


# 真空荧光显示屏（VFD）模组

## 「GU-7000 系列模组」

### 应用指南

APF200 R2.0  
(2012. 6. 14 改订)

 为则武真空荧光显示屏（VFD）的注册商标。VFD源自英文Vaccum Fluorescent Display，是一种自发光的显示器件，与液晶（LCD）、LED等其它显示器件比较，具有发光柔和、没有视角限制、工作温度范围宽等特点。GU-7000系列是全点阵显示的图案型显示模组，同时在外观尺寸上和常用的LCD字符型模组保持了兼容，是我公司字符型模组CU-U系列的功能加强版。本手册是方便您使用该系列显示模组的辅助技术资料，希望能对您的正确使用有所帮助。

February 3, 2011 Copy rights reserved.

## 目次

真空荧光显示屏（VFD）模组	- 1 -
「GU-7000 系列模组」	- 1 -
1 概要	- 5 -
1.1 关于显示字符大小	- 5 -
1.2 关于 GU-7000B	- 5 -
2 VFD 模组标准品一览表	- 6 -
2.1 品名	- 6 -
2.2 VFD 模组产品系列	- 7 -
3 GU-7000 系列产品（2012 年 6 月现在）	- 8 -
4 硬件信息	- 9 -
4.1 方框图	- 9 -
4.1.1 方框图（GU-7000）	- 9 -
4.1.2 方框图（GU-7901）	- 9 -
4.1.3 方框图（GU7032）	- 10 -
4.1.4 方框图（GU-7003）	- 10 -
4.1.5 方框图（GU-7900）	- 11 -
4.1.6 方框图（GU-7903）	- 11 -
4.1.7 方框图（GU-7040）	- 12 -
4.2 插座	- 12 -
4.2.1 GU-7xx0	- 12 -
4.2.2 GU-7901	- 12 -
4.2.3 GU-7003	- 12 -
4.3 接口	- 13 -
4.4 输入输出端口等价电路	- 14 -
4.4.1 RS-232C 输入端等价电路	- 14 -
4.4.2 RS-232C 输出端等价电路	- 14 -
4.4.3 C-MOS 输入端等价电路	- 14 -
4.4.4 C-MOS 输出端等价电路	- 14 -
4.4.5 并口输入输出端等价电路（GU140X16G-7000）	- 15 -
4.5 接口连接示例	- 16 -
4.5.1 RS-232C 串口和 PC 连接示例	- 16 -
4.5.2 RS-232C 在嵌入式应用中和 CPU 的连接示例 1	- 16 -
4.5.3 RS-232C 在嵌入式应用中和 CPU 的连接示例 2	- 17 -
4.5.4 并口输入 连接示例 1	- 17 -
4.5.5 并口输入 连接示例 2 使用 BUSY 信号	- 18 -
4.5.6 并口输入 连接示例 3 使用 RESET 信号	- 18 -
4.5.7 C-MOS 串口（异步）嵌入式应用和 CPU 连接示例	- 19 -

4.5.8	C-MOS 串口（同步）嵌入式应用和 CPU 连接示例 .....	19 -
4.5.9	使用 BUSY 信号输出和硬件 Reset 信号输入功能.....	19 -
4.5.10	C-MOS 串口（异步）和 PC 的连接示例 .....	20 -
5	软件 .....	21 -
5.1	初始设定及通信协议.....	21 -
5.2	显示内存（RAM） .....	21 -
5.3	自动字符间距.....	21 -
5.4	搭载字库 （末尾带 B 的型号和不带 B 的型号相同） .....	22 -
5.5	字库基础 .....	23 -
5.5.1	5x7dot ANK （单字节字符） .....	23 -
5.5.2	指定国际字符集（特殊字符集） .....	24 -
5.5.3	16x16dot JIS, 简体字, 繁体字, 韩国语 （仅限 GU-79xx） .....	25 -
5.6	指令集 .....	27 -
5.6.1	指令集 1 GU-7xxx 共通.....	27 -
5.6.2	指令集 2 扩充指令集 GU-7xxxx 共通.....	27 -
5.6.3	指令集 3 扩充指令集 FROM 操作 GU-79xxx.....	30 -
5.6.4	指令集 4 扩充指令集 双字节文字操作 GU-79xx .....	30 -
5.6.5	指令集 5 GU-7xxxB 追加指令 点单位任意位置显示.....	31 -
5.7	光标移动及显示模式.....	32 -
5.8	Windows PC 环境下使用 Microsoft 产品 Visual Studio 2010 的程序制作举例 .....	33 -
5.8.1	使用 Visual C# 2010 集成开发环境，通过串口连接 GU-7000 模组并显示字符。 ..	33 -
5.9	示例程序 .....	36 -
5.9.1	清除画面.....	36 -
5.9.2	设定光标位置.....	36 -
5.9.3	放大字符.....	36 -
5.9.4	设定字符间距（英数字） .....	37 -
5.9.5	设定亚洲字符显示模式(仅适用 7900 系列).....	37 -
5.9.6	使用中文汉字字符（简体）显示功能.....	38 -
5.9.7	显示图片.....	38 -
5.9.8	图像显示实例.....	39 -
5.9.9	滚动显示图像.....	40 -
5.9.10	滚屏显示示例程序.....	41 -
5.9.11	显示非可见区域画面.....	41 -
5.9.12	字符的滚动显示.....	41 -
5.9.13	字符滚动显示的示例程序.....	42 -
5.9.14	使用用户自定义窗口功能.....	42 -
5.9.15	使用用户自定义窗口示例程序.....	44 -
5.10	使用并口通信的示例程序.....	45 -
6	故障排查 .....	45 -

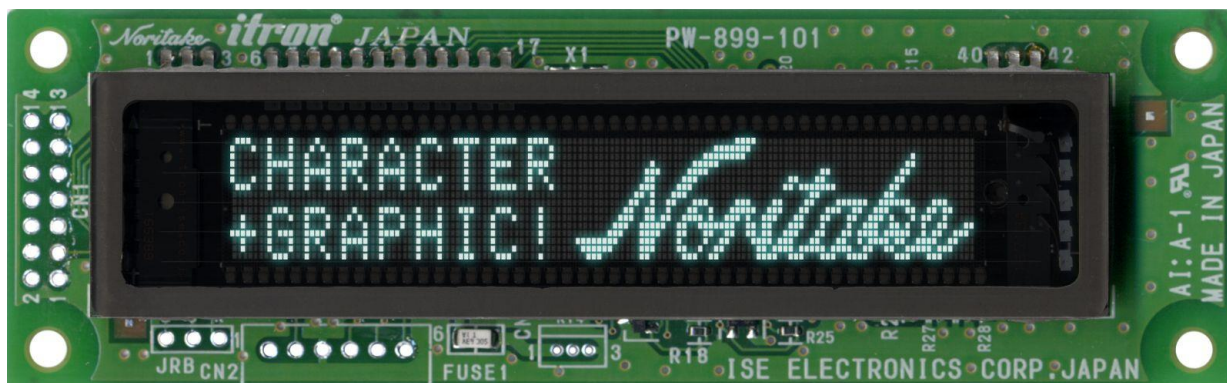
6.1	关于 BUSY 信号.....	45
6.2	关于 RESET.....	45
6.3	显示器无法点亮---自检模式.....	45
6.4	自检模式的设定方法.....	47
6.4.1	GU112X16G-7000, -7003, -7900, -7000B, -7003B, -7900B.....	47
6.4.2	GU128X32D-7000, -7003, -7900, -7000B, -7003B, -7900B.....	47
6.4.3	GU128X32D-7050, -7950.....	47
6.4.4	GU128X32D-7901.....	47
6.4.5	GU128X64D-7000, -7003, -7900.....	48
6.4.6	GU128X64F-7000, -7003, -7900, -7900BX.....	48
6.4.7	GU128X128D-7203.....	48
6.4.8	GU140X16G-7040A.....	48
6.4.9	GU140X16G-7000, -7003, -7900, -7903, -7042, -7000B, -7003B.....	48
6.4.10	GU140X16J-7000, -7003, -7000B, -7003B, -7900B.....	49
6.4.11	GU140X32F-7000, -7003, -7900, -7903, -7032, -7000B, -7003B, -7900B.....	49
6.4.12	GU140X32F-7050, GU140X32F-7053, GU140X32F-7950.....	49
6.4.13	GU144X16D-7053B.....	49
6.4.14	GU160X32D-7050, -7950.....	50
6.4.15	GU160X80E-7900B.....	50
6.4.16	GU256X64C-7000, -7003, -7900.....	50
6.4.17	GU256X64D-7000, -7900.....	51
6.4.18	GU280X16G-7000, -7003.....	51
7	技术支持 .....	52
8	Environment .....	53
8.1	RoHS Compliance.....	53
9	Safety standard.....	53
10	Disclaimers and limitations.....	53
11	Contact us .....	53

