CUnetの使い方

June 25, 2009

■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。

MPC-CUnet、USB-CUnet は CUnet デバイス「MKY40」を搭載した通信システムで、複数の MPC 間、MPC～PC 間のメモリ共有を実現します。

■MPC-CUnet、USB-CUnet の概要

CUnet のグローバルメモリは 512byte です。

8byte を一ブロックとして、64 ブロックを SA0～SA63 として管理します。

各 CUnet ステーション(MKY40)は、すべてのエリアの値を読み取り参照することができます。対して、書き換えることのできるエリアは、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(OWN)で指定します。

MPC-2000 のコマンドでは以下のようにします。

CUNET 5 4 31

5（開始ブロック）　SA5～

4（占有ブロック数） 4ブロック占有する。(SA8まで)

31（共有ブロック）　共有エリアを31までに限定する

この例では、SA5～SA8 を書き換え可能となります。

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、MKY40 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

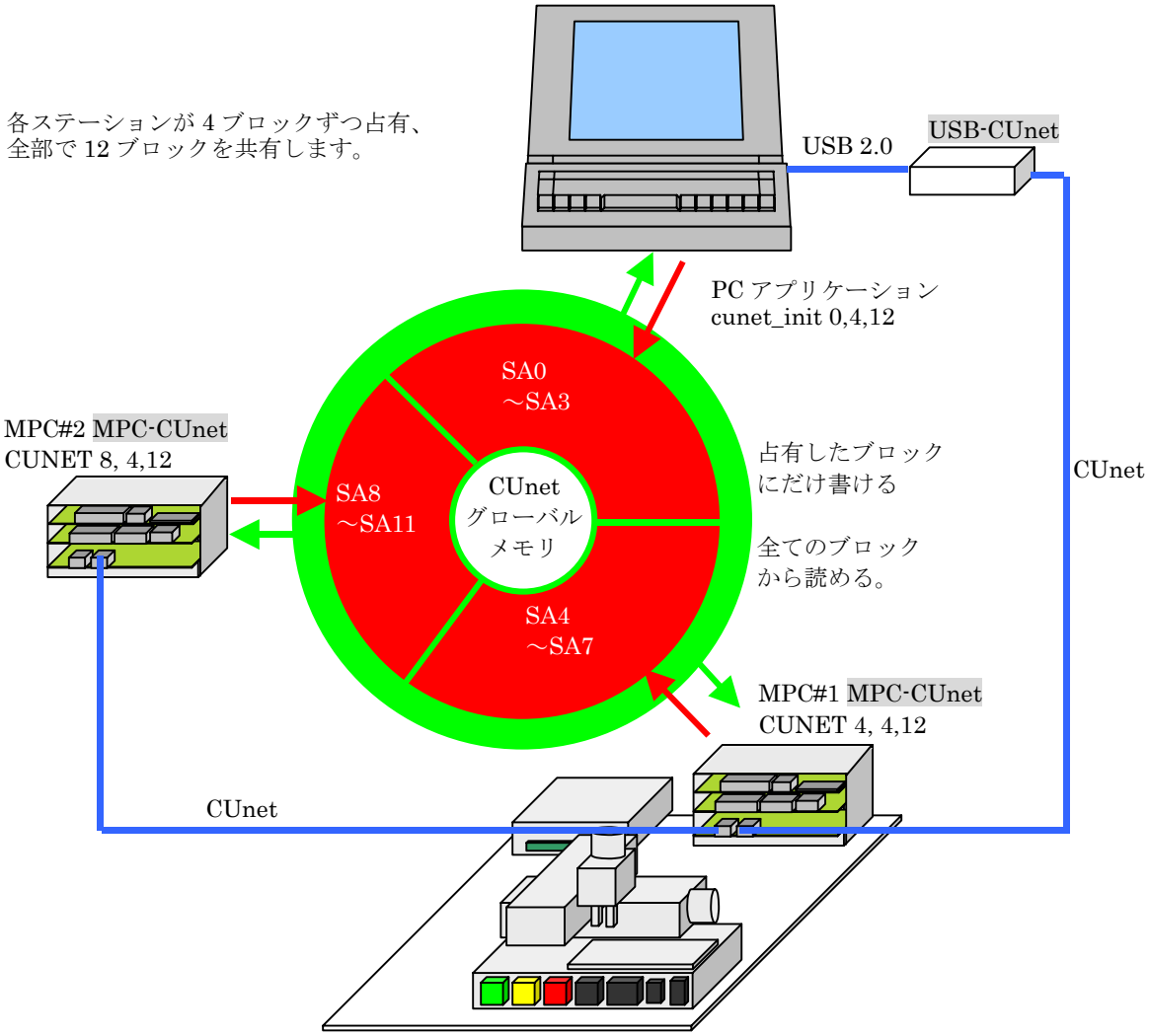
256byte 32 ブロックでは、1.2msec

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。

SA	CUnet Address	IN/OUT バンク番号	ON/OFF/SW ビット番号
0	0～7	2000～2007	2000～2063
1	8～15	2008～2015	2064～2127
2	16～23	2016～2023	2128～2191
3	24～31	2024～2031	2192～2255
4	32～39	2032～2039	2256～2319
5	40～47	2040～2047	2320～2383
6	48～55	2048～2055	2384～2447
7	56～63	2056～2063	2448～2511
8	64～71	2064～2071	2512～2575
9	72～79	2072～2079	2576～2639
10	80～87	2080～2087	2640～2703
11	88～95	2088～2095	2704～2767
12	96～103	2096～2103	2768～2831
13	104～111	2104～2111	2832～2895
14	112～119	2112～2119	2896～2959

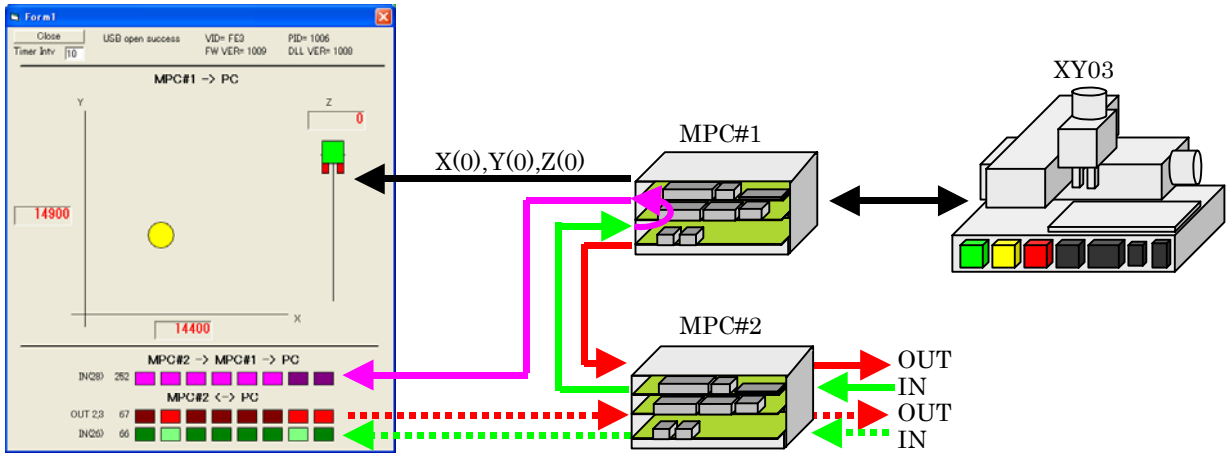
15	120～127	2120～2127	2960～3023
16	128～135	2128～2135	3024～3087
17	136～143	2136～2143	3088～3151
