# 蛍光表示管モジュール

# 「CU-Y シリーズ モジュール」

## アプリケーションノート

APF500　R1.00　2012 年 3 月 8 日

itron はノリタケ蛍光表示管の登録商標です。 蛍光表示管は、青緑色発光で目にやさしい自発光タイプの表示素子です。 液晶（ＬＣＤ）やＬＥＤなどの他の表示素子に比べて、高い視認性、広い動作温度範囲などを特長とします。 CU-Yシリーズは、高精細化により従来品種と比べて同じ面積で表示可能文字数を増やした蛍光表示管モジュールです。

本書は表示モジュールをご使用いただく上での技術サポート資料として準備いたしましたのでご利用下さい。

## 目次

## 1. 概要

高精細でコンパクトな文字表示用モジュールです。


CU24043-Y1A

### 1.1. 文字の大きさについて

蛍光表示管は高輝度自発光型のディスプレイですので、小さな文字で高い視認性を得ることができます。CU-Y シリーズはこの特徴を生かして狭い表示領域で従来と比較して多くの文字数を表示できるように設計しました。

文字の寸法と視認性につきましては、仕様書の数値からではなく現品をご覧いただき蛍光表示管の読みやすさをお確かめ頂ければ幸いです。

## 2. 蛍光表示管モジュールラインナップ

### 2.1. 品名について

品名で製品の概要がわかります。

品名例：CU200211-Y100

CU ：キャラクタ表示専用シリーズ
20 ：20 桁
02 ：２行
11 ：字高 11mm
-Y100 ：CU-Y シリーズ
インターフェイス等仕様を表します。
Y1A：シリアルインターフェイスが C-MOS レベル。
Y100：シリアルインターフェイスが RS-232C レベル
YX1A：C-MOS レベルシリアルインターフェイスで高輝度品。
YX100：RS-232C シリアルインターフェイスで高輝度品。

## 2.2. 標準シリーズ系統図

CU-Y シリーズ以外の表示器に関しましては、各々のアプリケーションノート並びに製品ごとの仕様書を参照してください。

ノリタケ蛍光表示管モジュール

- ├ カスタムモジュール
- ├ 標準 CU シリーズ
  - |
  - ├ CU-T, CU-TW シリーズ
  - | 小型～中型サイズの文字表示シリーズ。
  - | 永くご愛用頂いているシリーズです。
  - |
  - ├ CU-U, CU-W, CU-UW シリーズ
  - | 標準的な文字タイプの LCD 表示器の外形とコマンドに仕様を合わせた
  - | シリーズです。
  - |
  - | CU-U は民生用温度範囲品。CU-W は産業用温度範囲品です。
  - | コントローラの世代交代により CU-UW シリーズに移行統合しています。
  - | UW には、シリアルインターフェイスが追加されています。
  - |
  - └ CU-Y シリーズ
    CU-UW の外形から高精細化により文字数を増やしたシリーズです。
    強力なコマンド群を装備し機能 UP を図っています。
- |
- └ 標準 GU シリーズ
  連続ドットで切れ目のない表示が可能なシリーズです。
  英数、漢字フォントを搭載し文字表示端末としてご利用頂ける品種も揃えております。

### 2.3. 対象品種

本資料はＹシリーズの標準仕様につき説明していますので、ここにない品種についても大きさを読み替えてご利用ください。

CU20027-YX100,　CU20027-YX1A

CU200211-Y100,　CU200211-Y1A
CU20027-Y100,　CU20027-Y1A
CU22042-Y100,　CU22042-Y1A
CU24043-Y100,　CU24043-Y1A
CU24063-Y100,　CU24063-Y1A

## 3. ハードウエア

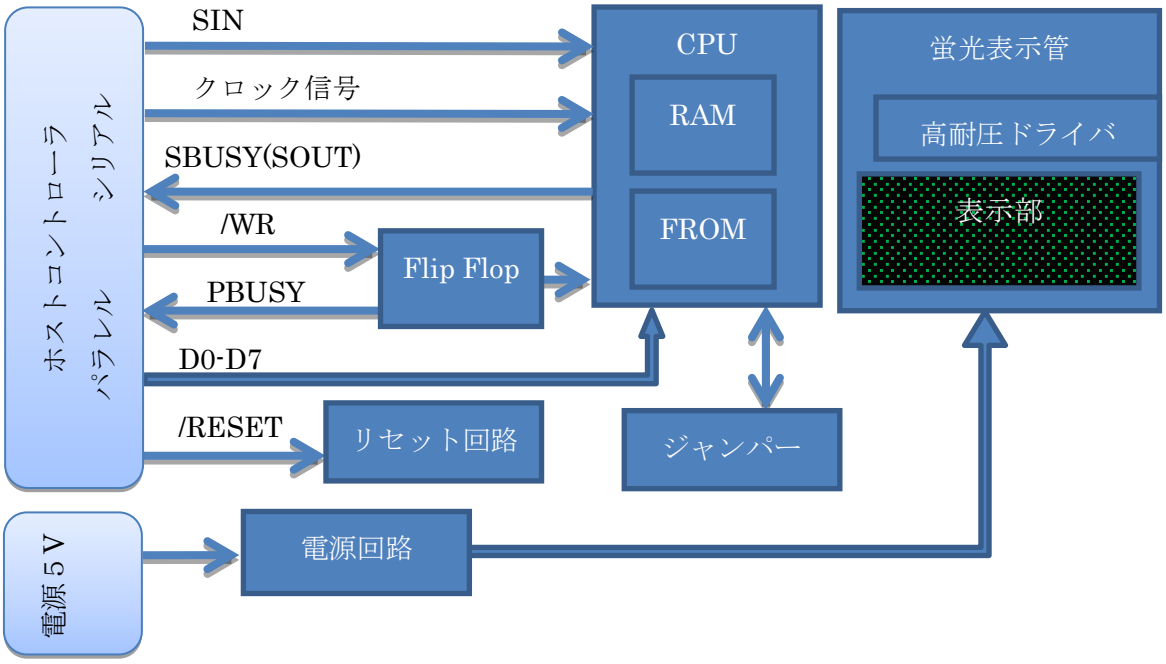
### 3.1. 外形比較

外形寸法の比較表です。他シリーズからの置き換え等の参考にしてください。

| 品名 | 文字数<br>文字高 | 表示エリア | 外形 | 外形相当品 |
|---|---|---|---|---|
| CU20027-YX100<br>CU20027-YX1A<br>CU20027-Y100<br>CU20027-Y1A | 20 文字 2 行<br>7.19mm | 73.4 x 15 | 116.0 x 37.0 | CU20025-UW1J<br>GU140X16G-7x0x |
| CU200211-Y100<br>CU200211-Y1A | 20 文字 2 行<br>10.51mm | 23.6 x 159 | 190.0x55.0 | CU20029-TE200K<br>CU200211-TE200A<br>(CU40045-UW1J:<br>若干異なります。) |
| CU22042-Y100<br>CU22042-Y1A | 22 文字 4 行<br>2.348mm | 52.5x11.5 | 80.0x36.0 | CU16025-UW6J<br>GU112X16G-700 x |
| CU24043-Y100<br>CU24043-Y1A | 24 文字 4 行<br>3.34mm | 73.4x14.95 | 116.0x37.0 | CU20025-UW1J<br>GU140X16G-700 x |
| CU24063-Y100<br>CU24063-Y1A | 24 文字 6 行<br>3.34mm | 70.38x42.5 | 98.0x47.0 | GU140X32F-705x |

