(最高0xff)

備考

LCARE発生時はボード上のLED2(赤)が約50ms点灯します。

使用例

```
c = cunet_chk_lcare            ' 読み込み後カウンタを0にする
If c <> 0 Then
    MsgBox("LCARE発生" + CStr(c))
End If
```

Visual Basic 参照宣言例

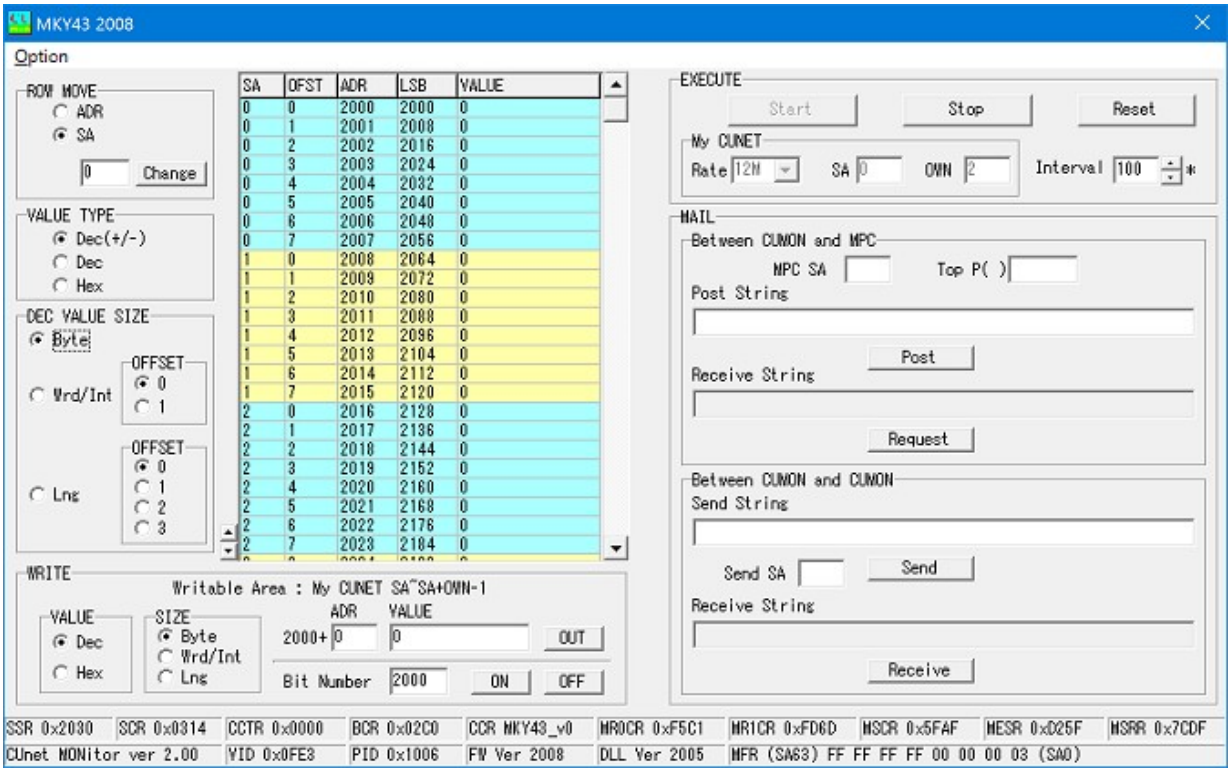
コメント付きの宣言モジュールをホームページからダウンロードできます。
セマフォ付き関数宣言とセマフォ無し関数宣言がありますが、前者の使用を推奨します。

ダウンロード＞開発ツール＞No0250 usbcunet.dll Visual Basic 参照宣言例
https://accelmpc.co.jp/downloads/dev_uty.html

CUnet Monitor (CUMON.EXE)

CUMON は USB-CUnet のグローバルメモリ、メール送受信、動作状況をモニタするツールで、MPC-CUnet とのデータ授受、I/O チェック、MPC-CUnet 間の状況監視等が行えます。
(CUnet Monitor Ver2.00 以降で新旧 USB-CUnet 共通です)

■ 画面



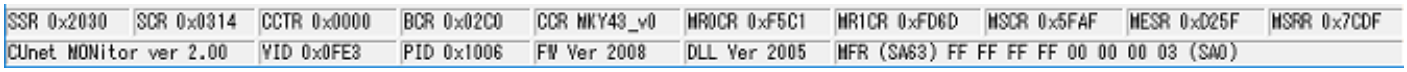
グループ	説明
EXECUTE	モニタの開始と停止
	・開始手順：「My CUNET」と「Interval」設定→「START」でモニタ実行。 ・SA：CUnet Monitor で占有する SA(ステーションアドレス)の先頭番地(0～63)。 ・OWN：占有するブロック数。 ユーザーが作成するアプリケーションの代替確認をする場合はそのアプリケーションと同じ SA、OWN を設定。グローバルメモリをモニタするだけ(例えば MPC 同士の通信監視)なら他のステーションと重複しない値を設定します。(EN は 63 固定) ・Interval：モニタサイクル(msec)。 ・Reset：USB-CUnet リセット(実行中の場合は停止)。SA、OWN を変更する場合は「Reset」→「Start」。
ROW MOVE	モニタグリッドの行を移動
	・操作手順：「ADR」(バイト単位)、「SA」(ブロック単位)を選択→番地入力→「Change」
VALUE YPE	表示する値の型を指定
	・Dec(+/-)：符号付 10 進数 ・Dec：符号無 10 進数 ・Hex：16 進数
DEC VALUE SIZE	10 進数表示時の値のサイズを指定
	・Byte：バイト ・Wrd/Int：2 バイト ・Lng：4 バイト ・OFFSET：Wrd/Int･Lng 表示時のバイトオフセット
WRITE	占有ブロックへの書込み
	・数値書き込み 「VALUE」(10 進、16 進)選択→「SIZE」(バイト、2 バイト、4 バイト)選択→「ADR」(バイト単位)入力→「VALUE」(書き込む値)入力→「OUT」 ・ビット操作 「Bit Number」入力→「ON」または「OFF」

MAIL	MKY43 のメール機能によるデータ送受信(CUMON では文字列の送受信)
	CUMON<->MPC ・対 MPC-CUnet 間メール。MPC 側は事前に CUNET 初期化、CU_POST 実行。 ・「MPC SA」(MPC 側 SA 先頭番号)と「Top P()」(メールで使用する点データの先頭番号)入力 ・送信：「Post String」に送信する文字列(240 文字以下)入力→「Post」で上記設定 MPC の点データエリアに送信。 ・受信：「Request」で上記設定 MPC の点データエリアからリクエスト&受信
	CUMON<->CUMON ・対 CUMON 間メール(別の PC との通信)。 ・送信：「Send String」に送信する文字列(256 文字以下)入力→「Send SA」(受信側 SA 先頭番号)入力→「Send」。 ・受信：「Receive」で受信バッファからメール読み込み。
モニタグリッド	グローバルメモリのモニタ
	・SA 列：ステーションアドレスの番号(8 バイトで 1 ブロック) ・OFST 列：ブロック内のバイト位置 ・ADR 列：バイト単位のアドレスの番号(MPC の OUT、IN コマンドに対応) ・LSB 列：各バイトの最下位のビット番号(MPC の ON、OFF、SW コマンドに対応)
メインメニュー	・Option MKY43 R/W Check MKY43 のグローバルメモリをチェックします。正常終了しない場合は USB-CUnet のハード故障、USB 通信障害などが考えられます。

■ ステータスバー

下部のステータスバーに MKY43 の主要レジスタの内容、ファームウェアや DLL のバージョン等を表示します。

例えば、CUnet 間に通信障害が生じてメンバーが変化すると LED1(MCARE),LED2(LCARE)が点灯し、その発生回数が CCTR に加算されます(maxFF 回)。実行中に増えていく場合は各ステーションの SA、OWN、ケーブル、終端抵抗設定等を確認してください。(※CCTR は他のステーションがリセットされたり、途中参加した場合でも加算されますが、それは異常ではありません)