18	144～151	2144～2151	3152～3215
19	152～159	2152～2159	3216～3279
20	160～167	2160～2167	3280～3343
21	168～175	2168～2175	3344～3407
22	176～183	2176～2183	3408～3471
23	184～191	2184～2191	3472～3535
24	192～199	2192～2199	3536～3599
25	200～207	2200～2207	3600～3663
26	208～215	2208～2215	3664～3727
27	216～223	2216～2223	3728～3791
28	224～231	2224～2231	3792～3855
29	232～239	2232～2239	3856～3919
30	240～247	2240～2247	3920～3983
31	248～255	2248～2255	3984～4047
32	256～263	2256～2263	4048～4111
33	264～271	2264～2271	4112～4175
34	272～279	2272～2279	4176～4239
35	280～287	2280～2287	4240～4303
36	288～295	2288～2295	4304～4367
37	296～303	2296～2303	4368～4431
38	304～311	2304～2311	4432～4495
39	312～319	2312～2319	4496～4559
40	320～327	2320～2327	4560～4623
41	328～335	2328～2335	4624～4687
42	336～343	2336～2343	4688～4751
43	344～351	2344～2351	4752～4815
44	352～359	2352～2359	4816～4879
45	360～367	2360～2367	4880～4943
46	368～375	2368～2375	4944～5007
47	376～383	2376～2383	5008～5071
48	384～391	2384～2391	5072～5135
49	392～399	2392～2399	5136～5199
50	400～407	2400～2407	5200～5263
51	408～415	2408～2415	5264～5327
52	416～423	2416～2423	5328～5391
53	424～431	2424～2431	5392～5455
54	432～439	2432～2439	5456～5519
55	440～447	2440～2447	5520～5583
56	448～455	2448～2455	5584～5647
57	456～463	2456～2463	5648～5711
58	464～471	2464～2471	5712～5775
59	472～479	2472～2479	5776～5839
60	480～487	2480～2487	5840～5903
61	488～495	2488～2495	5904～5967
62	496～503	2496～2503	5968～6031
63	504～511	2504～2511	6032～6095