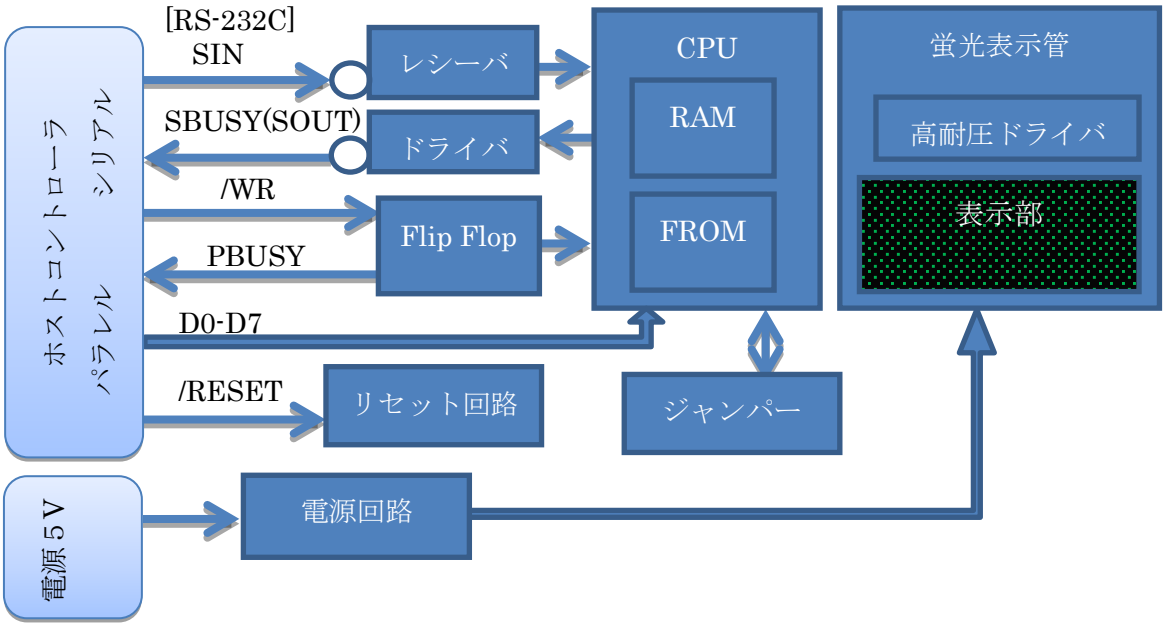
### 3.2. ブロック図

CU-Y1A, CU-YX1A




シリアルのクロック信号は、同期式シリアルの時に使用します。
非同期シリアルインタフェースを使用中に「情報読み出しモード」に移行すると SBUSY 信号が SOUT(シリアル出力)機能に変わります。

CU-Y100, CU-YX100




「情報読み出しモード」に移行すると SBUSY 信号が SOUT(シリアル出力)機能に変わります。

### 3.3. 取り付け、フィルター、

ケースや筐体への取り付けはプリント基板が反らないように取り付けてください。反りがあると表示管にダメージが加わり永久破壊に至る可能性があります。

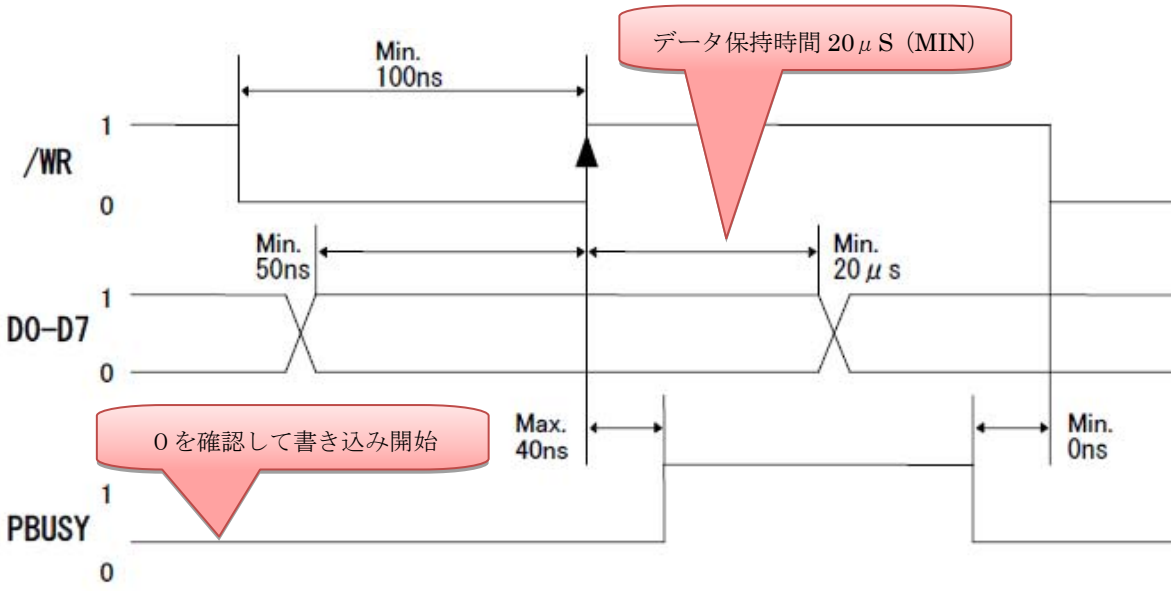
表示管の前面に着色した透明な樹脂板を置いてください。光学的なフィルターとなってコントラストを改善すると共に、表示器の保護となります。
また、色は赤を除いたいろいろな色に対応していますので、装置のイメージにあった色をお選び頂けることと思います。

### 3.4. インターフェイス、

ホストインターフェイスとしてシリアルインターフェイスとパラレルインターフェイスを用意しています。使用しないインターフェイスについて特になにもする必要はありません。
併用も可能ですが、同時には書きこまないでください。

#### 3.4.1. パラレルインターフェイス

書込みのみ可能です。　読み出しはできません。
PBUSY=0 (Low レベル) を確認して 8 Bit パラレルデータを D0-D7 (MSB) に与えてから/WR に負パルスを与えます。データホールド時間が 20μS 必要ですのでご注意ください。長くなるのは差し支えありませんので、次の書き込み時までそのままにしておいて結構です。