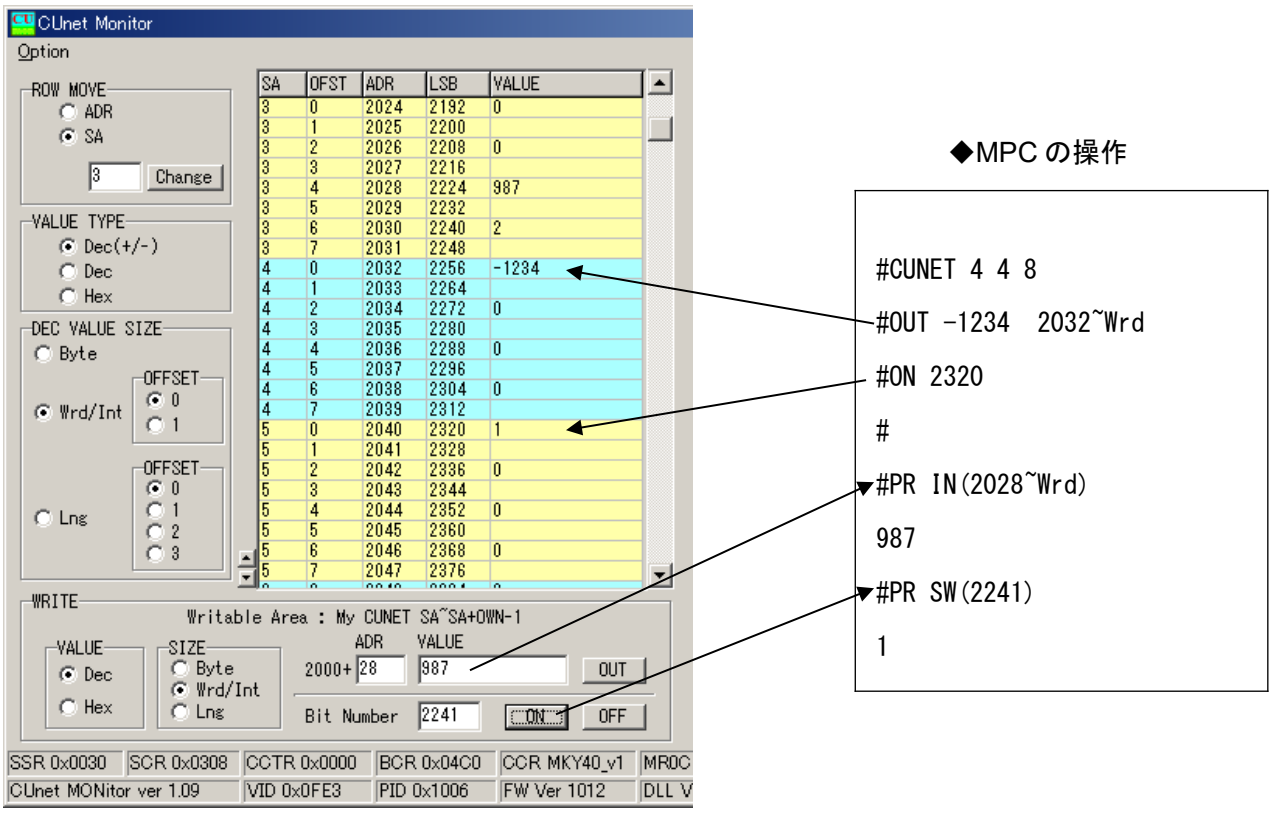


表示	概要
SSR (System Status Register)	ネットワークの稼働における各種ステータスが格納されるレジスタです。
SCR (System Control Register)	CUnet のネットワークをコントロールするレジスタです。
CCTR (Care CounTer Register)	LCARE 信号および MCARE 信号の発生回数が格納されるレジスタです。 上位バイト MCARE(LED1)、下位バイト LCARE(LED2)
BCR (Basic Control Register)	CUnet を構築する MKY43 の基本設定が格納されるレジスタです。
CCR (Chip Code Register)	MKY43 チップコード。“MKY43_v0”
MR0CR (Mail Receive 0 Control Register)	MRB0(Mail Receive Buffer 0)に対応するメール受信をコントロールするレジスタです。
MR1CR (Mail Receive 1 Control Register)	MRB1(Mail Receive Buffer 1)に対応するメール受信をコントロールするレジスタです。
MSCR (Mail Send Control Register)	MSB(Mail Send Buffer)へライトしたデータセットのメール送信をコントロールするレジスタです。
MESR (Mail Error Status Register)	メール送信の開始以後、メール送信エラーが発生した場合のエラー内容を示すレジスタです。
MSRR (Mail Send Result Register)	メール送信に要した所要時間が格納されるレジスタです。
MFR (Member Flag Register)	個々のメンバ状態が格納されています。 下位ビットより SA0～63 に対応しています。

※レジスタの詳細内容は ㈱ステップテクニカ「ユーザーズマニュアル」を参照。

CUnet MONitor Ver	本アプリケーションのバージョン
VID (Vender ID)	USB ベンダー ID(FE3)です。
PID (Product ID)	USB プロダクト ID(1006)です。
FW Ver (Firmware Version)	USB-CUnet ファームウェア(Cyload1.spt)のバージョンです。
DLL Ver (DLL Version)	usbcunet.dll のバージョンです。

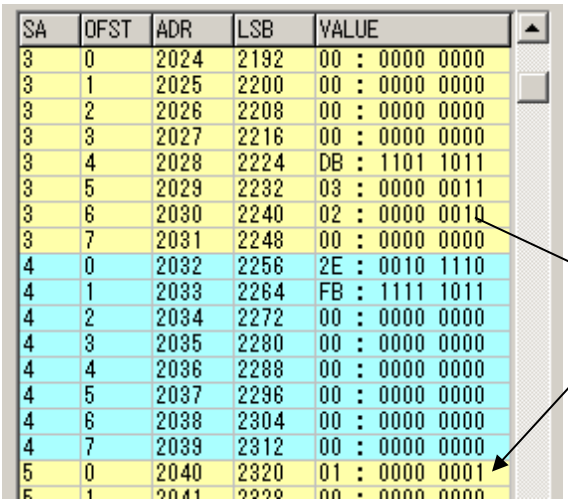
■ 実行例１[グローバルメモリのモニタ]



◆MPC の操作

```
#CUNET 4 4 8
#OUT -1234  2032~Wrd
#ON 2320
#
#PR IN(2028~Wrd)
987
#PR SW(2241)
1
```

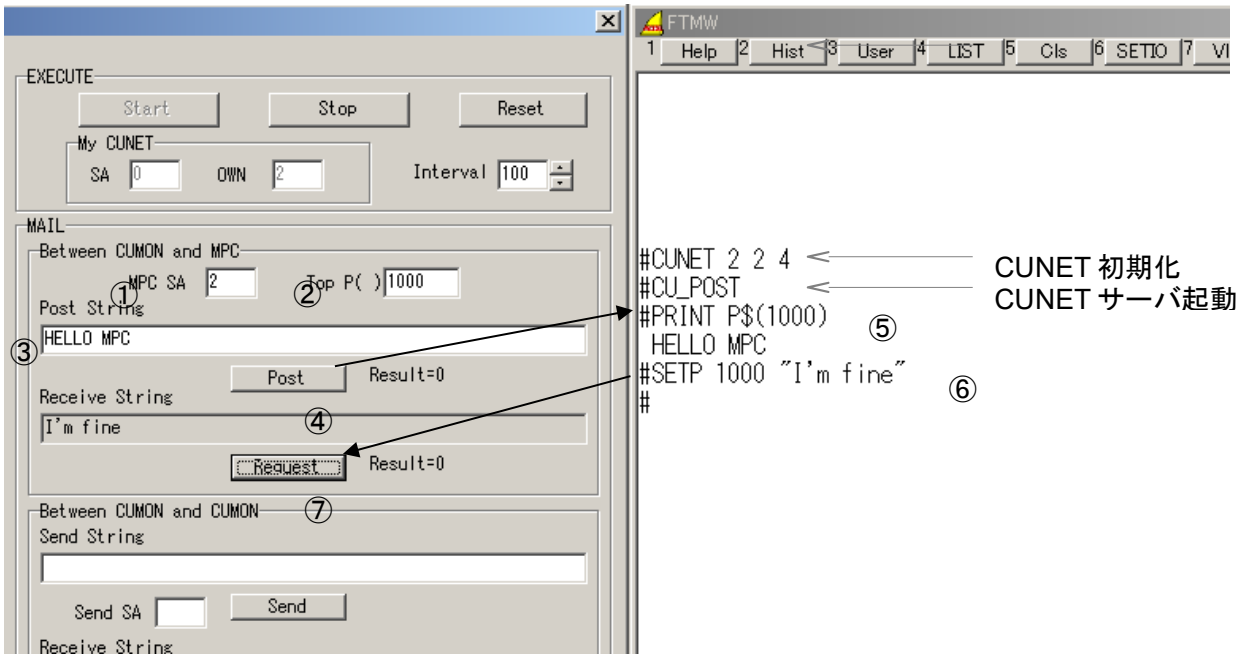
- ON/OFF 操作の時は Hex 表示にすると分かりやすくなります。



SA	OFST	ADR	LSB	VALUE
3	0	2024	2192	00 : 0000 0000
3	1	2025	2200	00 : 0000 0000
3	2	2026	2208	00 : 0000 0000
3	3	2027	2216	00 : 0000 0000
3	4	2028	2224	DB : 1101 1011
3	5	2029	2232	03 : 0000 0011
3	6	2030	2240	02 : 0000 0010
3	7	2031	2248	00 : 0000 0000
4	0	2032	2256	2E : 0010 1110
4	1	2033	2264	FB : 1111 1011
4	2	2034	2272	00 : 0000 0000
4	3	2035	2280	00 : 0000 0000
4	4	2036	2288	00 : 0000 0000
4	5	2037	2296	00 : 0000 0000
4	6	2038	2304	00 : 0000 0000
4	7	2039	2312	00 : 0000 0000
5	0	2040	2320	01 : 0000 0001

```
#CUNET 4 4 8
#OUT -1234  2032~Wrd
#ON 2320
#
#PR IN(2028~Wrd)
987
#PR SW(2241)
1
```