■デモ機の構成と CUnet グローバルメモリのイメージ



■制御イメージ

MPC#1 = XY03 の現在座標値を PC に送ります。また、MPC#2 の I/O を制御します。
MPC#2 = PC アプリケーションと MPC#1 のリモート I/O として動作します。



■usbcunet ライブラリ コマンド・関数(VB6 標準モジュール宣言例)

```
Declare Function cunet_usb_open Lib "usbcunet.dll" () As Long
'USB オープン。
' 引数:無し、戻値:1=成功、1<>失敗

Declare Function cunet_fw_vid Lib "usbcunet.dll" () As Long
'VID 取得。
' 引数:無し、戻値:VID 番号

Declare Function cunet_fw_pid Lib "usbcunet.dll" () As Long
'PID 取得。
' 引数:無し、戻値:PID 番号

Declare Function cunet_fw_ver Lib "usbcunet.dll" () As Long
'ファームウェアのバージョン番号取得。
' 引数:無し、戻値:Ver 番号

Declare Function cunet_dll_ver Lib "usbcunet.dll" () As Long
'DLL のバージョン番号取得。
' 引数:無し、戻値:Ver 番号

Declare Sub cunet_init Lib "usbcunet.dll" (ByVal sa As Long, ByVal ow As Long, ByVal en As Long)
'初期化
' sa:占有するブロックの先頭 SA(Station Address)の番号。例) SA0～SA3 を占有するなら sa=0,SA12～SA15 なら sa=12
' 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:共有するブロック数。例) SA0～SA15 を共有するなら en=16
' 戻値:無し

Declare Function cunet_in Lib "usbcunet.dll" (ByVal adr As Long, ByVal siz As Long) As Long
'パラレル読み込み。全てのブロックを参照可能。
' adr:バンク番号(2000~)
' siz:読み込みサイズ

Declare Sub cunet_out Lib "usbcunet.dll" (ByVal dat As Long, ByVal adr As Long, ByVal siz As Long)
'パラレル書き込み。占有したブロックのみ。
' dat:書き込みデータ
' adr:バンク番号(2000~)
' siz:書き込みサイズ

Declare Function cunet_sw Lib "usbcunet.dll" (ByVal adr As Long) As Long
'ビット読み込み。全てのブロックを参照可能。
' adr:ビット番号(2000~)
' 戻値:1=オン、0 オフ

Declare Sub cunet_on Lib "usbcunet.dll" (ByVal adr As Long)
'ビットオン。占有したブロックのみ。
' adr:ビット番号(2000~)
' 戻値:無し

Declare Sub cunet_off Lib "usbcunet.dll" (ByVal adr As Long)
'ビットオフ。占有したブロックのみ。
' adr:ビット番号(2000~)
' 戻値:無し

'================================================================
'定数
'================================================================

'cunet_in、cunet_out のサイズ
Public Const Cu_Chr = 1  '1byte 符号無
Public Const Cu_Int = 2  '2Byte 符号付
Public Const Cu_Wrd = 4  '2Byte 符号無
Public Const Cu_Lng = 8  '4Byte 符号付
```