## 1 概要

GU-7000 系列 VFD 模组是把高可视性、高可靠性的 VFD 显示屏和电源回路、控制用 CPU 以及 Flash 内存组合在一起的独立显示单元。由于搭载有 CPU，通过从主机控制器传送相应的指令就可以轻松控制显示，并提供有搭载 JIS 日文字库或中文、韩文字库的产品。

外形尺寸上与常见的字符型 LCD 模组保持了一致，这个设计概念与我公司 CU-U 系列 LCD 兼容模组相同。Photo. 1 是该系列模组的示例。

PHOTO. 1 GU140X16G-7000



GU-7000 系列内置有文字显示功能指令集，可以方便地实现文字的显示。

### 1.1 关于显示字符大小

VFD 是使用荧光粉作为显示介质的自发光型显示器产品，和利用反射光 / 透射光的 LCD 相比，大大提高了其可视性。同样文字大小可以从更远距离阅读，可视距离一定的则文字可以做得更小。具体选型时请以实际产品确认显示效果。

### 1.2 关于 GU-7000B

我公司正在顺次推出 GU-7000 系列的强化版 GU-7000B 系列。新系列在设计上向下兼容原-7000 系列，原则上是可以直接替代，但是由于无法完全避免时序等的细微差异，敬请在替代前进行充分评测为盼。

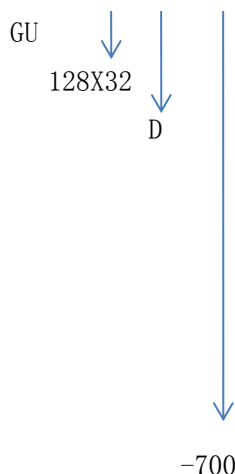
	GU-7000/ GU-7900	GU-7000B, GU-7900B
串口输入缓冲大小	12 字节	60 字节
任意指定显示位置功能（点单位）	无	有 参考「 <a href="#">指令集 5 GU-7xxxB 追加指令 任意位置显示</a> 」。

## 2 VFD 模组标准品一览表

### 2.1 品名

通过品名了解产品概要。

品名例：GU128X32D-7003



- : 图案型显示模组
- : 128x32 像素
- : 像素尺寸 0.4~0.49mm/像素

C=0.3~0.39mm

D=0.4~0.49mm

E=0.5~0.59mm

F=0.6~0.69mm

G=0.7~0.79mm

J=1.0~1.09mm

- : 7000 系列

接口等规格:

7x3x=电源电压 3.3V

70xx=单字节字符专用

72xx=面向欧美的单字节字符显示功能强化品。

79xx=同时支持单字节和双字节字符

(支持 JIS 日文字库、中文以及韩文字库)

7xx0=RS232C + 并口

7xx1= USB 接口

7xx2=C-MOS 串口 + 并口

7xx3=C-MOS 串口

品名例：GU128X32D-7000B

:

末尾的 B 表示为功能增强的强化版系列。

同一系列产品其指令和主要规格是统一的，上面表格为共通部分的内容。具体型号的详细规格请参考各个型号的产品规格书。

## 2.2 VFD 模组产品系列

GU-7000 系列以外产品，请参考各自系列的应用指南以及各个产品的详细规格书。

则武真空荧光显示器（VFD）模组

- └ 客户定制模组
- └ 标准品 CU 系列
- └ 标准品 GU 系列
  - | 可显示图像的图案型模组系列
  - └ GU-300 系列、GU9300 系列
  - |
  - └ GU-800 系列
    - └ GU-800, GU-800B
    - | 图案显示专用
    - └ GU-8000, GU-8000B
    - | 追加了汉字显示功能
  - |
  - └ GU-7000 系列 外形尺寸兼容常用字符型 LCD 模组。
    - └ GU-7000
      - | 小型、图案型模组。
    - └ GU-7032
      - | 7000 的 3.3V 电源产品。
    - └ GU-7900
      - | 小型、7000 + 汉字显示。
    - └ GU-7000B
      - | 小型、图案型模组，7000 的后继产品。
    - └ GU-7900B
      - | 小型、7000B + 汉字显示，7900 的后继产品。
    - └ GU-7900BX
      - | 7900 的高亮度产品。
  - |
  - └ GU-3000 系列
    - └ GU-3100
      - | 图案型模组产品。
      - | 搭载 16x16&32x32Dot 日语字库。
      - | 搭载 FROM 及宏功能，可不依赖外部控制而独立执行客户程序。
    - └ GU-3900
      - | 图案型模组产品。
      - | 搭载日文、中文简・繁体、韩文字库。
      - | 搭载 FROM 及宏功能，可不依赖外部控制而独立执行客户程序。
    - └ GU-3900B
      - | 图案型模组产品。
      - | 向下兼容 3100&3900 的该系列后继产品。

### 3 GU-7000 系列产品 (2012 年 6 月现在)

本应用指南的对象品种如下:

GU112X16G-7000	GU112X16G-7000B	GU112X16G-7003	GU112X16G-7003B
GU112X16G-7900	GU112X16G-7900B		
GU128X32D-7000	GU128X32D-7000B	GU128X32D-7003	GU128X32D-7003B
GU128X32D-7050	GU128X32D-7900	GU128X32D-7900B	
GU128X32D-7901			
GU128X32D-7950			
GU128X64D-7000	GU128X64D-7003	GU128X64D-7900	
GU128X64F-7000	GU128X64F-7003	GU128X64F-7900	GU128X64F-7900BX
GU128X128D-7203			
GU140X16G-7000	GU140X16G-7000B	GU140X16G-7003	GU140X16G-7003B
GU140X16G-7040A	GU140X16G-7900	GU140X16G-7900B	GU140X16G-7903
GU140X16J-7000	GU140X16J-7000B		
GU140X16J-7003	GU140X16J-7003B	GU140X16J-7900B	
GU140X32F-7000	GU140X32F-7000B	GU140X32F-7003	GU140X32F-7003B
GU140X32F-7032			
GU140X32F-7050	GU140X32F-7053	GU140X32F-7900	GU140X32F-7900B
GU140X32F-7903	GU140X32F-7950		
GU144X16D-7053B			
GU160X32D-7000	GU160X32D-7900		
GU160X80E-7900B			
GU256X64C-7000	GU256X64C-7003	GU256X64C-7900	
GU256X64D-7000	GU256X64D-7900		
GU280X16G-7000	GU280X16G-7003		