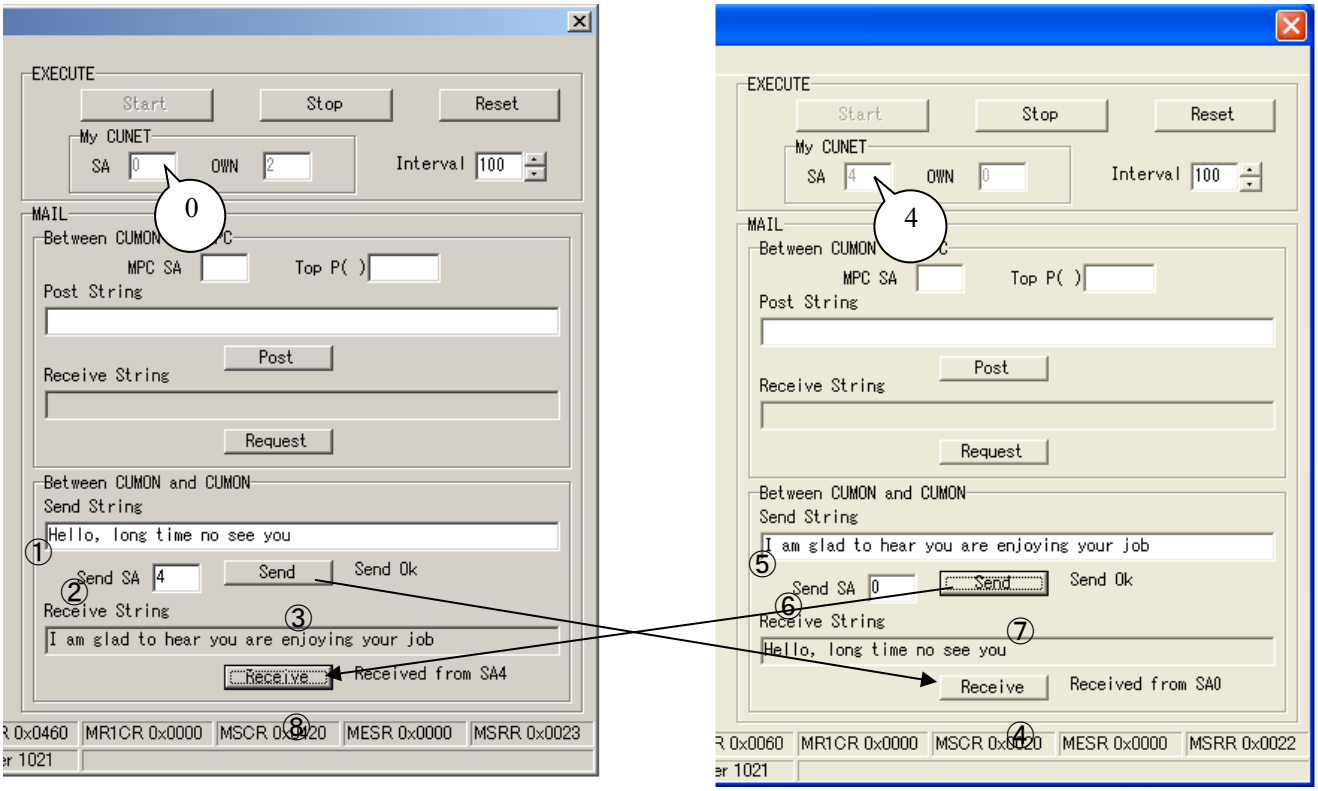
■ **実行例 2 [対 MPC 間メール]**



①メールを送受信する MPC の SA を入力
②MPC 点データのメールで使用するエリアの先頭番号を入力
③送信する文字列を入力
④「Post」を押す → MPC の P(1000)以降に文字列が入る
⑤MPC 側で確認表示
⑥MPC 側で P(1000)以降にメール文字列作成
⑦MPC にメールをリクエスト&受信

※送受信結果(Result)は Mail Error Status Register の内容です。正常なら Result=0 になります。

■ **実行例 3 [対 CUMON 間メール]**



Windows 2000　　Windows XP

①(2000)送信する文字列を入力
②(2000)送信先の SA を入力
③(2000)「Send」を押す → Send Buffer に書込み → 指定 SA へ送信
④(XP)受信側「Receive」を押す → Receive バッファから読み込む
⑤(XP)送信する文字列を入力
⑥(XP)送信先の SA を入力
⑦(XP)「Send」を押す → Send Buffer に書込み → 指定 SA へ送信
⑧(2000)受信側「Receive」を押す → Receive バッファから読み込む

関係資料

■ **サンプルプログラム**

MPC-2000 情報のアプリケーションノートや技術情報に使用例がありますのでご参照ください。

■ **出典、引用**

「CUnet」は、株式会社ステップテクニカの登録商標です。

株式会社ステップテクニカ　http://www.steptechnica.com/
「CUnet ユーザーズマニュアル」 第 2 版
「Let's Try! CUnet 入門者編・フィールドエンジニア編」
「CUnet 導入ガイド　（CUnet プロトコル基本解説）」
「CUnet テクニカルガイド（ネットワーク用）」

これらの資料から、引用または原文のまま掲載しています。

諸注意

■ グローバルメモリについて

アプリケーションを終了してもグローバルメモリは保持されます。グローバルメモリで他局とインターロックをしている場合、各機器の再起動時のタイミングによってはインターロックが不整合になることもあります。必要に応じて FormClosing 等で cunet_init(0, 4, 7) などと実行してグローバルメモリをクリアしてください。

■ usbcunet.dll の MPC 間メール転送について

MPC 間メール転送関数の開始時に送信先の MPC-CUnet が受信不可状態(NORDY)であると、転送作業を実行することができません。その場合、これらの関数は直ちに戻ってきます。post 関数ならデータは MPC に伝わりません。req 関数なら受信データに 0 が格納されて戻ります。

MPC のタスク数が多い、時間浪費タスク(MON コマンドで!マークが付くタスク)ある場合 NORDY 状態が発生します。このような状況下では正常なデータ授受を行えません。レスポンスを確認し、NORDY ならば MPC のプログラムを見直して下さい。

戻値は MKY43 のレジスタの内容を表します。
0:正常です。
-1:MSC レジスタ SEND が下がらない。 = 送信が完了しない。
-2:MRC0 レジスタ RDY が立たない。 = メール着信の許可ができない (post 関数に-2 はありません)
1 以上:MESR(Mail Error Status Register)の内容。

MKY43 Mail Error Status Register		
bit	status	意味
5	STOP	通信フェーズが RUN フェーズ以外のため通信できなかった。
4	LMFLT	メールセンドリミットレジスタに設定された値が不正。
3	SZFLT	メールの送信サイズは不正
2	TOUT	メールセンドリミットレジスタに設定された制限時間内に送信が完了しなかった
1	NOEX	送信先のステーションアドレスが存在しない
0	NORDY	送信先のステーションが受信待機中でない

- リトライの例

```
For i = 0 To 17      ' pnt データ連続書き込み
    Do
        res = cunet_post_pnt(4, 1000 + i * 15, ar(i * 15 * 4))
        TextBox_res.SelectedText = CInt(i) & " Mail Error Status register " & Hex$(res) &
vbCrLf
        If res = 0 Then Exit Do
    Loop
Next i
-----
For i = 0 To 15      ' MBK データ連続読み込み
    Do
        res = cunet_req_mbk_lng(4, i * 120 + 1000, ar(i * 60))
        If res = 0 Then Exit Do
    Loop
Next i
```

■ MPC の CU_POST の実行タスク

MPC のプログラムで CU_POST のパラメータを省略すると CUnet メールサーバは空タスクで実行されますが、次のような順番で宣言すると、LOAD 後の初回実行時にタスク 31 が重複します。

```
CU_POST             /* パラメータを省略するとタスク 31 から空タスクを探して割り当てられる
MEWNET   38400 1    /* ch1 はタスク 31 で実行。MEWNET は ch によりタスクが固定される。
```

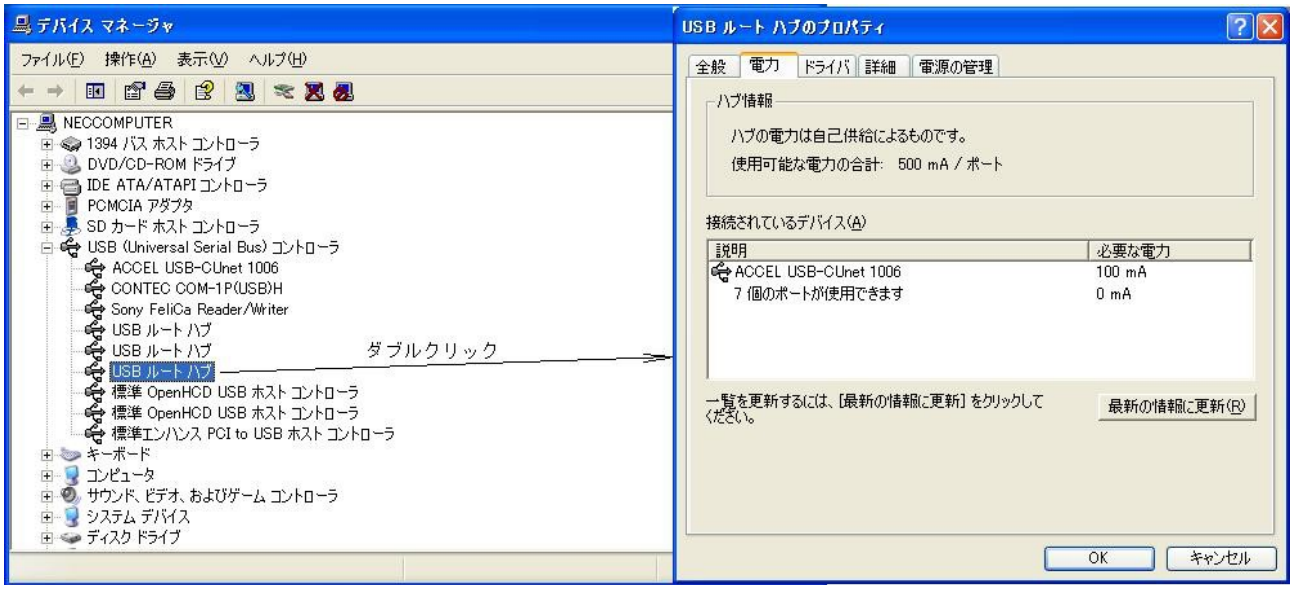
FORK されたタスクとかぶることもあります。かぶるとメール通信に障害が発生します(NORDY)。
明確なタスク管理のために CU_POST は明示的にタスクを指定することを推奨します。

```
CU_POST   25        /* タスク 25
```