■デモ機 CUnet I/O MAP

機器	SA	UCnet Address	Offset	IN/OUT バンク番号	ON/OFF/SW ビット番号	内容
PC (VB)	0	00	0	2000	2000~2007	
		01	1	2001	2008~2015	
		02	2	2002	2016~2023	
		03	3	2003	2024~2031	
		04	4	2004	2032~2039	
		05	5	2005	2040~2047	
		06	6	2006	2048~2055	
		07	7	2007	2056~2063	
	1	08	0	2008	2064~2071	
		09	1	2009	2072~2079	
		10	2	2010	2080~2087	
		11	3	2011	2088~2095	
		12	4	2012	2096~2103	
		13	5	2013	2104~2111	bit2104 インターロック 座標読込完了
		14	6	2014	2112~2119	
		15	7	2015	2120~2127	
	2	16	0	2016	2128~2135	MPC#2 出力
		17	1	2017	2136~2143	MPC#2 出力
		18	2	2018	2144~2151	
		19	3	2019	2152~2159	
		20	4	2020	2160~2167	
		21	5	2021	2168~2175	
		22	6	2022	2176~2183	
		23	7	2023	2184~2191	
	3	24	0	2024	2192~2199	
		25	1	2025	2200~2207	
		26	2	2026	2208~2215	
		27	3	2027	2216~2223	
		28	4	2028	2224~2231	
		29	5	2029	2232~2239	
		30	6	2030	2240~2247	
		31	7	2031	2248~2255	
MPC #1	4	32	0	2032	2256~2263	X 軸座標
		33	1	2033	2264~2271	
		34	2	2034	2272~2279	
		35	3	2035	2280~2287	
		36	4	2036	2288~2295	Y 軸座標
		37	5	2037	2296~2303	
		38	6	2038	2304~2311	
		39	7	2039	2312~2319	
	5	40	0	2040	2320~2327	Z 軸座標
		41	1	2041	2328~2335	
		42	2	2042	2336~2343	
		43	3	2043	2344~2351	
		44	4	2044	2352~2359	bit2352 チャック状態
		45	5	2045	2360~2367	bit2360 インターロック 座標書込完了
		46	6	2046	2368~2375	
		47	7	2047	2376~2383	
	6	48	0	2048	2384~2391	MPC#2 出力
		49	1	2049	2392~2399	MPC#2 出力
		50	2	2050	2400~2407	
		51	3	2051	2408~2415	
		52	4	2052	2416~2423	
		53	5	2053	2424~2431	
		54	6	2054	2432~2439	
		55	7	2055	2440~2447	
	7	56	0	2056	2448~2455	(MPC#2 入力)
		57	1	2057	2456~2463	
		58	2	2058	2464~2471	
		59	3	2059	2472~2479	
		60	4	2060	2480~2487	
		61	5	2061	2488~2495	
		62	6	2062	2496~2503	
		63	7	2063	2504~2511	