最新标准产品信息请参考我[公司网站](http://www.noritake-itron.jp/)或者就近咨询我公司营业部门负责人员。

<http://www.noritake-itron.jp/>

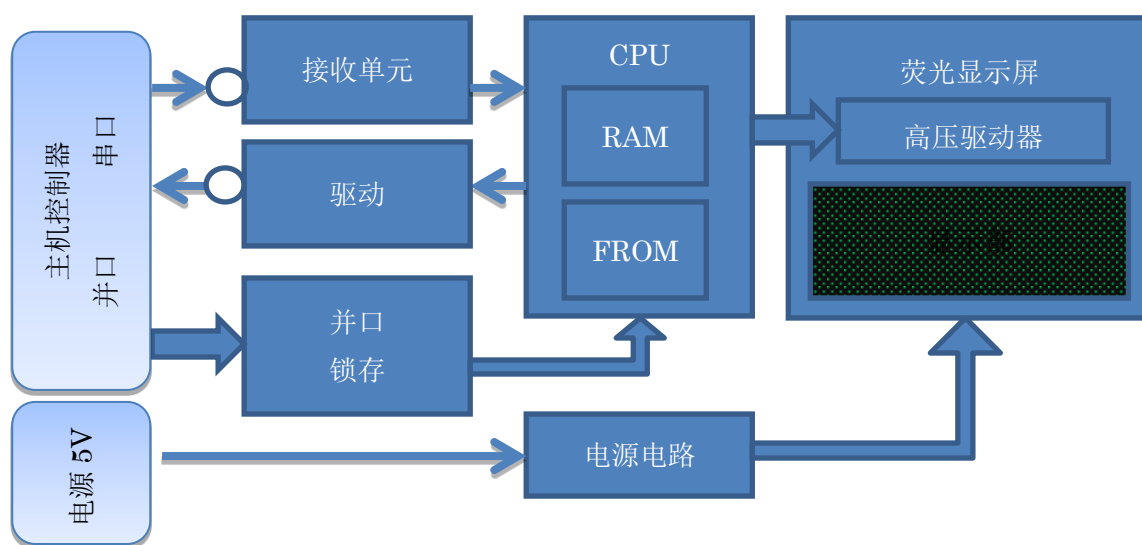


## 4 硬件信息

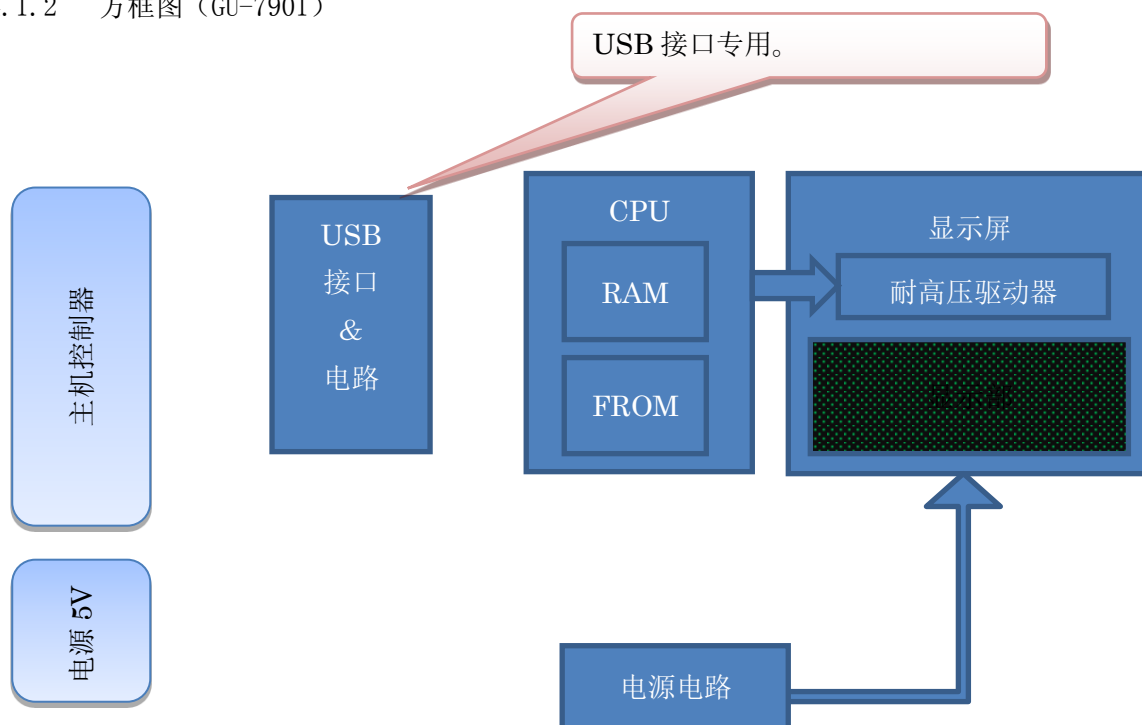
### 4.1 方框图

VFD 模组主要由输入输出部分、CPU（控制电路）、电源电路、VFD 屏组成。