データ損失を防ぐ為、PBUSY=0の状態でデータ書き込みを行ってください。

### 3.4.2. シリアルインターフェイス

PC 等と接続するための RS-232C レベル（非同期のみ）の品種（CU-Y100, CU-YX100）と組み込み用コントローラに接続するための C-MOS レベル（同期、非同期）の品種(CU-Y1A, CU-YX1A)に分かれます。
両者の違いはインターフェイスのみでコマンドや文字コードは同じです。
原則的に「ハードウエアハンドシェークを行う書き込み」のシリアルインターフェイスです。

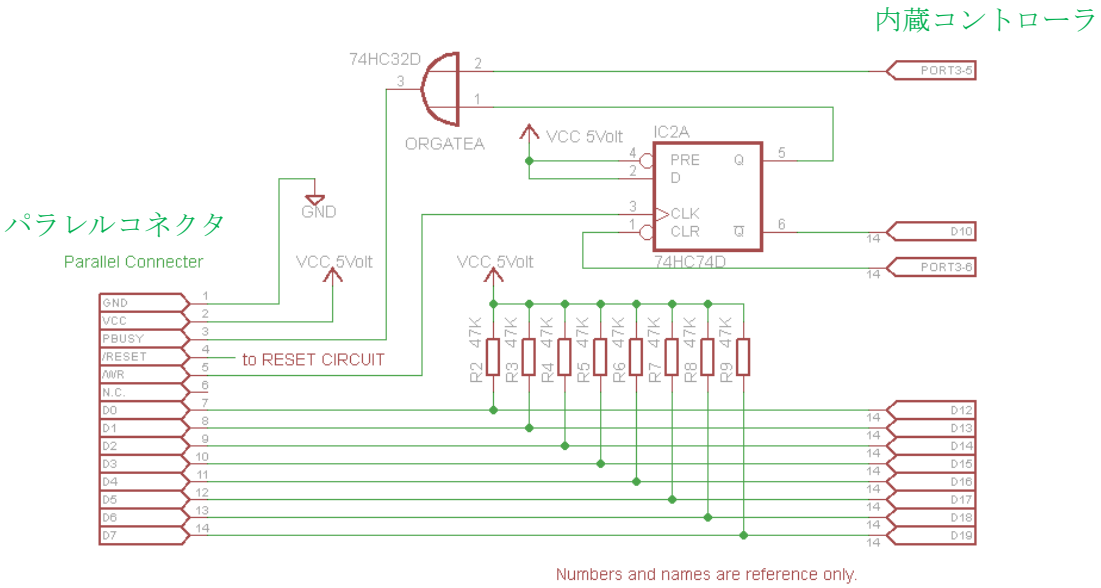
非同期シリアルの場合のみ、「情報読み出しモード」に移行するとハンドシェーク無しの双方向シリアルインターフェイスになります。　出力は、「SBUSY」に出力されます。

## 3.5. 入出力等価回路（表示器内部回路）

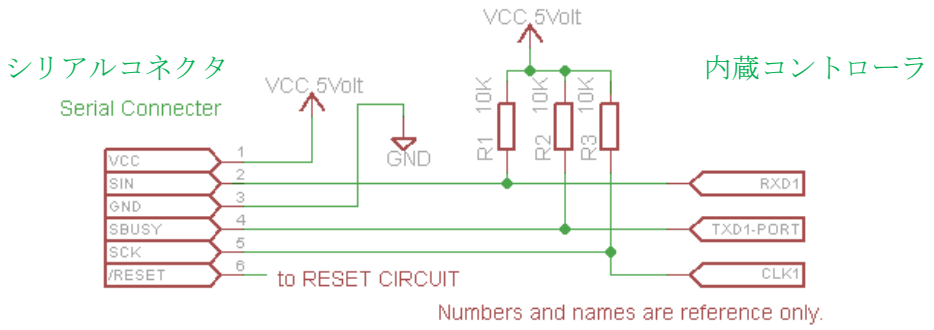
入出力の等価回路を以下に示します。
部品名や番号等は参考用であり実機とは異なる場合があります。