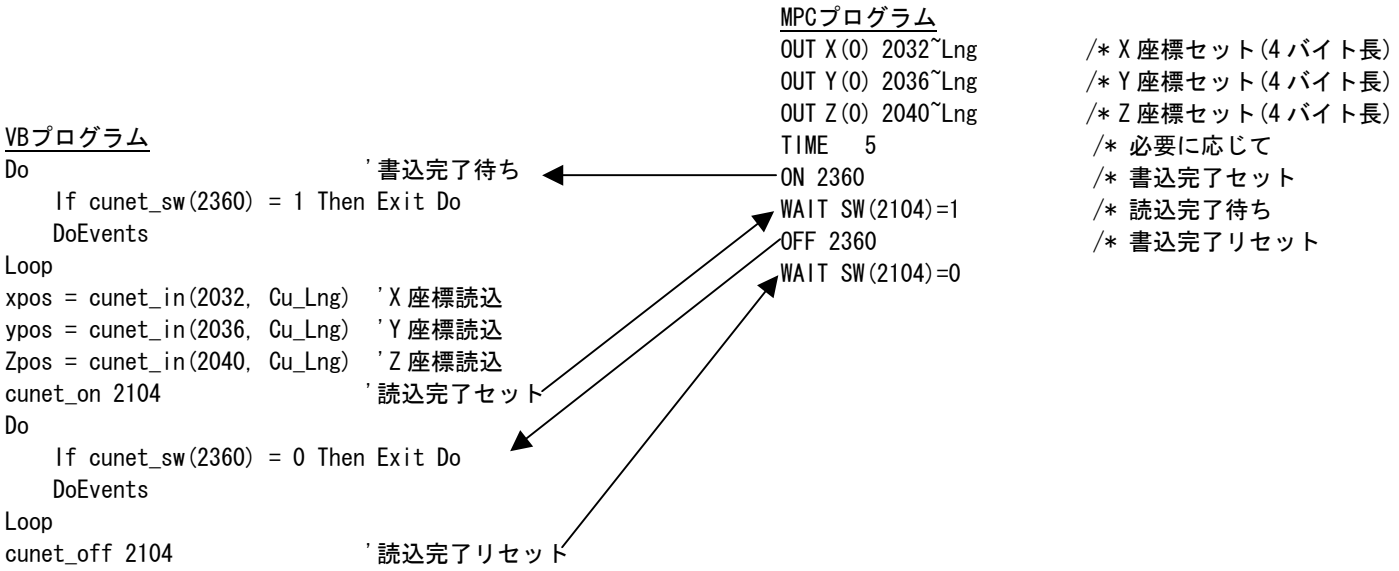
機器	SA	CUnet Address	Offset	IN/OUT バンク番号	ON/OFF/SW ビット番号	プログラム例
MPC #2	8	64	0	2064	2512~2519	MPC#2 入力
		65	1	2065	2520~2527	
		66	2	2066	2528~2535	
		67	3	2067	2536~2543	
		68	4	2068	2544~2551	
		69	5	2069	2552~2559	
		70	6	2070	2560~2567	
		71	7	2071	2568~2575	
	9	72	0	2072	2576~2583	
		73	1	2073	2584~2591	
		74	2	2074	2592~2599	
		75	3	2075	2600~2607	
		76	4	2076	2608~2615	
		77	5	2077	2616~2623	
		78	6	2078	2624~2631	
		79	7	2079	2632~2639	
	10	80	0	2080	2640~2647	MPC#2 入力
		81	1	2081	2648~2655	
		82	2	2082	2656~2663	
		83	3	2083	2664~2671	
		84	4	2084	2672~2679	
		85	5	2085	2680~2687	
		86	6	2086	2688~2695	
		87	7	2087	2696~2703	
	11	88	0	2088	2704~2711	
		89	1	2089	2712~2719	
		90	2	2090	2720~2727	
		91	3	2091	2728~2735	
		92	4	2092	2736~2743	
		93	5	2093	2744~2751	
		94	6	2094	2752~2759	
		95	7	2095	2760~2767	

「SA(Station Address)」　8 バイト=1 ブロックでの通し番号。
　　MPC の CUNET と usbcunet.dll の cunet_init コマンドの引数にはこの値を使います。
「CUnet Address」　バイトの通し番号
「Offset」　ブロック内のバイト番号。
　　MPC プログラムで予約定数 SAm_B+n を使ったときの n になります。
「IN/OUT バンク番号」　パラレル入出力の番号。
「ON/OFF/SW ビット番号」　ビット入出力の番号

SA、IN/OUT バンク番号、ON/OFF/SW ビット番号 は MPC、usbcunet.dll 共通です。

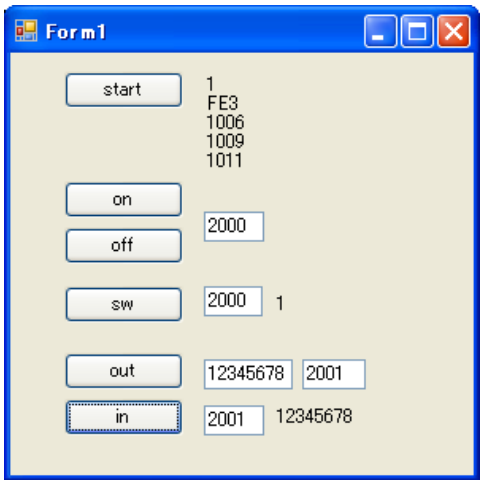
■2 バイト長以上のデータの読み書き

MPC および usbcunet.dll と UCnet(MKY40)間のデータは 8 ビットアクセスです。
グローバルメモリーに反映させるまでにバイト単位でタイムラグが発生することが想定されますので、2 バイト長以上のデータの授受を行う場合にはインターロックを推奨します。