■ パソコン USB ハブ消費電流確認 (Windows XP)

パソコンの同一 USB ハブに多数の USB 機器を接続し消費電流が多くなると USB-CUnet のアクセスが不安定になるようです。
グローバルメモリのアクセスに時間がかかるといった症状がでるようです。
1 ポート当たりの消費電流が定格未満でもハブの総消費電流に注意してください。
マザーボード直結の USB ポートでも不具合が出る PC もあるようです。そういう場合は増設ボードをご検討ください。

↓Windows XP



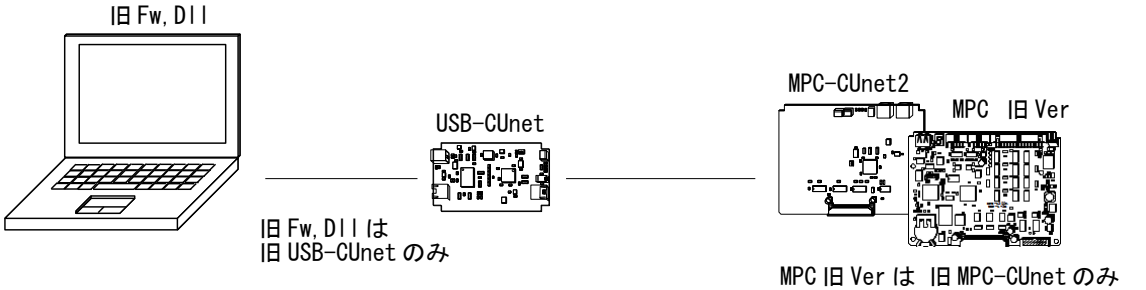
互換性

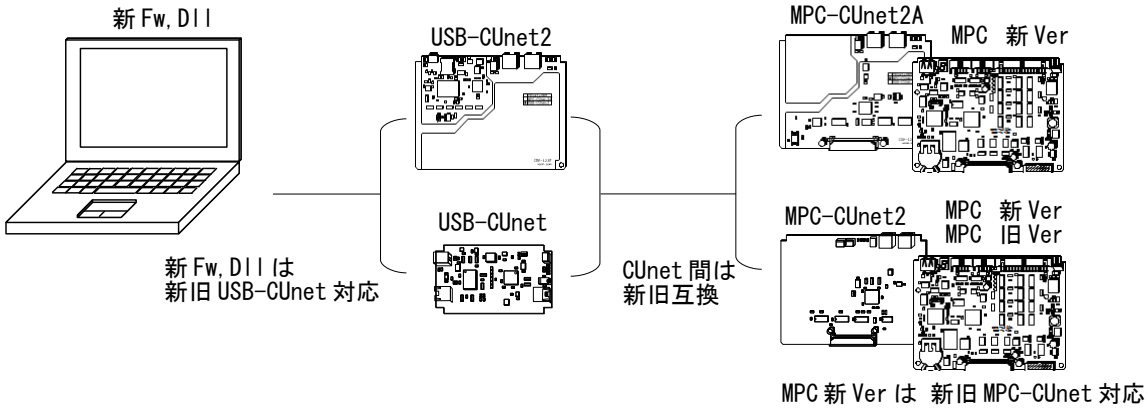
PC の「旧 Fw,Dll」とは スクリプトファイル Cyload1.spt とライブラリ usbcunet.dll のタイムスタンプが 2014 年以前のもの、「新 Fw,Dll」はそれらが 2015 年以降のものです。

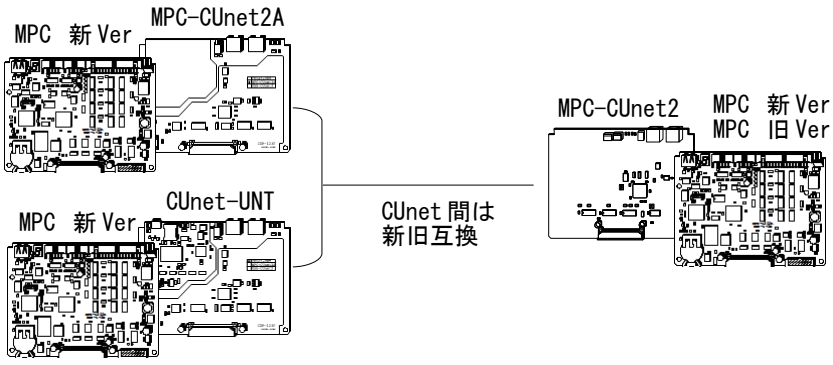
インストーラ Accel_Setup_jpn.msi は Ver8.14 以前が「旧 Fw,Dll」、Ver8.15 以降が「新 Fw,Dll」です。既に「旧 Fw,Dll」がインストールされている PC の更新方法はホームページ MPC2000 情報 ＞ 技術情報 の

「USB-CUnet ドライバパッケージの更新方法」をご参照ください。

MPC メイン CPU のバージョンは 1.14_52 以前を「旧 Ver」、1.14_53 以降を「新 Ver」としています。

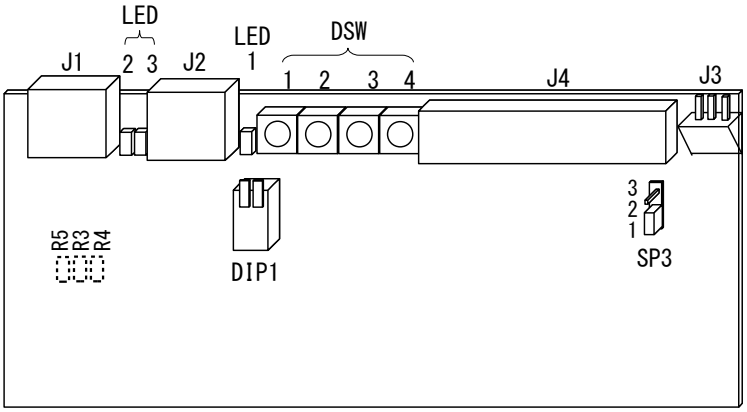






MPC-CUIO

MPC-CUIO は入力 16 点、出力 16 点を備えたリモート I/O です。MPC の遠隔・増設・省配線 I/O として、また、パソコンの I/O としても利用できます。



基板サイズ
（突起含まず）
縦 65mm
横 135mm

専用ケース有り

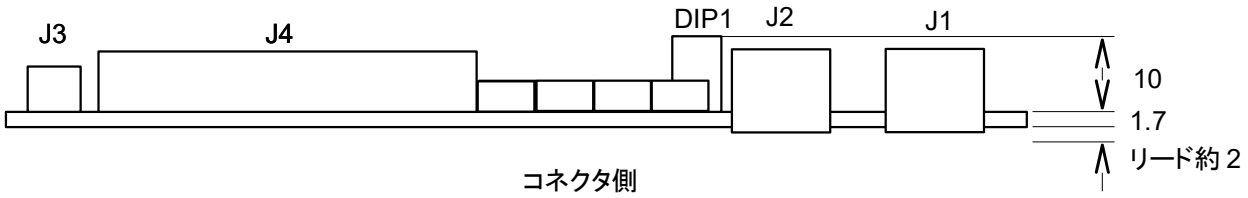
J1,J2	RJ-45 8 極	他局間 CUnet 接続
J3	電源	DC24V 給電
LED1	POWER	内部電源給電　点灯で正常
LED2	DONA 信号	出力正常更新　点灯で正常
LED3	#MON 信号	リンク成立　　点灯で正常
DSW1	DOSA(H)	出力 DOSA アドレス上位 (Hex)
DSW2	DOSA(L)	出力 DOSA アドレス下位 (Hex)
DSW3	SA(H)	入力（自己）SA アドレス上位 (Hex)
DSW4	SA(L)	入力（自己）SA アドレス下位 (Hex)
J4	I/O	入力 16 点，出力 16 点
DIP1	終端抵抗設定	抵抗無し /100Ω/200Ω
SP3	ショートピン	J4 33 番ピン GND/DC24V 選択
R3	通信レート (BPS1)	0Ω 実装で 3M
R5	通信レート (BPS0)	0Ω 実装で 6M

■ **J4,J3 ピン配**

	pin	port no.	bank no.	
J4	1	SW(SA*64+2000)		
	2	SW(SA*64+2001)		
	3	SW(SA*64+2002)		
	4	SW(SA*64+2003)	IN(SA*8+2000)	
	5	SW(SA*64+2004)		
	6	SW(SA*64+2005)		
	7	SW(SA*64+2006)		SA=(&H)DSW3<<4+(&H)DSW4
	8	SW(SA*64+2007)		
	9	SW(SA*64+2008)		
	10	SW(SA*64+2009)		
	11	SW(SA*64+2010)		フォトカプラ入力
	12	SW(SA*64+2011)	IN(SA*8+2001)	2 線式無接点センサー対応（漏れ電流 1mA 以下）
	13	SW(SA*64+2012)		
	14	SW(SA*64+2013)		
	15	SW(SA*64+2014)		
	16	SW(SA*64+2015)		
	17	ON DOSA*64+2048		
	18	ON DOSA*64+2049		
	19	ON DOSA*64+2050		
	20	ON DOSA*64+2051	OUT DOSA*8+2006	
	21	ON DOSA*64+2052		
	22	ON DOSA*64+2053		
	23	ON DOSA*64+2054		DOSA=(&H)DSW1<<4+(&H)DSW2
	24	ON DOSA*64+2055		
	25	ON DOSA*64+2056		
	26	ON DOSA*64+2057		17 ～ 30pin トラジスタオープンコレクタ　Max100mA
	27	ON DOSA*64+2058		
	28	ON DOSA*64+2059	OUT DOSA*8+2007	
	29	ON DOSA*64+2060		
	30	ON DOSA*64+2061		
	31	ON DOSA*64+2062		31,32pin FET オープンドレイン　Max600mA
	32	ON DOSA*64+2063		
	33	SP3 1-2 ショート :GND（出荷時）,2-3 ショート :DC24V（接続機器の電源として使用可　Max500mA）		
	34	GND		