### 3.5.1. パラレルインターフェイス（CU-Y(X)1A, CU-Y(X)100 共通）

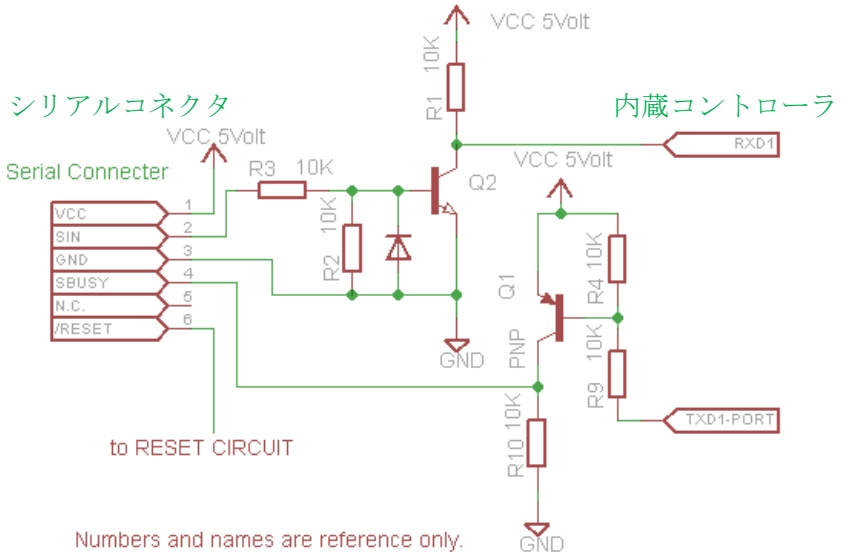
（表示器内部回路）



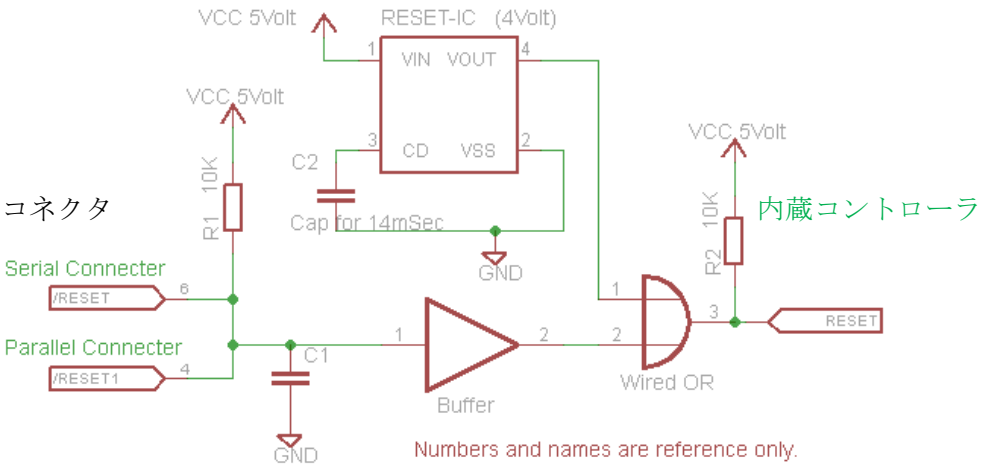

### 3.5.2. C-MOS シリアルインターフェイス（CU-Y1A, CU-YX1A）（表示器内部回路）



### 3.5.3. RS-232C レベルシリアルインターフェイス（CU-Y100, CU-YX100）（表示器内部回路）



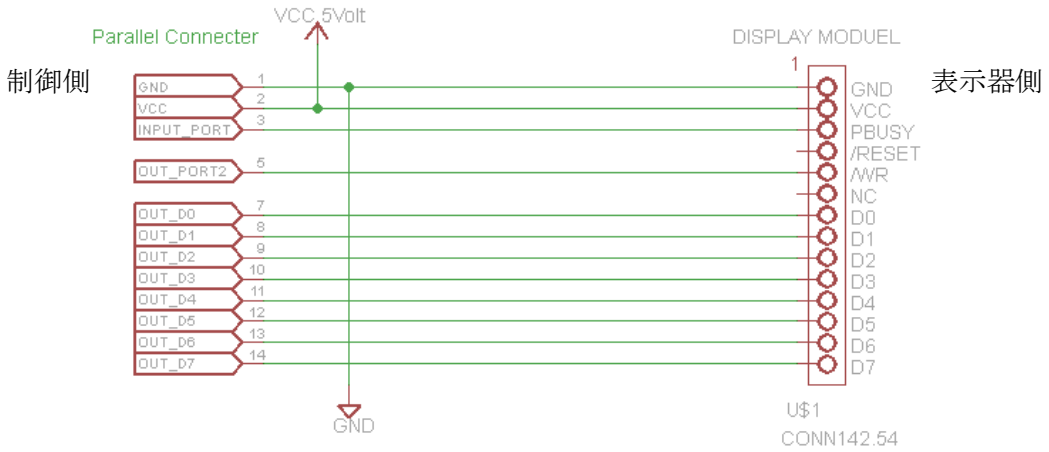

### 3.5.4. リセット回路（表示器内部回路）



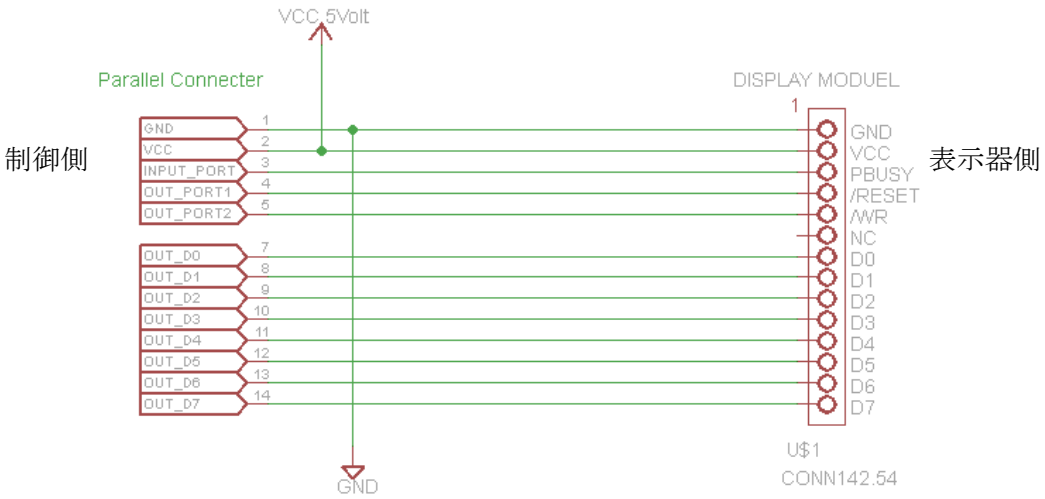

## 4. 接続例

### 4.1. パラレルインターフェイス接続例

#### 4.1.1. パラレル接続例　リセット無

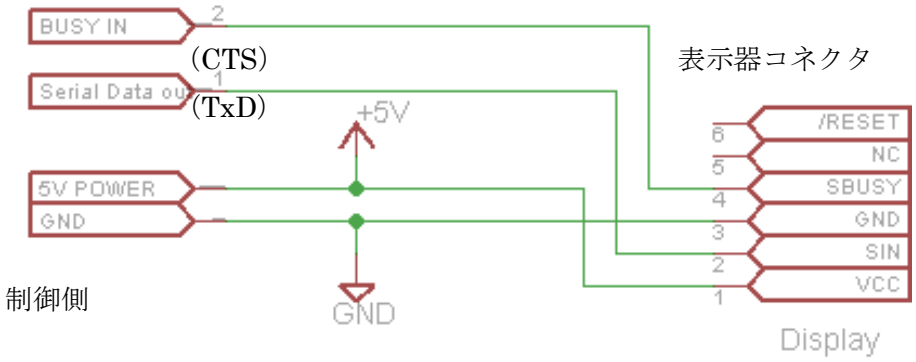


### 4.1.2. パラレル接続例　リセット有



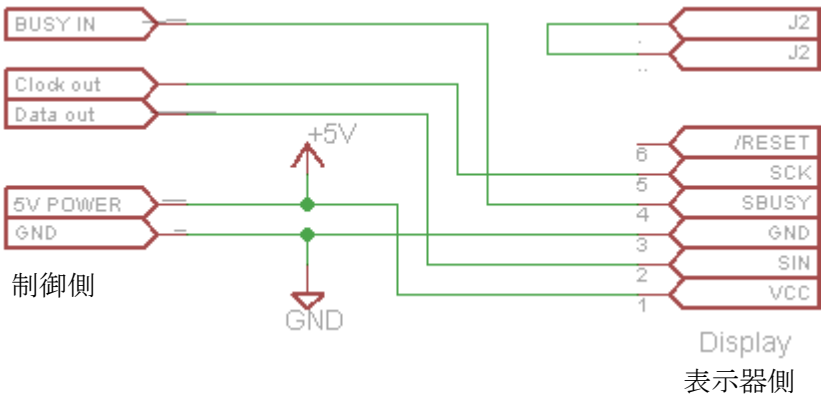

## 5. シリアル接続例

### 5.1.1. 組み込み CPU から CU-Y1A, CU-YX1A タイプへの接続例　調歩同期シリアル

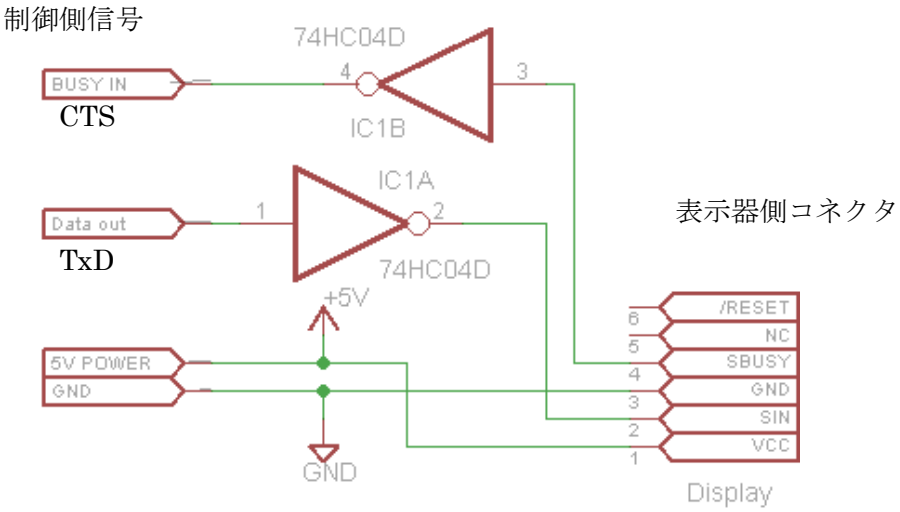



### 5.1.2. 組み込み CPU から CU-Y1A, CU-YX1A タイプへの接続例　同期シリアル

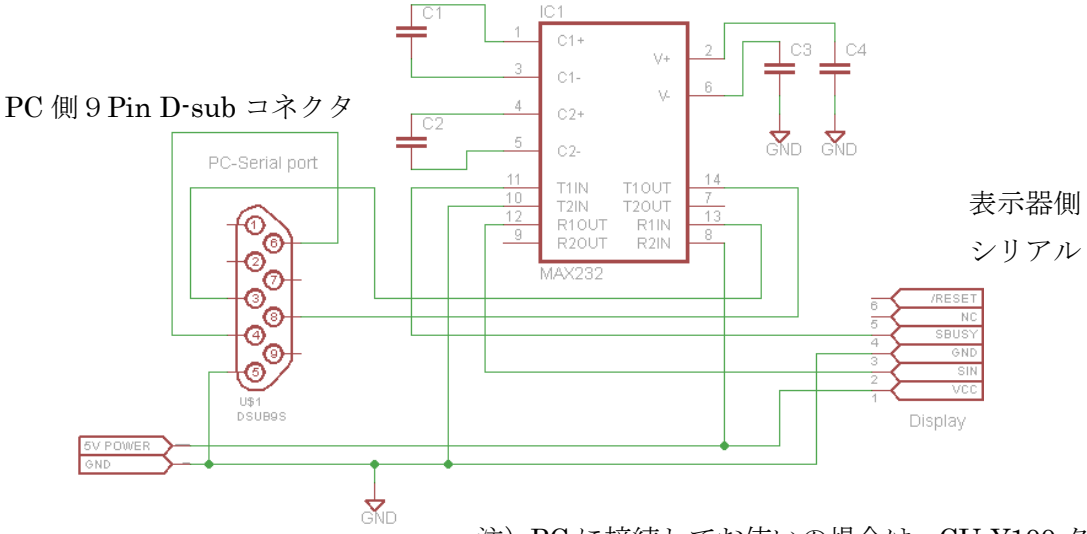
同期式シリアルでご使用の場合、J2 をショートしてください。