（実用上もデータ授受のタイミングを計るためにインターロックは必要です。）

■VB2008 Express Edition での usbcunet.dll 使用例



VB6 と Express Edition はデータ型のサイズが異なります。

VB6 ： 32bit = Long

Express Edition ： 32bit = Integer

◆Module1.vb ファイル

```
Module Module1
    Declare Function cunet_usb_open Lib "usbcunet.dll" () As Integer
    Declare Function cunet_fw_vid Lib "usbcunet.dll" () As Integer
    Declare Function cunet_fw_pid Lib "usbcunet.dll" () As Integer
    Declare Function cunet_dll_ver Lib "usbcunet.dll" () As Integer
    Declare Function cunet_fw_ver Lib "usbcunet.dll" () As Integer
    Declare Sub cunet_init Lib "usbcunet.dll" (ByVal sa As Integer, ByVal ow As Integer, ByVal en As Integer)

    Declare Function cunet_in Lib "usbcunet.dll" (ByVal adr As Integer, ByVal siz As Integer) As Integer
    Declare Sub cunet_out Lib "usbcunet.dll" (ByVal dat As Integer, ByVal adr As Integer, ByVal siz As Integer)
    Declare Sub cunet_on Lib "usbcunet.dll" (ByVal adr As Integer)
    Declare Sub cunet_off Lib "usbcunet.dll" (ByVal adr As Integer)
    Declare Function cunet_sw Lib "usbcunet.dll" (ByVal adr As Integer) As Integer

    Public Const Cu_Chr As Integer = 1
    Public Const Cu_Int As Integer = 2
    Public Const Cu_Wrd As Integer = 4
    Public Const Cu_Lng As Integer = 8

End Module
```

◆Form1.vb ファイル

```
Public Class Form1
    Private Sub Button_start_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button_start.Click
        Label1.Text = cunet_usb_open
        Label2.Text = Hex$(cunet_fw_vid)
        Label3.Text = Hex$(cunet_fw_pid)
        Label4.Text = cunet_fw_ver
        Label5.Text = cunet_dll_ver
        cunet_init(0, 4, 12)
    End Sub

    Private Sub Button_on_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button_on.Click
        cunet_on(TextBox1.Text)
    End Sub

    Private Sub Button_off_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button_off.Click
        cunet_off(TextBox1.Text)
    End Sub

    Private Sub Button_sw_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button_sw.Click
        Label6.Text = cunet_sw(TextBox2.Text)
    End Sub

    Private Sub Button_out_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button_out.Click
        cunet_out("&H" + TextBox3.Text, TextBox4.Text, Cu_Lng)
    End Sub

    Private Sub Button_in_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button_in.Click
        Label7.Text = Hex$(cunet_in(TextBox5.Text, Cu_Lng))
    End Sub

End Class
```

■Delphi での usbcunet.dll 使用例

◆宣言

```
  function cunet_usb_open(): Integer; stdcall; external 'usbcunet.dll';
  function cunet_fw_vid(): Integer; stdcall; external 'usbcunet.dll';
  function cunet_fw_pid(): Integer; stdcall; external 'usbcunet.dll';
  function cunet_fw_ver(): Integer; stdcall; external 'usbcunet.dll';
  function cunet_dll_ver(): Integer; stdcall; external 'usbcunet.dll';
  procedure cunet_init(sa, ow, en :integer); stdcall; external 'usbcunet.dll' ;
  function cunet_in(adr, siz :integer): Integer; stdcall; external 'usbcunet.dll';
  procedure cunet_out(dat, adr, siz :integer); stdcall; external 'usbcunet.dll';
  function cunet_sw(adr:integer): Integer; stdcall; external 'usbcunet.dll';
  procedure cunet_on(adr :integer); stdcall; external 'usbcunet.dll';
  procedure cunet_off(adr :integer); stdcall; external 'usbcunet.dll';
```