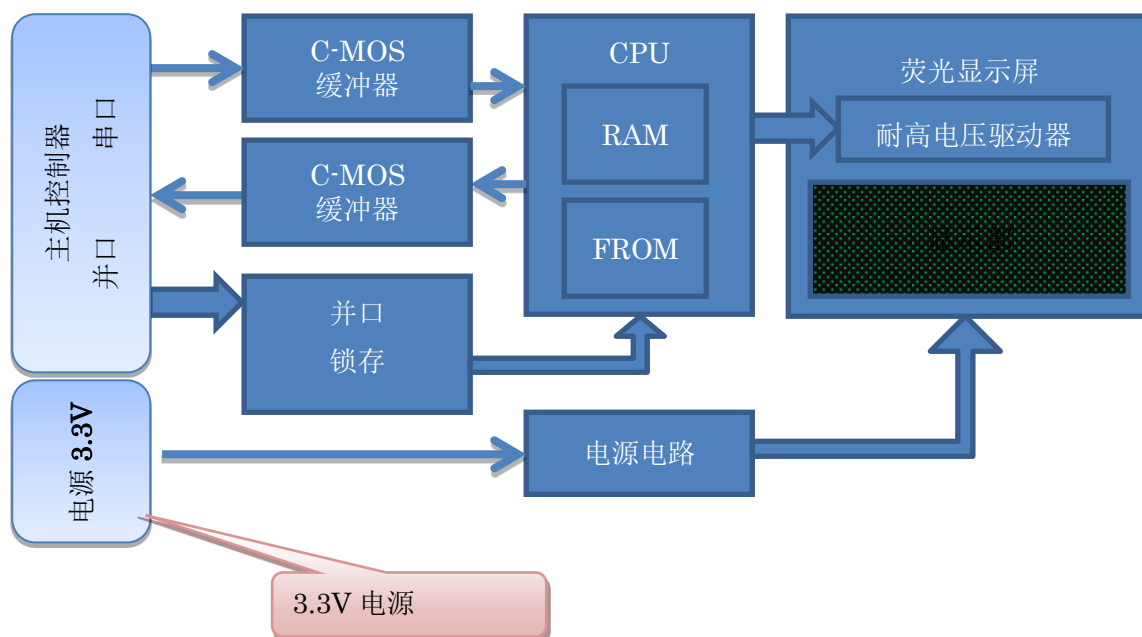
#### 4.1.1 方框图（GU-7000）



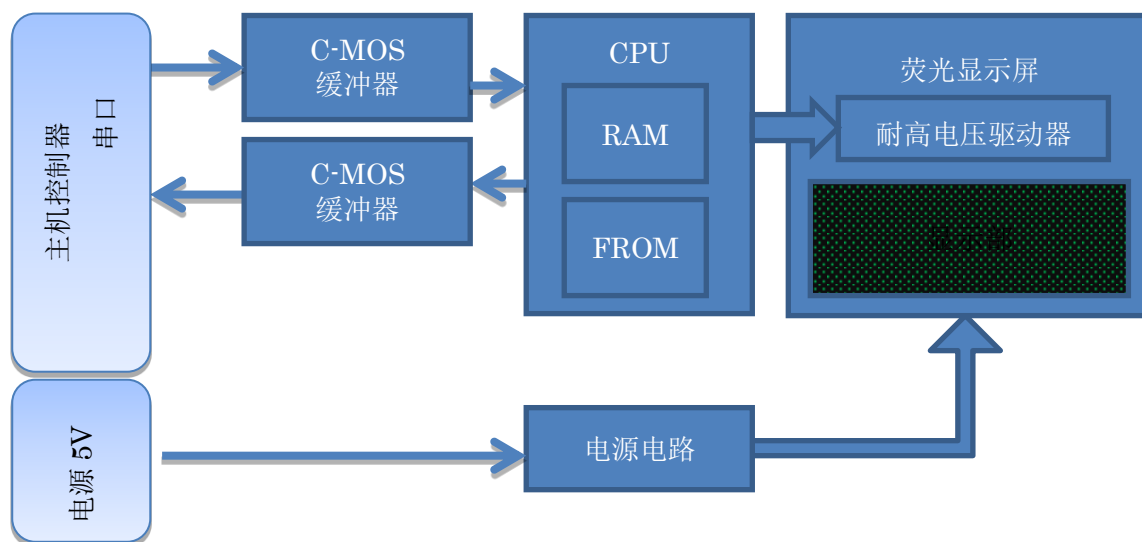
#### 4.1.2 方框图（GU-7901）



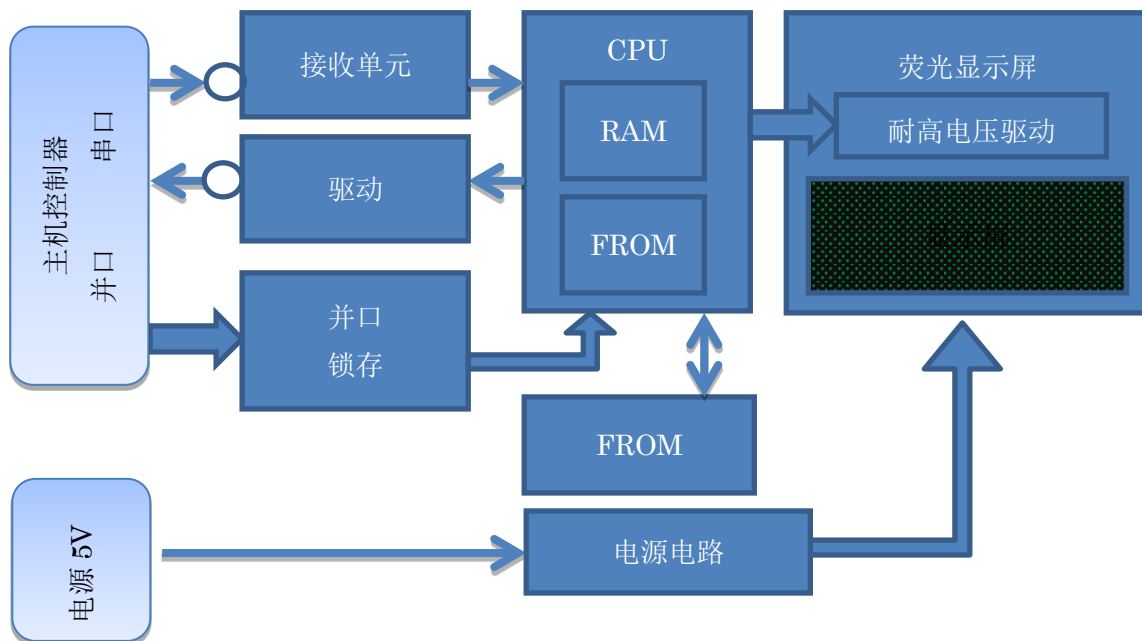
#### 4.1.3 方框图 (GU7032)



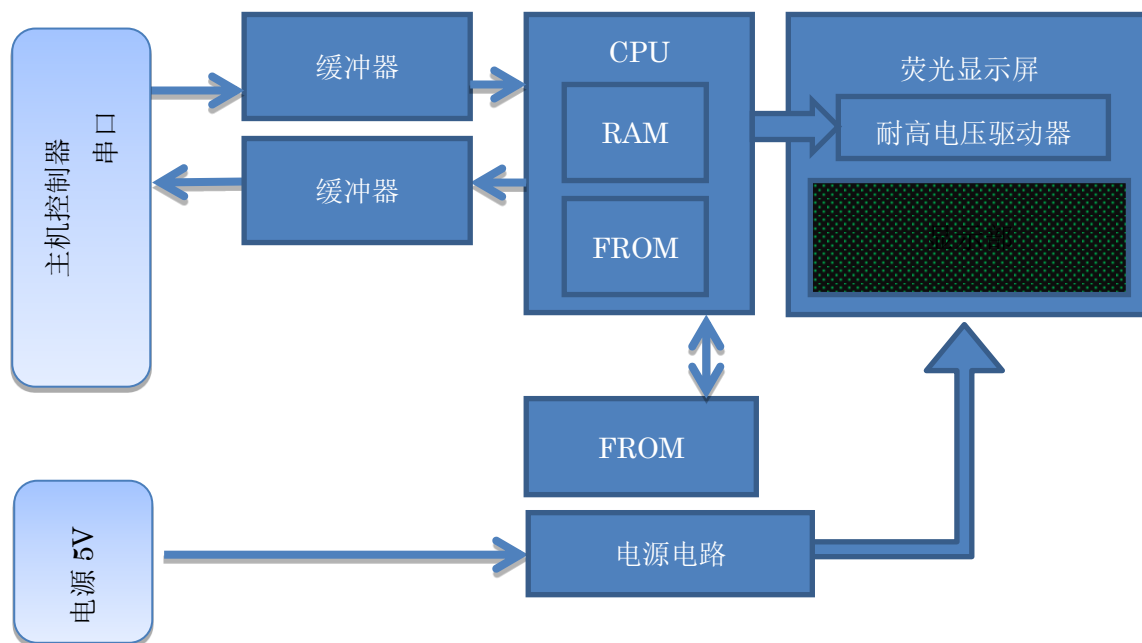
#### 4.1.4 方框图 (GU-7003)