### 5.1.3. 組み込み CPU から CU-Y100, CU-YX100 タイプへの接続例

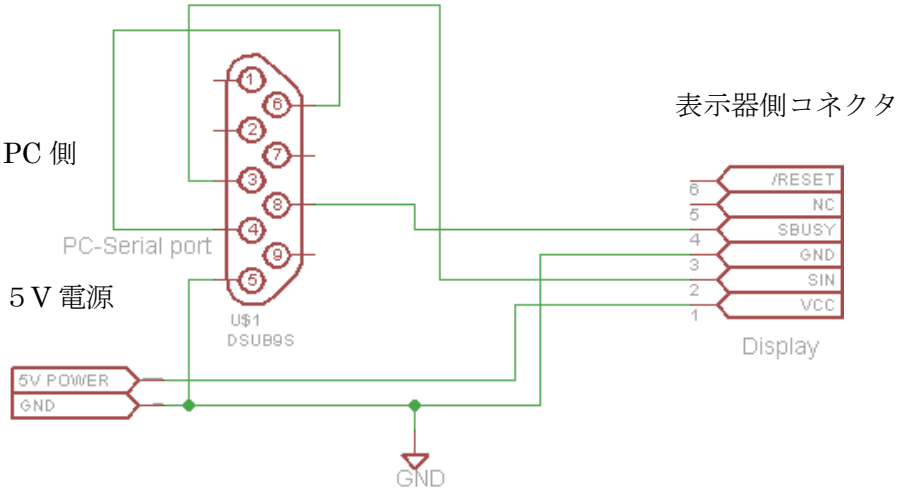



### 5.1.4. PC（COM ポート）から CU-Y1A, CU-YX1A タイプへの接続例




注）PC に接続してお使いの場合は、CU-Y100 タイプをお薦めします。

### 5.1.5. PC（COM ポート）から CU-Y100, CU-YX100 タイプへの接続例



### 5.1.6. ソフトウエア

CU-Y シリーズ表示器は内蔵コントローラの初期化機能により面倒な初期設定なしで基本的な動作を行います。　電源を投入したらそのまま文字コードを書き込めば表示を行います。　組み込まれた機能につきましても、文字コードと同様にコマンドを順次書き込んでいただければ、その機能をご利用いただけます。

## 5.2. 主な機能

CU-Y シリーズに搭載した主な機能を紹介します。分りやすさをモットーに詳細を省略しています。厳密な動作につきましては、仕様書を参照いただくようお願いします。

### 5.2.1. 基本動作

CU-Y シリーズ表示管は、表示端末として動作するように設計されています。
使い方はタイプライターをイメージしてください。
文字を書き込むと順次表示されます。次の書き込む位置をカーソル位置と呼びます。
カーソルセットコマンド（24Hex）を送ってカーソル位置を移動してから文字を書くことで、好きな位置に文字を表示させることができます。
次のコマンドは、よく使うカーソル移動を１バイトで行います。

| | |
|---|---|
| バックスペース | 左に１文字分移動 |
| ホリゾンタルタブ | 右に１文字分移動 |
| ラインフィード | 下に１行分移動 |
| ホームポジション | 左上隅に移動 |
| キャリッジリターン | 同じ行の左端に移動 |