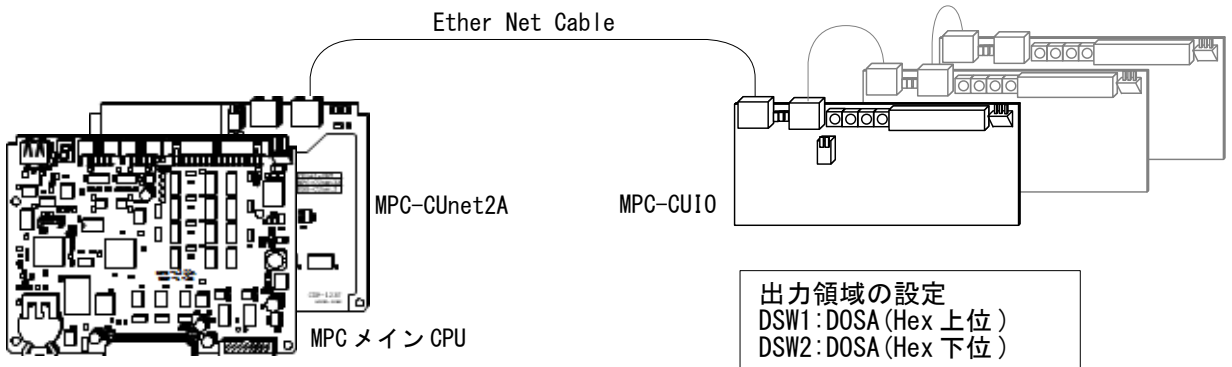
J3	1	DC24
	2	GND
	3	FG

■ ボードサイズ



アルミ製専用ケースがあります。スルーホール(4 箇所)でスタッド取り付けもできます。

■ MPC-CUnet2A との接続



CUNET sa own end
sa: 占有開始 SA } 占有領域内の SA の
own: 占有数 上位 2byte を出力できる
end: 最大共有 SA

例)CUNET 8 1 9 とすると
MPC 側は SA8(8byte) を占有し、その上位 2byte が右の設定の MPC-CUIO から出力される。
(※SA8 の下位 6byte は出力できない)

入力は共有範囲内ならどこでも読める。

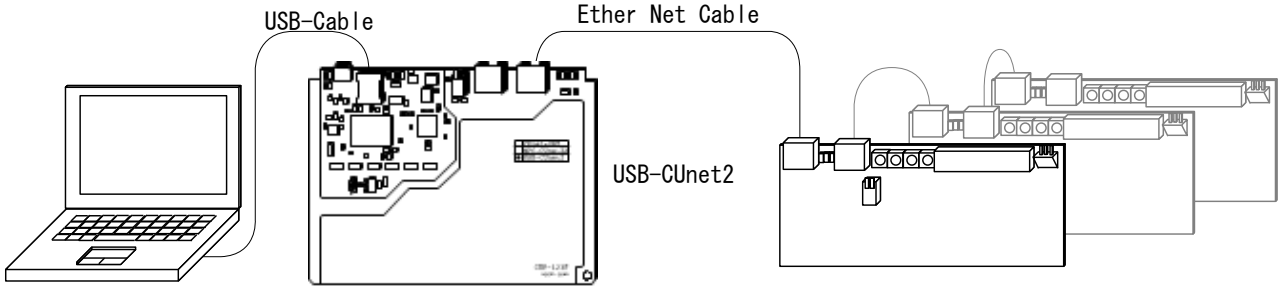
出力領域の設定
DSW1:DOSA(Hex 上位)
DSW2:DOSA(Hex 下位)

DOSA の上位 2byte が出力される。
例)DSW1:0 、 DSW2:8 とすると
SA8 の上位 2byte の内容を出力する。

入力領域の設定
DSW3:SA(Hex 上位)
DSW4:SA(Hex 下位)

SA の下位 2byte が入力となる。
例)DSW3:0 、 DSW4:9 とすると
入力範囲は SA9 の下位 2byte
(※SA9 の上位 6byte は入力できない)

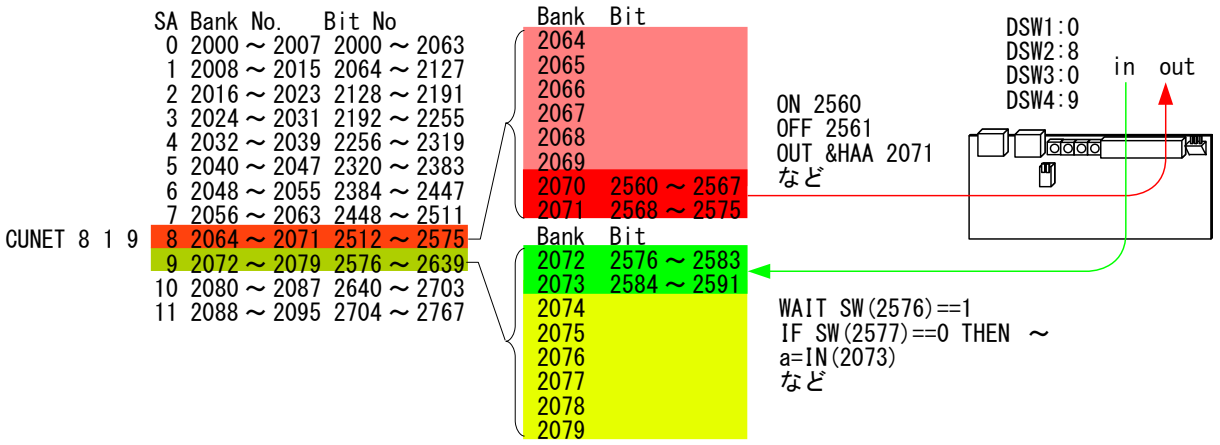
■ USB-CUnet2 との接続



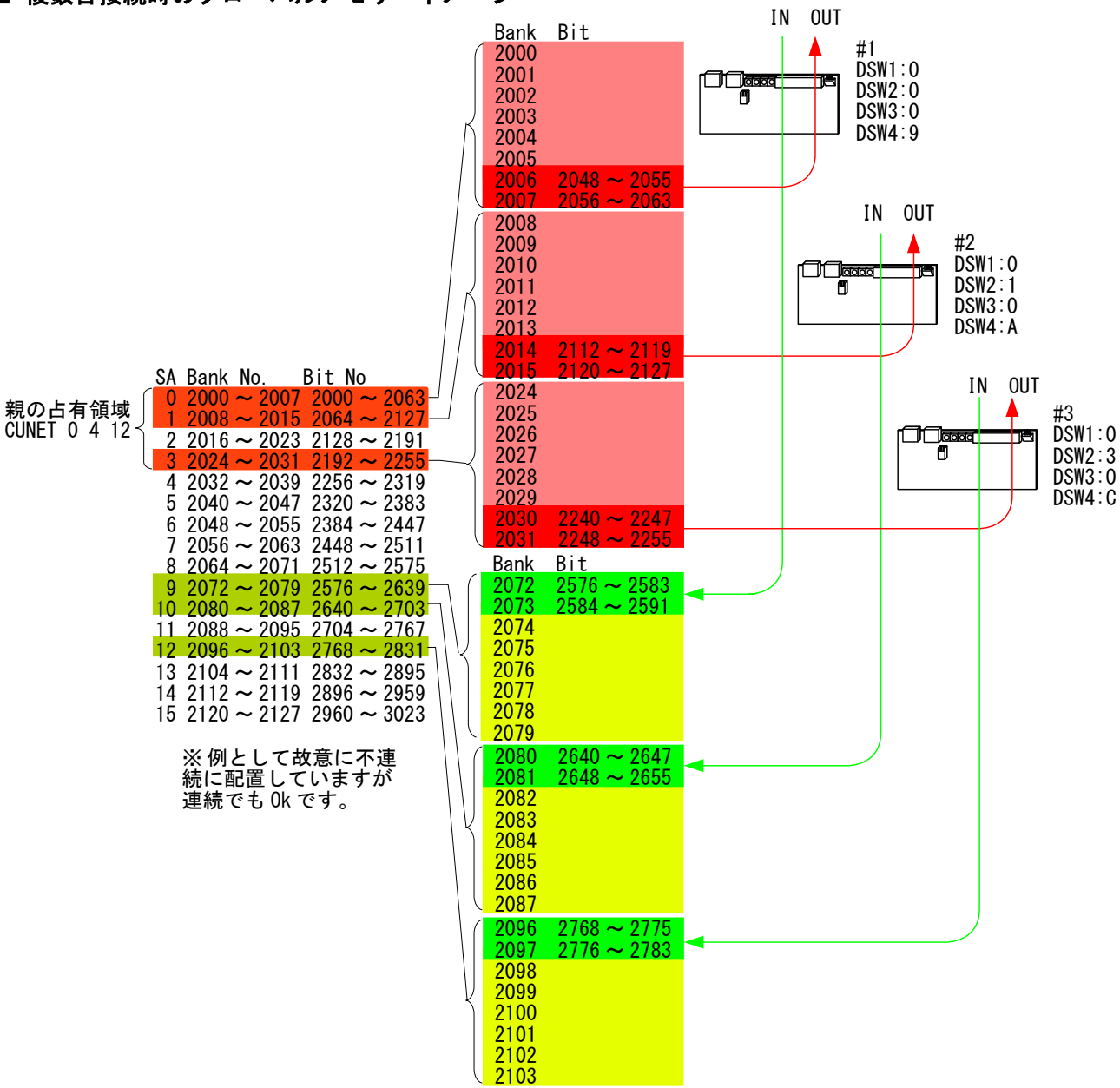
cunet_init(sa, own, end)
sa: 占有開始 SA } 占有領域内の SA の
own: 占有数 上位 2byte を出力できる
end: 最大共有 SA