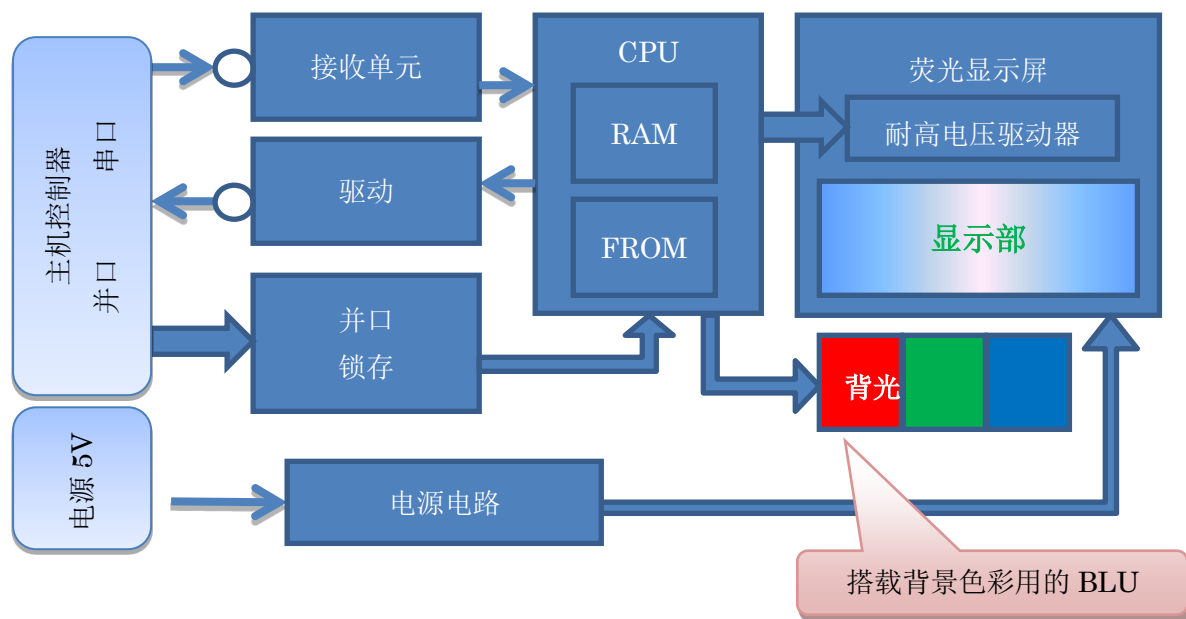
#### 4.1.5 方框图 (GU-7900)



#### 4.1.6 方框图 (GU-7903)



## 4.1.7 方框图 (GU-7040)



## 4.2 插座

## 4.2.1 GU-7xx0

GU-7000 系列, 除 USB 接口 (品名: -7xx1) 以外, 其他接口均未安装接口插座。请焊接插座后使用。另外、GU-7000 系列不适合回流焊接方法。

## 4.2.2 GU-7901

GU128X32D-7901 搭载 USB Mini-B 插座, 可以使用市售的 USB 线直接与 PC 连接, PC 的 USB 电源可以直接驱动。

## 4.2.3 GU-7003

GU-7000 系列, 除 USB 接口 (品名: -7xx1) 以外, 其他接口均未安装接口插座。请焊接插座后使用。另外、GU-7000 系列不适合回流焊接方法。

### 4.3 接口

型号名称末尾的 1 位数表明的是接口的规格。

- 0: RS-232C 电平异步串口输入 + 8 位并口输入
- 1: USB
- 2: C-MOS 电平同步&异步串口输入+ 8 位并口输入
- 3: C-MOS 电平同步&异步串口输入