### 5.2.2. 文字フォント操作

１バイトの文字で表現できるのは 20Hex から 7FHex と A0Hex から FFHex の 192 文字です。これでは少ないため用途によって一部を変更できるようになっています。変更してもそれ以前に書き込んだ文字に影響はありませんので、異なる文字セットの混在が可能です。
変更には次のようなものがあります。

**国際文字セット(コマンドコード 1Bh 52h n)**

8 ビット JIS の“¥”（円マーク）が ASCII ではバックスラッシュ“\”になることをご存じのかたも多いと思います。このように言語に依って一部の文字の割り当てが変わっています。これに対応するのが国際文字セットです。20Hex から 7FHex のあいだの文字が一部変わります。

**キャラクタテーブル指定／キャラクタフォント指定(コマンドコード 1Bh 74h n)**

もともと ASCII コードは７ビットの文字コードで 80Hex から FFHex は使っていませんでした。そこで 8 ビットの JIS コードでは、ここにカタカナを割り当てました。ヨーロッパではα、βなどのギリシャ文字やその言語特有の文字を割り当てています。
CU-Y シリーズ表示では、80Hex～FFHex の文字テーブルを複数用意し切り替えて使

えるようになっています。

**RAM フォント**

最大 16 文字を定義できます。これを使うためには、RAM フォント定義コマンドでフォントをダウンロードしてから、RAM ユーザーフォント指定コマンドで RAM フォント有効を設定してください。RAM に記憶するため電源オフやリセットで消えてしまいます。

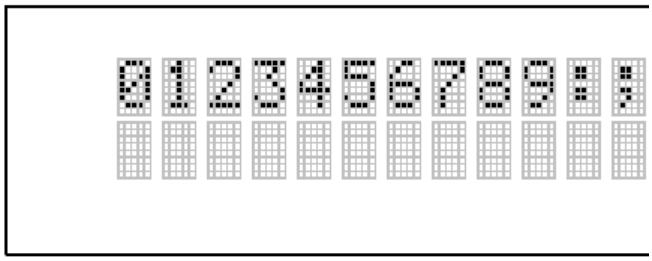
**FROM ユーザーフォント定義**

20Hex から FFHex までを使って 224 文字を一括して定義できます。部分的な差し替えはできません。FROM ユーザーフォントを定義するときは、FROM 書き換え用のモード（ユーザー設定モード）に切り替えます。FROM の書き換え回数には制限があります。またユーザーモードで電源オフなどで中断すると FROM に記憶されている表示器の制御プログラム(ファームウエア)が変わってしまい復旧できなくなる場合があります。このため FROM 書き変えは保守時など最低限とし、例えば電源投入時の初期化で毎回 FROM を書き変えるような運用は避けて頂くようお願いします。

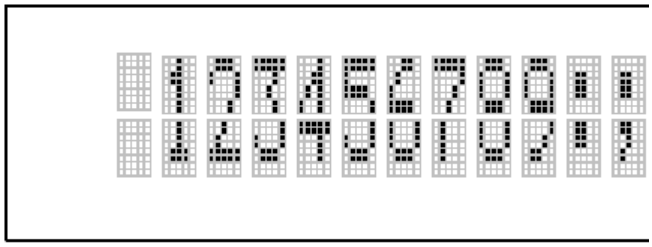
**拡大文字**

CU-Y シリーズは拡大文字機能を搭載しております。

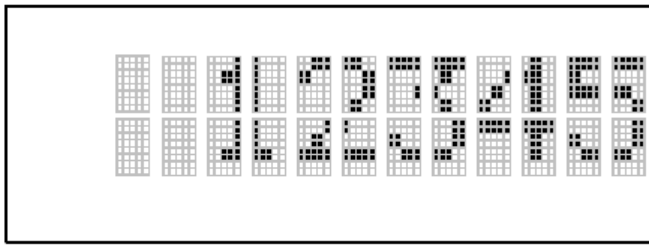
**拡大倍率：1×1倍**



**拡大倍率：1×2倍**



**拡大倍率：2×2倍**



### 5.2.3. 輝度調整

コマンドにより全画面の輝度調整が可能です。さらにキャラクタ諧調レベル設定で文字単位で諧調表示を行うことができます。諧調を設定してから文字を書き込むと、その文字の明るさが変わります。諧調設定以前に書かれた文字の明るさは変わりません。

### 5.2.4. スクロール設定

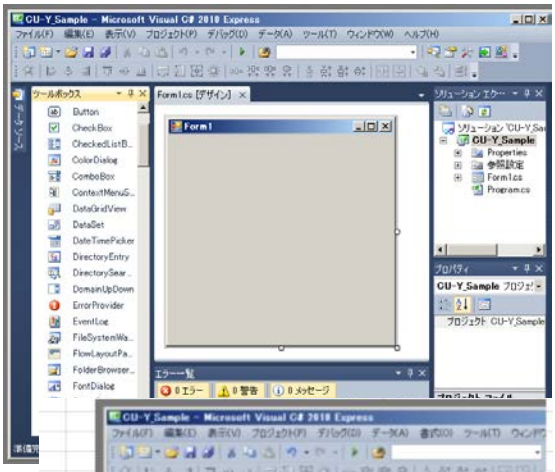
表示可能な文字数より多くの文字を表示したい場合、画面をスクロールさせるのはよく行われます。CU-Y シリーズはスクロール機能を内蔵しており、縦スクロールまたは横スクロールを設定してから文字を書き込むと、カーソルが画面端に届いた以降の表示を自動的にスクロールしながら行います。スクロールさせるためには、文字を書き込む必要があることにご注意ください。

## 5.3. サンプルプログラム１（Microsoft Visual C#2010）

WindowsPC とのシリアル接続例についてサンプルプログラムを用意しました。開発環境は Microsoft 社の提供している評価版を使用していますので、お気軽にお試しください。

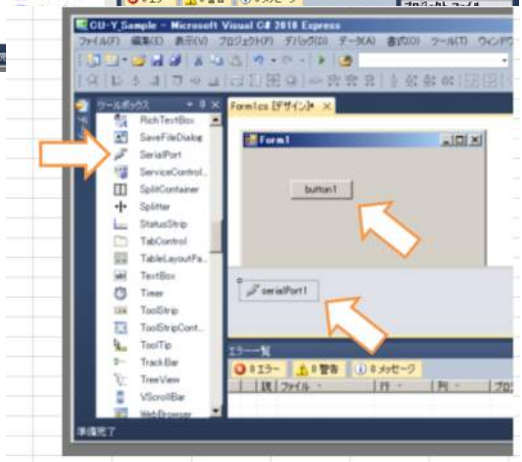
まず、評価版 C#2010 を Microsoft 社のサイトからダウンロードしてインストールしておきます。
次にファイルメニューから新しいプロジェクトを選びます。“Windows フォームアプリケーションをハイライトして、名前を記入し（初期値の WindowsFormsApplication1 のままでも構いません。ここでは“CU_Y_Sample”にしました。）OK をクリックします。