■ グローバルメモリーイメージ

上記の設定でのグローバルメモリーと入出力の関係



■ 複数台接続時のグローバルメモリーイメージ



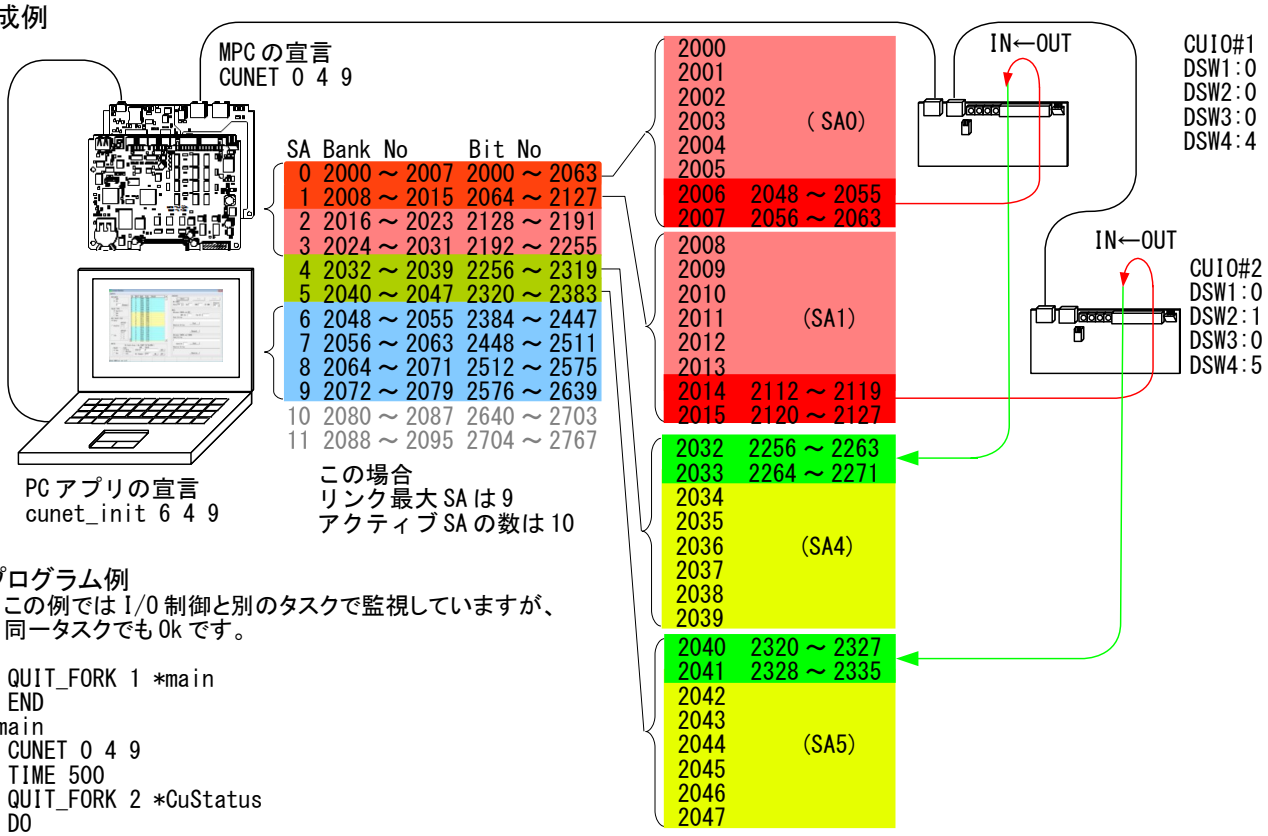
■ MPC サンプルプログラム

```
  CUNET 0 4 12
  QUIT_FORK 1 *main
  END
*main
  QUIT_FORK 2 *task2
  DO                          /* MPC-CUIO #1
    IF SW(2576)==1 THEN
      ON 2048
    ELSE
      OFF 2048
    END_IF
  LOOP
*task2
  DO                          /* MPC-CUIO #3
    FOR i=0 TO 255
      OUT i 2031
      TIME 100
      PR IN(2097)
    NEXT
  LOOP
```

■ リンク状態のチェック

MKY(n)関数でアクティブな SA 数の変化や SA の有無などを監視してリンク状態を確認します。
詳しくはコマンドリファレンス MKY を参照して下さい。

構成例



プログラム例

この例では I/O 制御と別のタスクで監視していますが、
同一タスクでも Ok です。

```
  QUIT_FORK 1 *main
  END
*main
  CUNET 0 4 9
  TIME 500
  QUIT_FORK 2 *CuStatus
  DO
    FOR i=0 TO &HFFFF
      OUT i 2006~Wrd              /* CUIO#1 出力
      OUT i 2014~Wrd              /* CUIO#2 出力
      TIME 100
      IF i<>IN(2032~Wrd) THEN     /* CUIO#1 入力
        PRINT "CUIO#1 出力<>入力" : END
      END_IF
      IF i<>IN(2040~Wrd) THEN     /* CUIO#2 入力
        PRINT "CUIO#2 出力<>入力" : END
      END_IF
    NEXT
  LOOP
*CuStatus
  /* 初期値
  sa_count=10                     /* アクティブな SA の数。
  CUM_MEM=0                       /* SA の消失または参加
  DO
    tmp=MKY(11)
    IF (tmp<>sa_count)|(CUM_MEM<>0) THEN /* 状態が変ったら個別に SA を調べる
      PRINT "SA 数が変化" sa_count "->" tmp
      sa_count=tmp : CUM_MEM=0
      PR "SA31 ～ 00" HEX$(MKY(6)) /* アクティブ SA のビットが 1 になる
      PR "SA63 ～ 32" HEX$(MKY(7)) /* アクティブ SA のビットが 1 になる
    END_IF
    SWAP
  LOOP
```

実行例

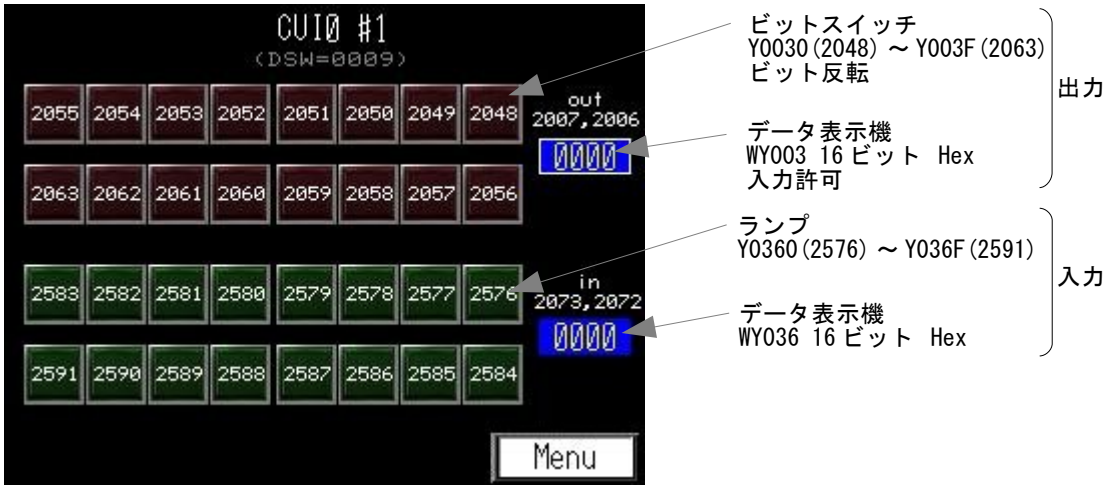
```
RUN
# CUIO#2 出力<>入力  ← CUOI#2 の電源断 / ケーブル断
 SA 数が変化 10 -> 9  ← アクティブな SA 数が変化
 SA31 ～ 00 000003DF   ← (上位 SA31) 0000 0000 0000 0000 0000 0011 1101 1111 (下位 SA0)
 SA63 ～ 32 00000000   ← (上位 SA63) 0000 0000 0000 0000 0000 0000 0000 0000 (下位 SA32)
 SA 数が変化 9 -> 10   ← CUOI#2 の電源入 / ケーブル接続
 SA31 ～ 00 000003FF   ← (上位 SA31) 0000 0000 0000 0000 0000 0011 1111 1111 (下位 SA0)
 SA63 ～ 32 00000000
 SA 数が変化 10 -> 6   ← PC アプリがリセットされて SA6 ～ SA9 が無くなる
 SA31 ～ 00 0000003F   ← (上位 SA31) 0000 0000 0000 0000 0000 0000 0011 1111 (下位 SA0)
 SA63 ～ 32 00000000
```