尾号 0 和 3 是标准品，请根据客户端接口情况选用。

关于 USB 接口产品：

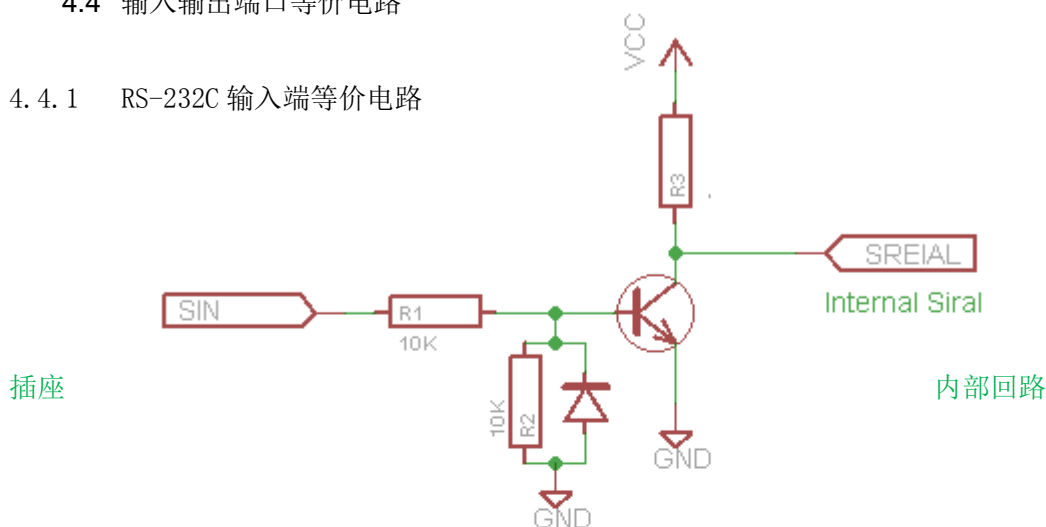
对象产品： GU-7xx1

GU128X32D-7901 是 USB 接口产品，组装在如下图的盒子里并以 COMEMO 的品名在销售。（目前只用在日本国内销售）

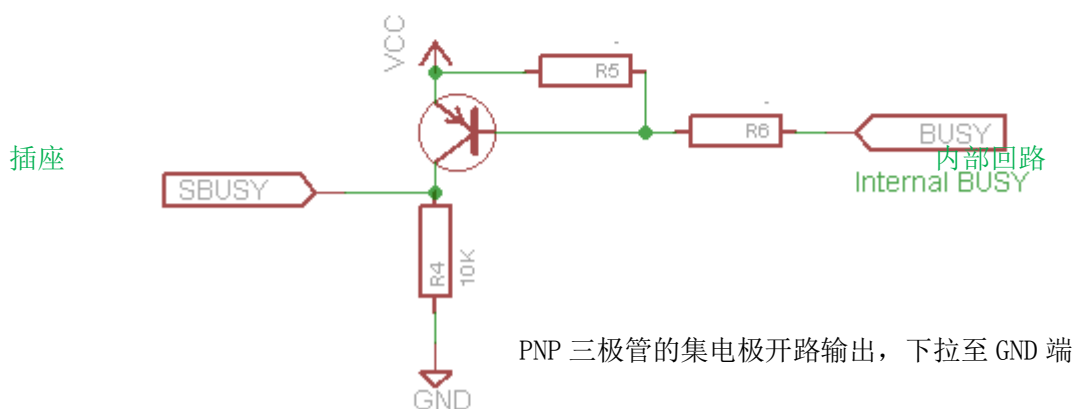


#### 4.4 输入输出端口等价电路

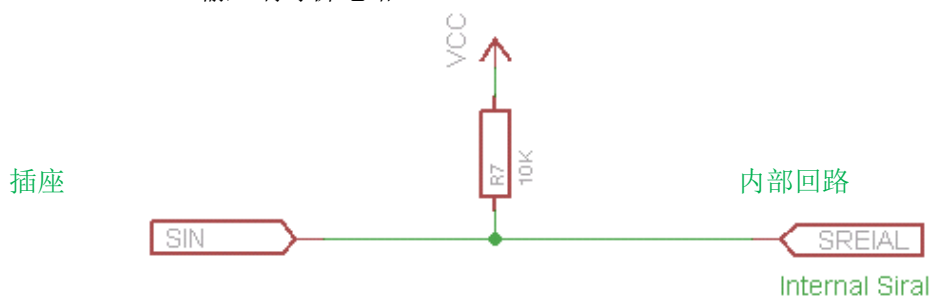
##### 4.4.1 RS-232C 输入端等价电路



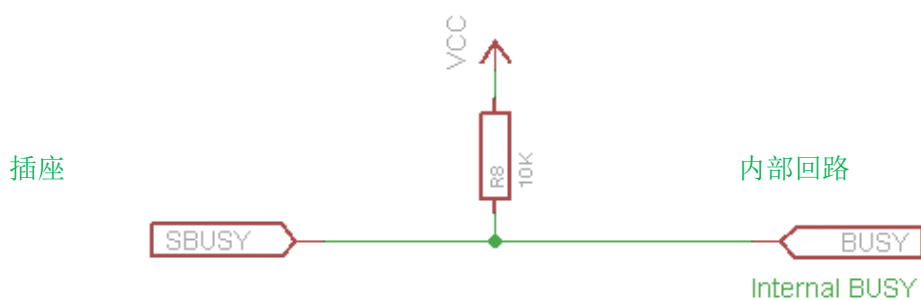
##### 4.4.2 RS-232C 输出端等价电路



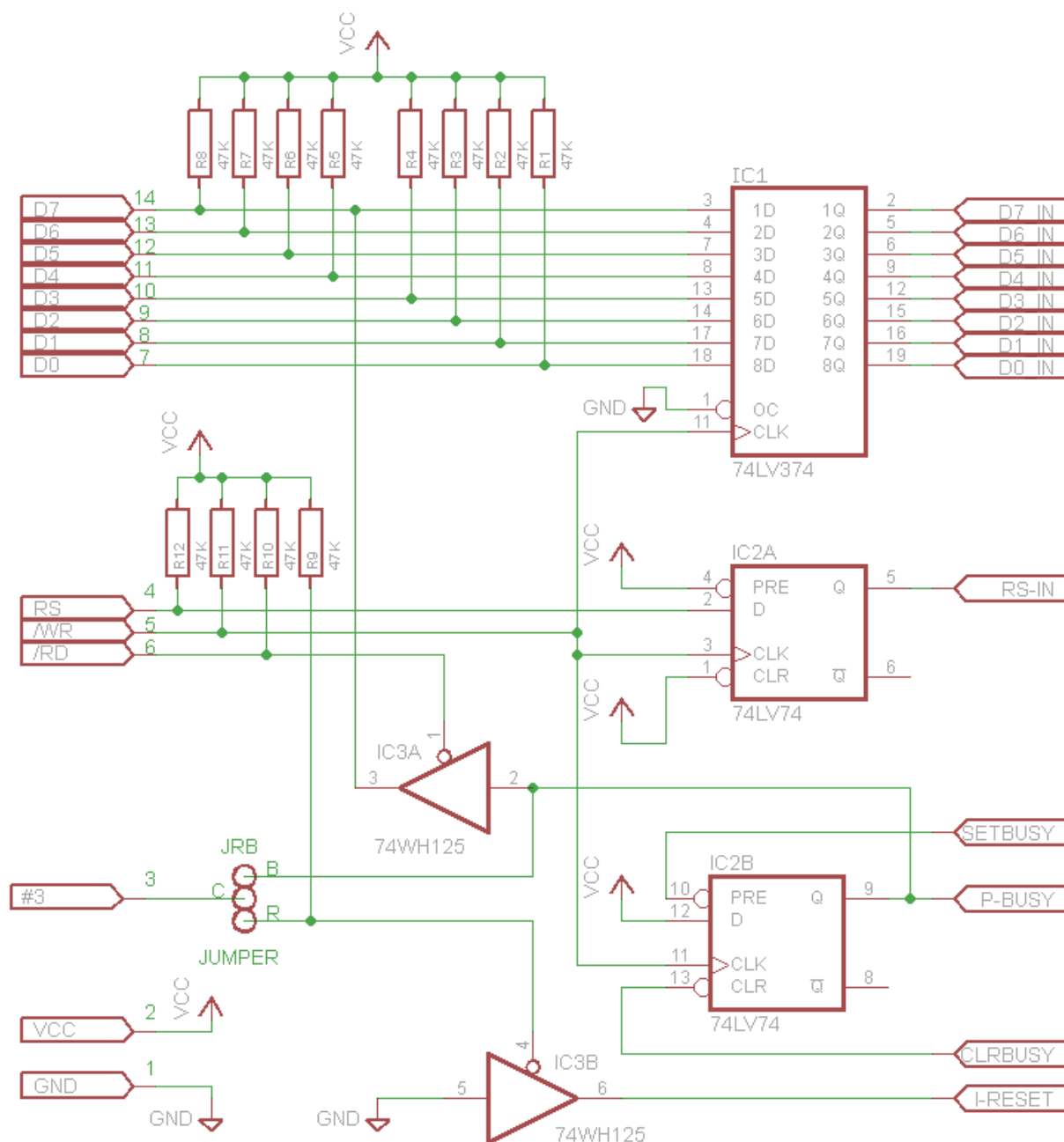
##### 4.4.3 C-MOS 输入端等价电路



##### 4.4.4 C-MOS 输出端等价电路



## 4.4.5 并口输入输出端等价电路 (GU140X16G-7000)



未使用端子的处理:

如等价回路所示, 由于显示器内部的复位端子在内部接续有上拉电阻, 所以未使用的端子无需做任何特别处理, 置空闲即可。但是, 请避免对空闲端子单独走配线, 以避免引入噪音造成误动作。

## 4.5 接口连接示例

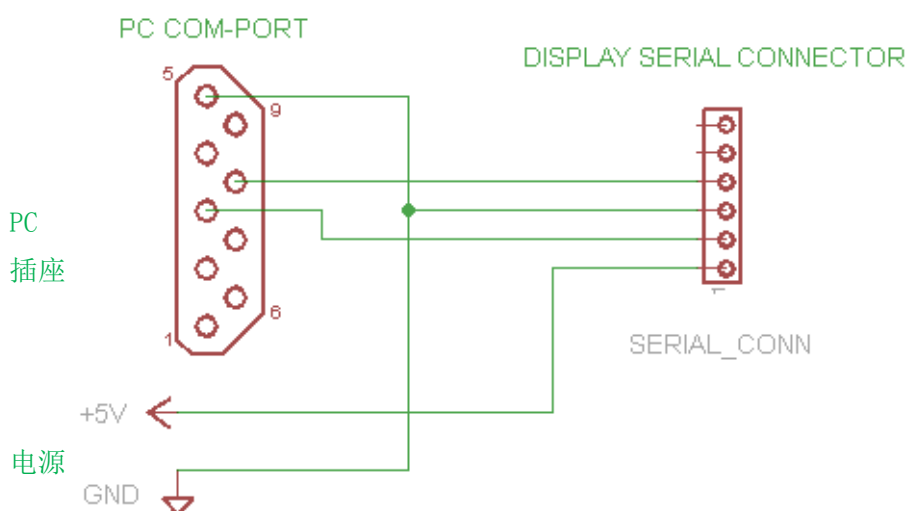
有些指令例如滚屏、等待等操作往往需要一定的时间，为了提高执行效率请在写入每个字节前查 BUSY 信号的状态（以 BUSY 作为硬件握手信号）。