◆使用例

```
procedure TForm1.Button1Click(Sender: TObject);
begin
        if cunet_usb_open()<>1 then
        begin
                memo1.lines.add('usb open failed');
                exit;
        end;
        memo1.Lines.Add('successful open usb' );
        memo1.Lines.Add('fw vid 0x'+inttohex(cunet_fw_vid(),4));
        memo1.Lines.Add('fw pid 0x'+inttohex(cunet_fw_pid(),4));
        memo1.Lines.Add('fw ver '+inttostr(cunet_fw_ver()));
        memo1.Lines.Add('dll ver '+inttostr(cunet_dll_ver()));
        cunet_init(strtoint(edit6.text),strtoint(edit7.text),strtoint(edit8.text));
end;

procedure TForm1.Button2Click(Sender: TObject);
var
i:integer;
begin
        i:=cunet_in(strtoint(trim(edit1.text)),1);
        label1.caption:='0x'+inttohex(i,2);
end;

procedure TForm1.Button3Click(Sender: TObject);
var
dt,adr:integer;
begin
        dt:=strtoint('$'+trim(edit2.text));
        adr:=strtoint(trim(edit3.text));
        cunet_out(dt,adr,1);
end;

procedure TForm1.Button4Click(Sender: TObject);
var
i:integer;
begin
        i:=cunet_sw(strtoint(trim(edit4.text)));
        label3.caption:=inttostr(i);
end;

procedure TForm1.Button5Click(Sender: TObject);
var
adr:integer;
begin
        adr:=strtoint(trim(edit5.text));
        cunet_on(adr);
end;

procedure TForm1.Button6Click(Sender: TObject);
var
adr:integer;
begin
        adr:=strtoint(trim(edit5.text));
        cunet_off(adr);
end;
```

■MPC-CUnet、USB-CUnet の LED

MPC-CUnet、USB-CUnet 共通

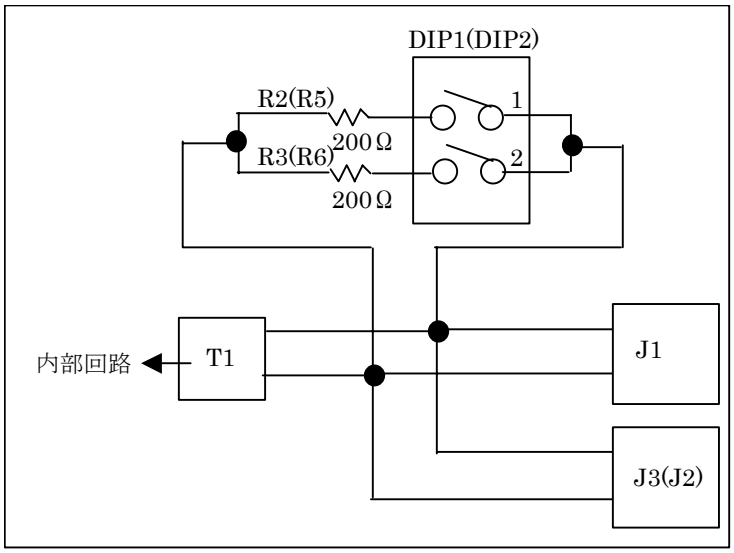
LED1 (R)	メンバー減少表示 (MKY40 #MCARE)	メンバステータスが減少したときに、約50ms 点灯。1サイクルが50ms 以内の場合、数サイクルに渡って 1 つのパルスが出力されることになります。
		点灯の可能性： 外来ノイズによる通信障害。
LED2 (R)	リンク切れ表示 (MKY40 #LCARE)	一旦リンクが成立していた装置が、リンク不成立になったことを検出した時に、約 50ms 点灯。1 サイクルが 50ms 以内の場合、数サイクルに渡って 1 つのパルスが出力されることになります。
		点灯の可能性：(正常に共有動作を継続しているステーションで点灯) 通信ケーブルの断線、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 (MKY40 #STB)	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 (MKY40 #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)

※参考：「Let's Try! CUnet フィールドエンジニア編」
・ネットワークにおいて異常が発生した場合、どうなりますか?
・MEM モードの MKY40 接続概要図
など

USB-CUnet

LED5 (G)	USB 電源	USB 通電中点灯
LED6 (G)	MKY40 電源	MKY40 通電中点灯
LED7 (G)	CPU(CY7C68013)動作	ファームウェア動作中点滅
LED8 (G)	USB アクセス	PC が USB アクセス中点灯

■MPC-CUnet、USB-CUnet の終端抵抗



※()内は USB-CUnet のロケーション

※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)

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の利用を推奨

※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。

① 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。

② 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。

③ 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

■出典、資料

「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。

MPC-CUnet、USB-CUnet は MKY40 の応用製品であり、MKY40 の全機能が使えるわけではありません。