CUnet のタッチパネル対応

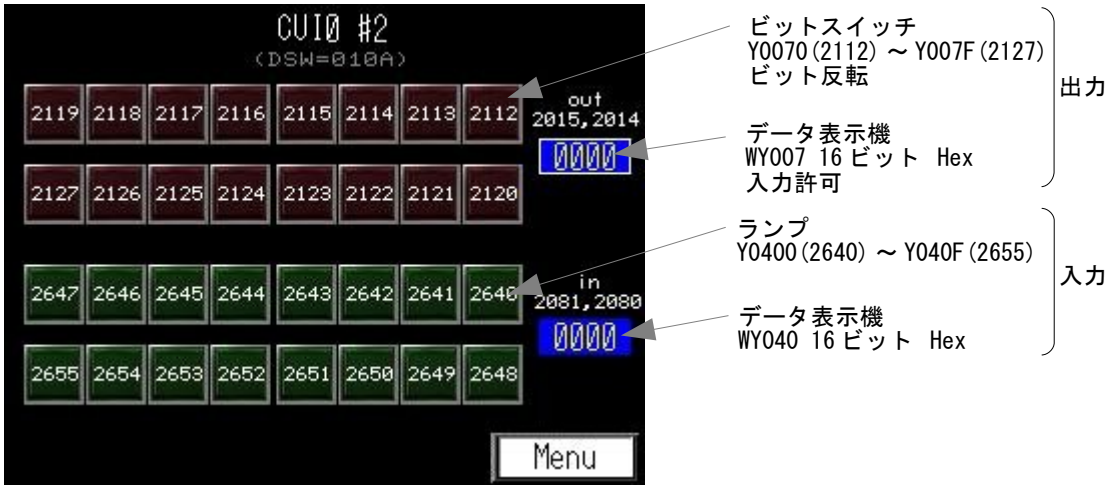
CUnet の I/O はタッチパネルの Y エリアに割り当てられています。
タッチパネルはワード(16 ビット)単位です。
(MPC バージョン BL/I 1.14_57 2017/01/20 以降)

SA	MPC IN/OUT	MPC ON/OFF/SW	タッチパネル ビット対応	タッチパネル ワード対応
0	2000, 2001	2000~2015	Y0000~Y000F	WY000
	2002, 2003	2016~2031	Y0010~Y001F	WY001
	2004, 2005	2032~2047	Y0020~Y002F	WY002
	2006, 2007	2048~2063	Y0030~Y003F	WY003
1	2008, 2009	2064~2079	Y0040~Y004F	WY004
	2010, 2011	2080~2095	Y0050~Y005F	WY005
	2012, 2013	2096~2111	Y0060~Y006F	WY006
	2014, 2015	2112~2127	Y0070~Y007F	WY007
中略				
9	2072, 2073	2576~2591	Y0360~Y036F	WY036
	2074, 2075	2592~2607	Y0370~Y037F	WY037
	2076, 2077	2608~2623	Y0380~Y038F	WY038
	2078, 2079	2624~2639	Y0390~Y039F	WY039
10	2080, 2081	2540~2555	Y0400~Y040F	WY040
	2082, 2083	2556~2671	Y0410~Y041F	WY041
	2084, 2085	2672~2687	Y0420~Y042F	WY042
	2086, 2087	2688~2703	Y0430~Y043F	WY043
以下略				

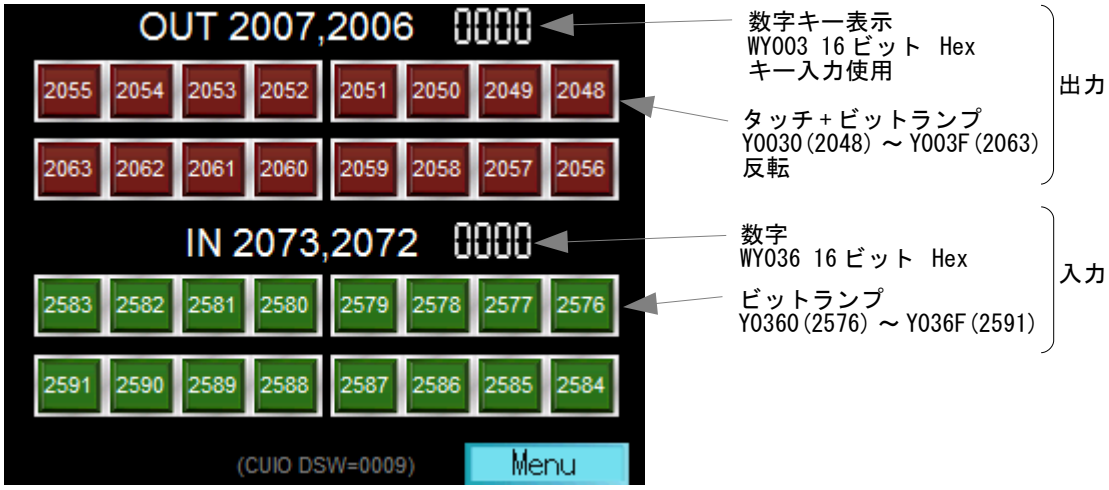
前ページの CUIO #1 (DSW=0009)のモニター画面作成例 (デジタル GP)