串口输入内部有 FIFO 缓冲区，但是该缓冲区是为了防止通信的溢出错误而设置，如果需要缩短数据写入时间、提高通信效率，最好的方法仍然是在写入每个字节前查 BUSY 信号的状态（按照规格书记载的方法）

注）连接示例仅供参考，请在实际操作时注意确认生产厂家具体规格。

### 4.5.1 RS-232C 串口和 PC 连接示例

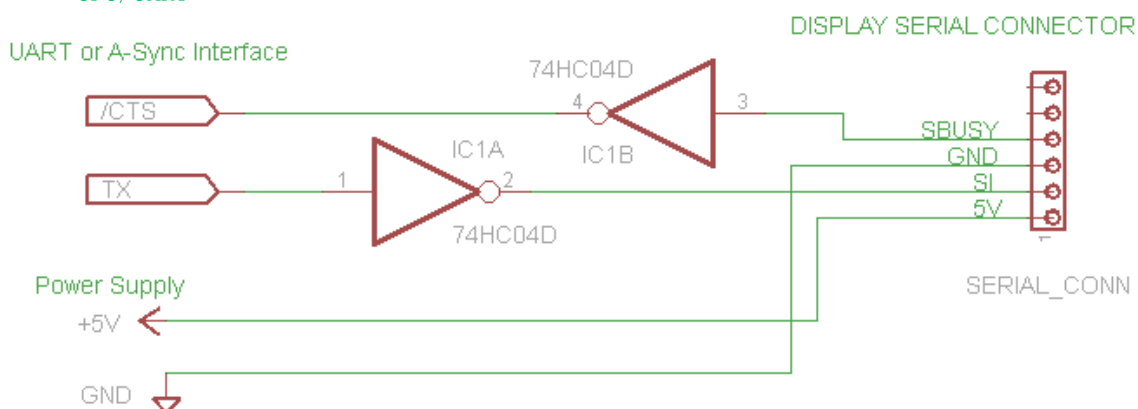
对象产品： GU-7xx0



### 4.5.2 RS-232C 在嵌入式应用中和 CPU 的连接示例 1

对象产品： GU-7xx0

CPU/UART

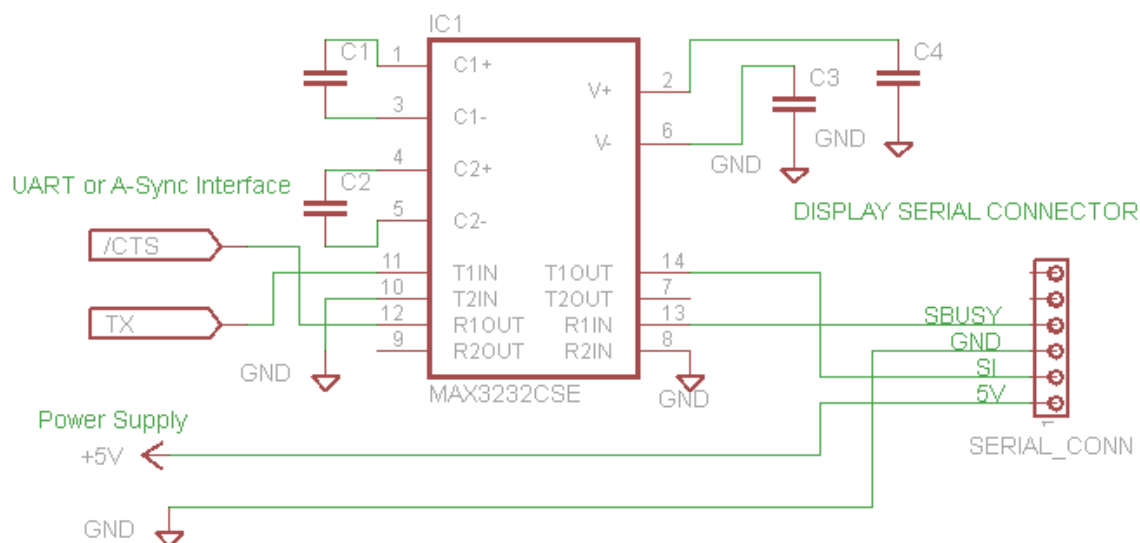


关于 /CTS 信号：如果所使用的 CPU 没有硬件握手信号功能，请连接任意通用输入端，通过软件控制确认 /CTS 信号的状态来完成数据通信。



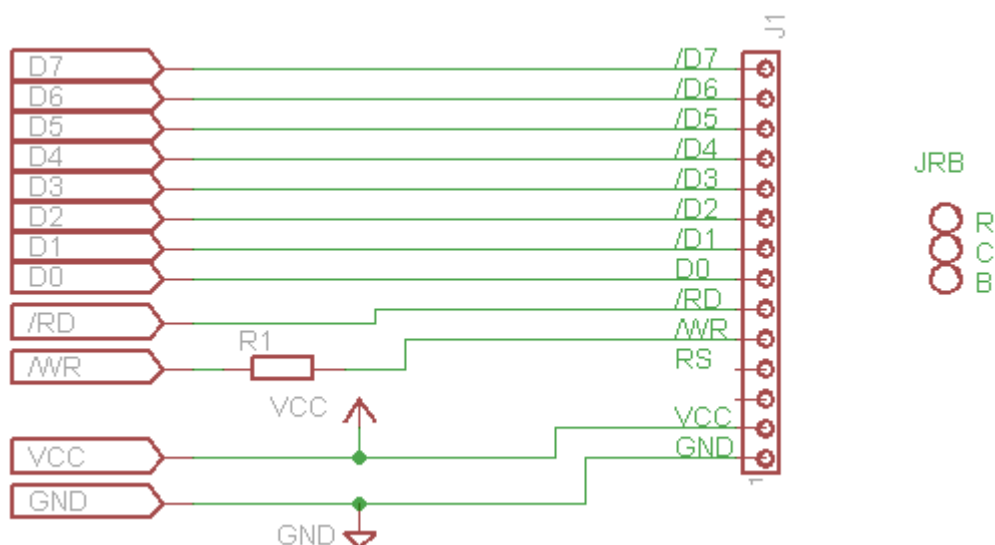
#### 4.5.3 RS-232C 在嵌入式应用中和 CPU 的连接示例 2

对象产品: GU-7xx0



#### 4.5.4 并口输入 连接示例 1

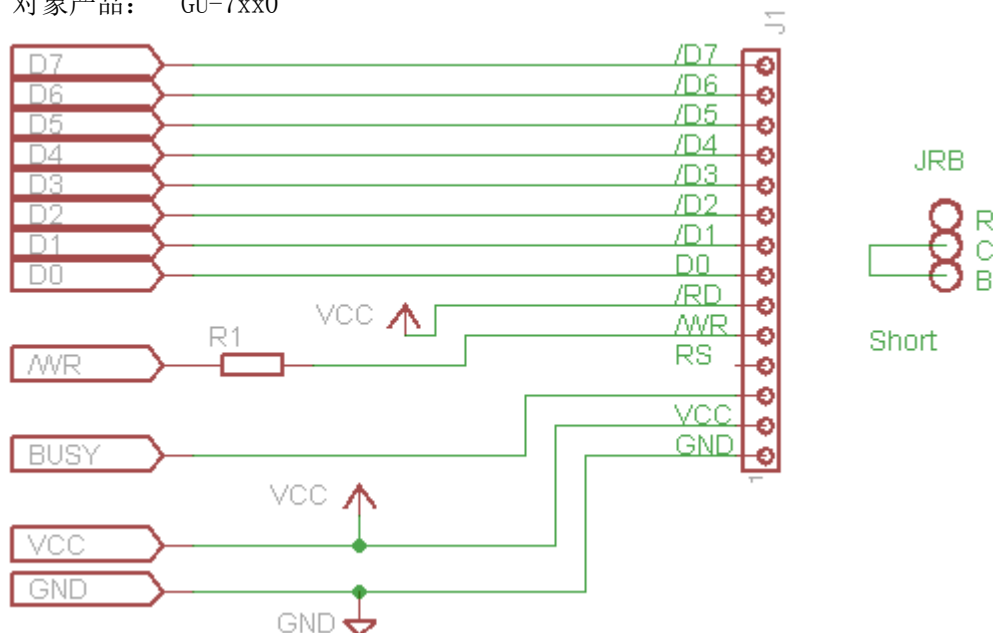
对象产品: GU-7xx0



注) 关于 R1 问题: /WR 信号波形发生过冲 (overshoot) 或者下冲 (undershoot) 则有可能导致误动作。此时追加 R1 (50~200Ω 程度) 可以获得改善。R1 请追加在靠近 /WR 信号输出端子的位置。