右のようなウィンドウが開きます。

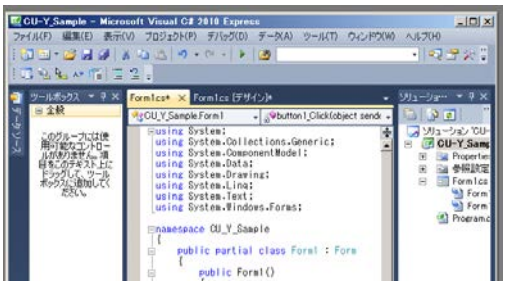


ここで、ツールボックスで Button をハイライトして Form1.cs[デザイン]にある Form1 ウインドウをクリックします。

さらに SerialPort をハイライトして Form1 ウインドウをクリックすると右下のようになります。これでウインドウデザインができました。



次に先ほどできた Button1 ボタンをダブルクリックすると Form1.cs というコードを書き込むウインドウが開きます。



ここに、次のようにプログラムを入力します。
グレーの部分は既にできている筈です。グレーでない部分のみを入力してください。

■ Hello World sample List

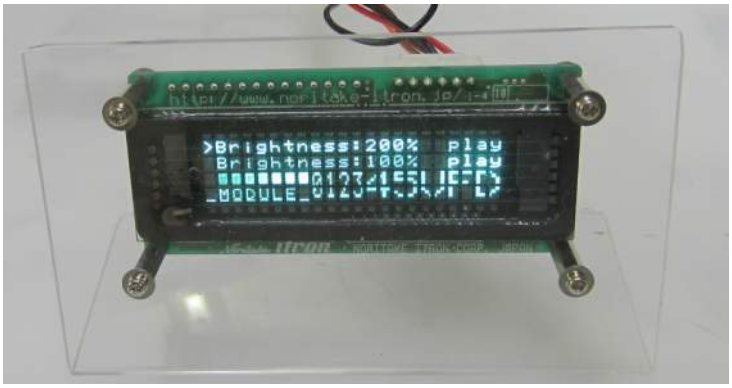
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace CU_Y_Sample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            this.serialPort1.PortName = "COM1";
            this.serialPort1.BaudRate= 38400;
            this.serialPort1.Handshake = System.IO.Ports.Handshake.RequestToSend;
            this.serialPort1.Parity = System.IO.Ports.Parity.None;
            this.serialPort1.StopBits = System.IO.Ports.StopBits.One;
            if ( this.serialPort1.IsOpen == false)
            {
                this.serialPort1.Open();
            }
            this.serialPort1.Write("Hello World");
            this.serialPort1.Close();
        }
    }
}
```

最初の行の”COM1”は、お使いの PC のポート名に合わせて変更してください。
シリアルポートを設定してから OPEN し、”Hello World”と送信しています。入力が終わったら、ファンクションキー“F5”押すかデバッグメニューからデバッグ開始を選択します。
右のようなウインドウが現れますので Button1 をクリックしてください。表示器に”Hello World”と表示される筈です。


### 5.3.1. サンプルプログラム２（Microsoft Visual C#2010）

表示器の色々な機能を使った右の表示をさせるサンプルプログラムを用意しました。COM1 はお使いの PC の設定に書き直してお使いください。




表示器の機能は CU-Y クラスにまとめてあります。

＝リスト１ Form1.cs＝

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace CU_Y_Sample
{
    public partial class Form1 : Form
    {
        const int Brightness25_Percent = 1; //  画面輝度設定の定数
        const int Brightness50_Percent = 2;
        const int Brightness75_Percent = 3;
        const int Brightness100_Percent = 4;
        const int Brightness125_Percent = 5;
        const int Brightness150_Percent = 6;
        const int Brightness175_Percent = 7;
        const int Brightness200_Percent = 8;
        const int Gray00Percent = 1;          //  文字用輝度諧調設定の定数
        const int Gray14Percent = 2;
        const int Gray29Percent = 3;
        const int Gray43Percent = 4;
        const int Gray57Percent = 5;
        const int Gray71Percent = 6;
        const int Gray86Percent = 7;
        const int Gray100Percent = 8;

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.serialPort1.PortName = "COM1";   //   ポート名は適宜直してください。
            this.serialPort1.BaudRate= 38400;
            this.serialPort1.Handshake = System.IO.Ports.Handshake.RequestToSend;
            this.serialPort1.Parity = System.IO.Ports.Parity.None;
            this.serialPort1.StopBits = System.IO.Ports.StopBits.One;
```

```
            if ( this.serialPort1.IsOpen == false)
            {
                this.serialPort1.Open();
            }
            CU_Y Display = new CU_Y();
            byte[] data = new byte[10];
            this.serialPort1.Write(Display.Intialize(),0,Display.bytecount);
            const int Katakana = 1;
            this.serialPort1.Write(Display.CharacterTableSet(Katakana), 0, Display.bytecount);
            this.serialPort1.Write(">Brightness:200%");
            this.serialPort1.Write(Display.CharacterGrayScale(4) , 0, Display.bytecount);
            this.serialPort1.Write("    play");
            this.serialPort1.Write(Display.CursorSet(1,1), 0, Display.bytecount);
            this.serialPort1.Write("Brightness:100%");
            this.serialPort1.Write(Display.CharacterGrayScale(8), 0, Display.bytecount);
            this.serialPort1.Write("    play");
            this.serialPort1.Write(Display.CursorSet(0, 2), 0, Display.bytecount);
            data[0] = 0x94;
            this.serialPort1.Write(Display.CharacterGrayScale(Gray00Percent), 0, Display.bytecount);
            this.serialPort1.Write(data,    0    ,1 );
            this.serialPort1.Write(Display.CharacterGrayScale(Gray14Percent), 0, Display.bytecount);
            this.serialPort1.Write(data, 0, 1);
            this.serialPort1.Write(Display.CharacterGrayScale(Gray29Percent), 0, Display.bytecount);
            this.serialPort1.Write(data, 0, 1);
            this.serialPort1.Write(Display.CharacterGrayScale(Gray43Percent), 0, Display.bytecount);
            this.serialPort1.Write(data, 0, 1);
            this.serialPort1.Write(Display.CharacterGrayScale(Gray57Percent), 0, Display.bytecount);
            this.serialPort1.Write(data, 0, 1);
            this.serialPort1.Write(Display.CharacterGrayScale(Gray71Percent), 0, Display.bytecount);
            this.serialPort1.Write(data, 0, 1);
            this.serialPort1.Write(Display.CharacterGrayScale(Gray86Percent), 0, Display.bytecount);
            this.serialPort1.Write(data, 0, 1);
            this.serialPort1.Write(Display.CharacterGrayScale(Gray100Percent), 0, Display.bytecount);
            this.serialPort1.Write(data, 0, 1);
            this.serialPort1.Write(Display.CharacterGrayScale(Gray43Percent), 0, Display.bytecount);
            this.serialPort1.Write(Display.CharacterSize(1, 2), 0, Display.bytecount);
            this.serialPort1.Write("0123");
            this.serialPort1.Write(Display.CharacterSize(2, 2), 0, Display.bytecount);
            this.serialPort1.Write("45");
            this.serialPort1.Write(Display.CursorSet(16,2), 0, Display.bytecount);
            this.serialPort1.Write("VFD");
            this.serialPort1.Write(Display.CursorSet(0, 3), 0, Display.bytecount);
            this.serialPort1.Write(Display.CharacterSize(1, 1), 0, Display.bytecount);
            this.serialPort1.Write(Display.UnderlineModeSet(true), 0, Display.bytecount);
            this.serialPort1.Write(" MODULE");
            this.serialPort1.Write(Display.ScreenBrightnessSet(Brightness200_Percent), 0, Display.bytecount);
            this.serialPort1.Close();
        }
    }
}
```