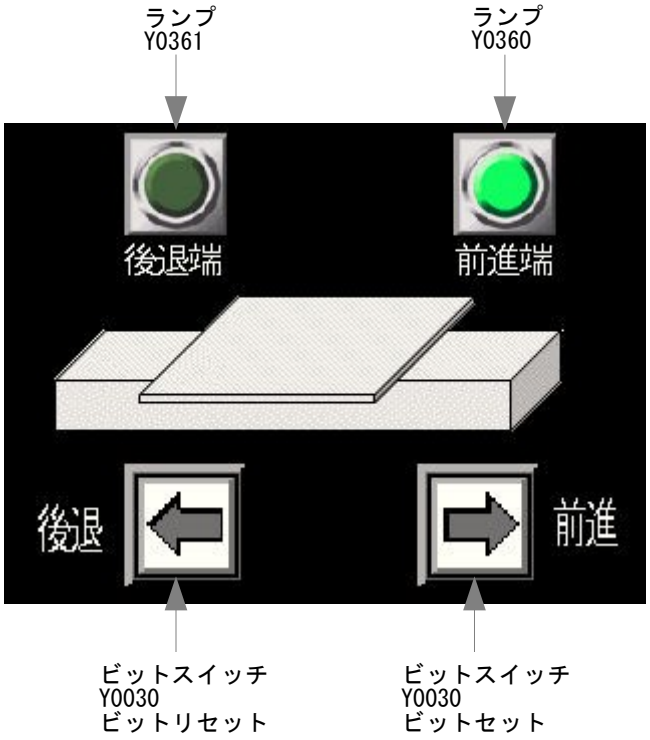
CUIO #2 (DSW=010A)のモニター画面作成例



ミスミ GX7 の CUIO #1 画面例

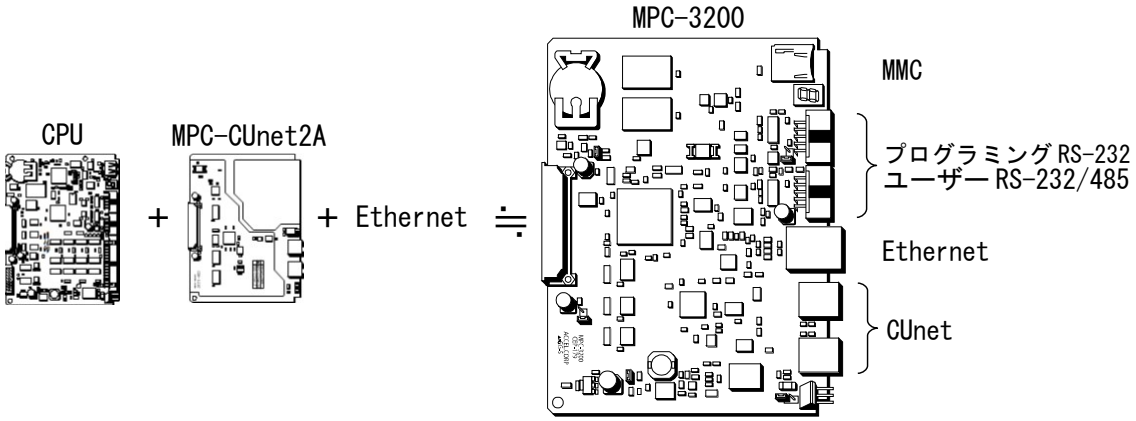


これらの部品はユーザーインターフェースとして個別に配置することができます。

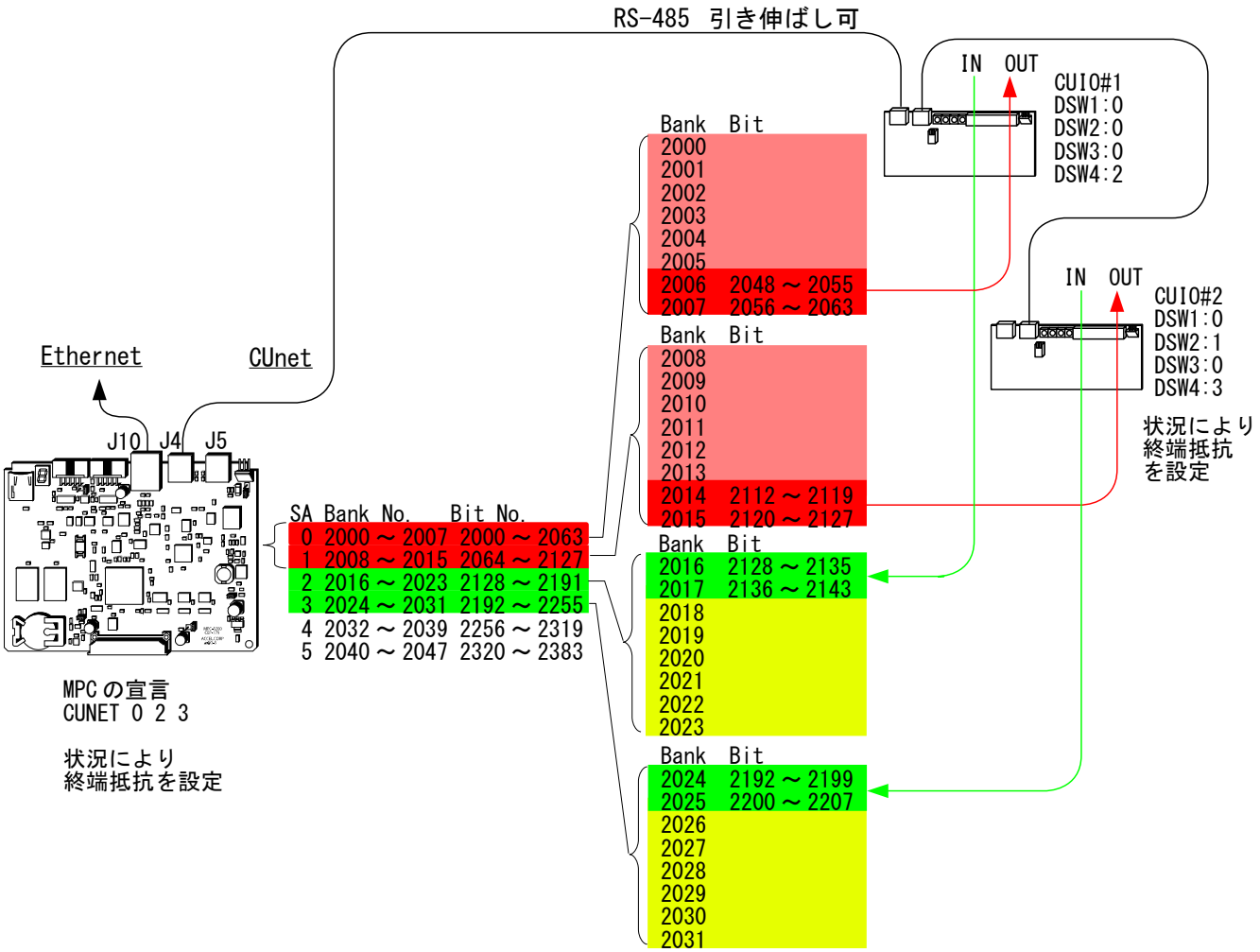


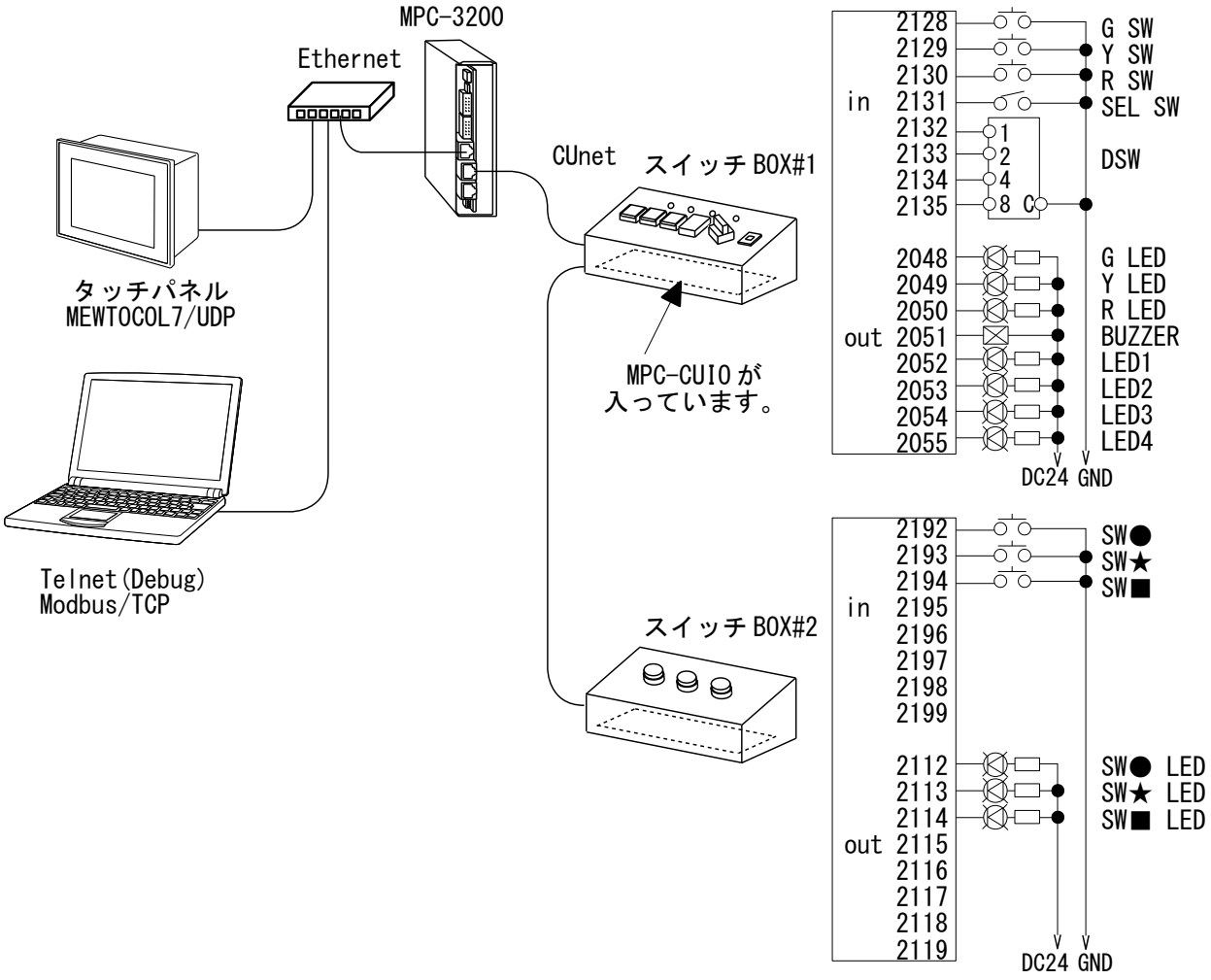
MPC-3200

■MPC-3200 は CUnet と Ethernet を搭載した CPU ボードです。
(実 I/O、USB メモリコネクタはありません。)



■使用例
リモート I/O ユニット「MPC-CUIO」を制御します。
パソコン、タッチパネルとのコミニュケーションは Ethernet TCP/UDP で行います。





PC アプリ、タッチパネルとスイッチ BOX が連動します （MPC-3200 はケースの蓋を外した状態です）

グローバルメモリアドレス表

SA	IN/OUT バンク番号	ON/OFF/SW ビット番号	タッチパネル WY	
0	2000～2007	2000～2063	000～003	
1	2008～2015	2064～2127	004～007	
2	2016～2023	2128～2191	008～011	
3	2024～2031	2192～2255	012～015	
4	2032～2039	2256～2319	016～019	
5	2040～2047	2320～2383	020～023	
6	2048～2055	2384～2447	024～027	
7	2056～2063	2448～2511	028～031	
8	2064～2071	2512～2575	032～035	
9	2072～2079	2576～2639	036～039	
10	2080～2087	2640～2703	040～043	
11	2088～2095	2704～2767	044～047	
12	2096～2103	2768～2831	048～051	
13	2104～2111	2832～2895	052～055	
14	2112～2119	2896～2959	056～059	
15	2120～2127	2960～3023	060～063	
16	2128～2135	3024～3087	064～067	
17	2136～2143	3088～3151	068～071	
18	2144～2151	3152～3215	072～075	
19	2152～2159	3216～3279	076～079	
20	2160～2167	3280～3343	080～083	
21	2168～2175	3344～3407	084～087	
22	2176～2183	3408～3471	088～091	
23	2184～2191	3472～3535	092～095	
24	2192～2199	3536～3599	096～099	
25	2200～2207	3600～3663	100～103	
26	2208～2215	3664～3727	104～107	
27	2216～2223	3728～3791	108～111	
28	2224～2231	3792～3855	112～115	
29	2232～2239	3856～3919	116～119	
30	2240～2247	3920～3983	120～123	
31	2248～2255	3984～4047	124～127	
32	2256～2263	4048～4111	128～131	
33	2264～2271	4112～4175	132～135	
34	2272～2279	4176～4239	136～139	
35	2280～2287	4240～4303	140～143	
36	2288～2295	4304～4367	144～147	
37	2296～2303	4368～4431	148～151	
38	2304～2311	4432～4495	152～155	
39	2312～2319	4496～4559	156～159	
40	2320～2327	4560～4623	160～163	
41	2328～2335	4624～4687	164～167	
42	2336～2343	4688～4751	168～171	
43	2344～2351	4752～4815	172～175	
44	2352～2359	4816～4879	176～179	
45	2360～2367	4880～4943	180～183	
46	2368～2375	4944～5007	184～187	
47	2376～2383	5008～5071	188～191	
48	2384～2391	5072～5135	192～195	
49	2392～2399	5136～5199	196～199	
50	2400～2407	5200～5263	200～203	
51	2408～2415	5264～5327	204～207	
52	2416～2423	5328～5391	208～211	
53	2424～2431	5392～5455	212～215	
54	2432～2439	5456～5519	216～219	
55	2440～2447	5520～5583	220～223	
56	2448～2455	5584～5647	224～227	
57	2456～2463	5648～5711	228～231	
58	2464～2471	5712～5775	232～235	
59	2472～2479	5776～5839	236～239	
60	2480～2487	5840～5903	240～243	
61	2488～2495	5904～5967	244～247	
62	2496～2503	5968～6031	248～251	
63	2504～2511	6032～6095	252～255	