＝リスト２ CU-Y.cs ＝
プロジェクトに"CU-Y″という名前で“クラス”を追加します。中に次のようにタイプしてください。
グレー部分は、C#が自動的に生成する部分です。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CU_Y_Sample
{
    class CU_Y
    {
        private byte[] outdata;
        public int bytecount;
        public byte[] Intialize()
        {
            outdata = new byte[2];
            outdata[0] = (byte)0x1b;
            outdata[1] = (byte)0x40;
            bytecount = 2;
            return outdata ;
        }
        public byte[] CharacterGrayScale(int br)
        {
            outdata = new byte[7];
            outdata[0] = (byte)0x1f;
            outdata[1] = (byte)0x28;
            outdata[2] = (byte)0x67;
            outdata[3] = (byte)0x50;
            outdata[4] = (byte)(((br-1) % 8) +1);
            outdata[5] = (byte)0x00;
            outdata[6] = (byte)0x00;
            bytecount = 7;
            return outdata;
        }
        public byte[] CursorSet(int X, int Y)
        {
            outdata = new byte[6];
            outdata[0] = (byte)0x1f;
            outdata[1] = (byte)0x24;
            outdata[2] = (byte)(X % 256);
            outdata[3] = (byte)(X / 256);
            outdata[4] = (byte)(Y % 256);
            outdata[5] = (byte)(Y / 256);
            bytecount = 6;
            return outdata;
        }
        public byte[] InternationalFontSet(int n)
        {
            outdata = new byte[3];
            outdata[0] = (byte)0x1b;
            outdata[1] = (byte)0x52;
            outdata[2] = (byte)(n % 256);
            bytecount = 3;
            return outdata;
        }
```

```
        public byte[] CharacterTableSet(int n)
        {
            outdata = new byte[3];
            outdata[0] = (byte)0x1b;
            outdata[1] = (byte)0x74;
            outdata[2] = (byte)(n % 256);
            bytecount = 3;
            return outdata;
        }
        public byte[] CharacterSize(int X, int Y)
        {
            outdata = new byte[6];
            outdata[0] = (byte)0x1f;
            outdata[1] = (byte)0x28;
            outdata[2] = (byte)0x67;
            outdata[3] = (byte)0x40;
            outdata[4] = (byte)(X % 256);
            outdata[5] = (byte)(Y % 256);
            bytecount = 6;
            return outdata;
        }
        public byte[] UnderlineModeSet(Boolean Flag)
        {
            outdata = new byte[2];
            outdata[0] = (byte)0x1b;
            if (Flag)
            {
                outdata[1] = (byte)0x55;
            }
            else
            {
                outdata[1] = (byte)0x57;
            }
            bytecount = 2;
            return outdata;
        }
        public byte[] ScreenBrightnessSet(int Brightness)
        {
            outdata = new byte[3];
            outdata[0] = (byte)0x1f;
            outdata[1] = (byte)0x58;
            outdata[2] = (byte)(((Brightness-1) % 8)+1);
            bytecount = 3;
            return outdata;
        }
    }
}
```

入力が終わりましたら、サンプル１と同じくメニューからデバッグ開始を選んで、現れたウインドウのボタンをクリックしてください。冒頭の写真のような表示が出れば成功です。

C#のステップ動作を使うと、どの行が表示のどの部分を制御しているかよくわかると思います。
色々と設定を変えてお試しください。

## 6. トラブルシュート

### 6.1. まったく点灯しない。

電源を投入してデータを書き込んでもまったく表示が出ない場合に原因を切り分けて調査する必要があります。　このような場合 CU-Y シリーズではテストモードを使って、表示器の単体動作確認を行うことができます。

方法：　電源をオフしてから JT ジャンパーを半田付けなどでショートします。この状態で電源を投入すると、コントローラからの信号なしでテスト表示を行います。

**テスト表示を行わない場合**は、電源が来ていないか、表示器が損傷を受けて動作していないと考えられます。

**テスト表示が正常な場合**は、表示器にデータがうまく入力されていない可能性があります。コネクタの端子をオシロスコープで観察するなどの方法で信号を確認してください。

## 7. 環境対応

グリーン調達ガイドラインを制定し含有する環境負荷物質の把握に努めております。
ISO14000 の認証も受けており環境問題への対応に努めております。

### 7.1. RoHS 指令対応

CU-Y シリーズ標準品は **RoHS 指令対応**となっております。
RoHS 適用除外となる次の鉛成分を含有する場合があります。

- 蛍光表示管に使用するガラス中の酸化鉛。
- セラミックコンデンサ以外の電子部品中のセラミックとガラスに含有する鉛成分。
- 半導体製品内部の接合に使用される高融点半田中の鉛成分。

## 8. 安全規格

プリント基板材に難燃性 UL:94-V0 グレード認定品を使用しております。
プリント基板上に UL 認定番号を表示しておりますのでご確認頂くことができます。

## 9. 免責と制限事項について

本文書に記載の情報は、注意をはらって検証しておりますがすべての環境で完全に動作することは確認できておりません。不具合等ございましたら検討致しますのでお問い合わせください。サンプルコードは弊社製品を使用するためのみ限り全てまたは一部をコピー頂いて構いませんが、アプリケーションソフトなどの最終成果物の動作検証はお客様の責任で実施頂きますようお願いします。

## 10. 問い合わせ先：

ご質問、ご要望等につきましては　弊社営業またはカスタマーサポートデスク cs@noritake-itron.jp　までお寄せください。