## 4.5.5 并口输入 连接示例 2 使用 BUSY 信号

对象产品: GU-7xx0

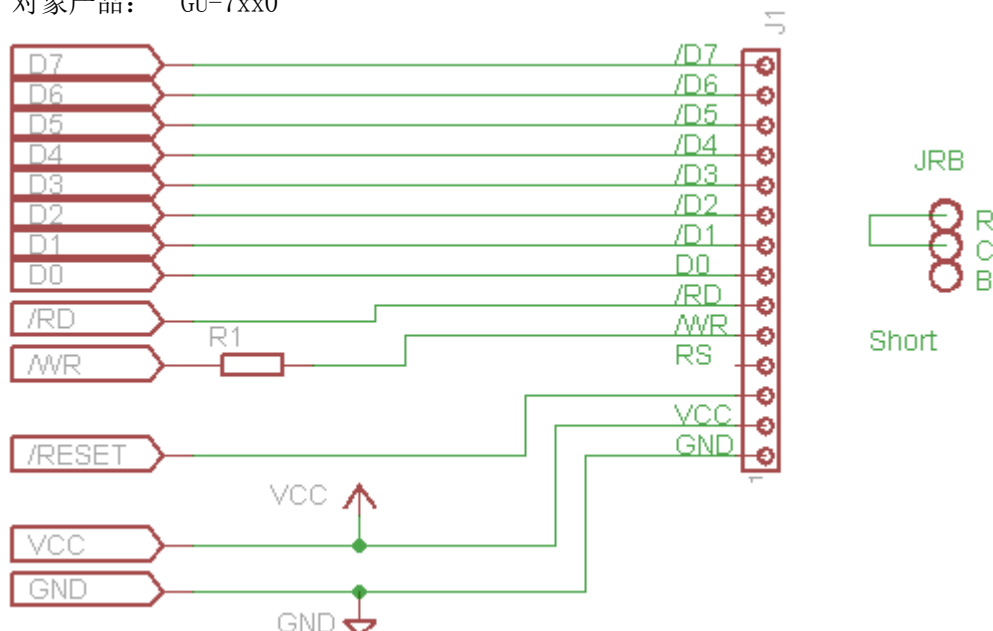


注) 关于 R1 问题: /WR 信号波形发生过冲 (overshoot) 或者下冲 (undershoot) 则有可能导致误动作。此时追加 R1 (50~200 Ω 程度) 可以获得改善。R1 请追加在靠近 /WR 信号输出端子的位置。

具体程序示例请参考 5.10 使用并口通信的示例程序

## 4.5.6 并口输入 连接示例 3 使用 RESET 信号

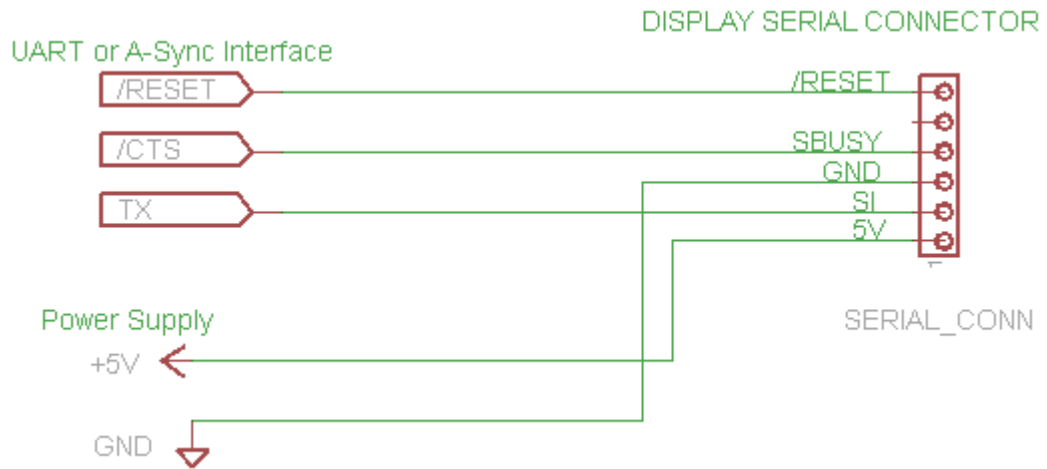
对象产品: GU-7xx0



注) 关于 R1 问题: /WR 信号波形发生过冲 (overshoot) 或者下冲 (undershoot) 则有可能导致误动作。此时追加 R1 (50~200 Ω 程度) 可以获得改善。R1 请追加在靠近 /WR 信号输出端子的位置。

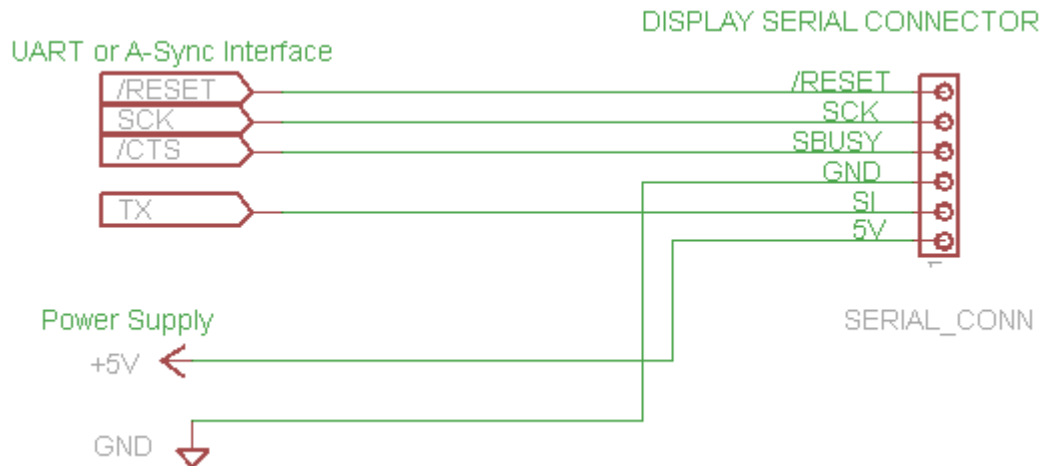
#### 4.5.7 C-MOS 串口（异步）嵌入式应用和 CPU 连接示例

对象产品： GU-7xx3



#### 4.5.8 C-MOS 串口（同步）嵌入式应用和 CPU 连接示例

对象产品： GU-7xx3



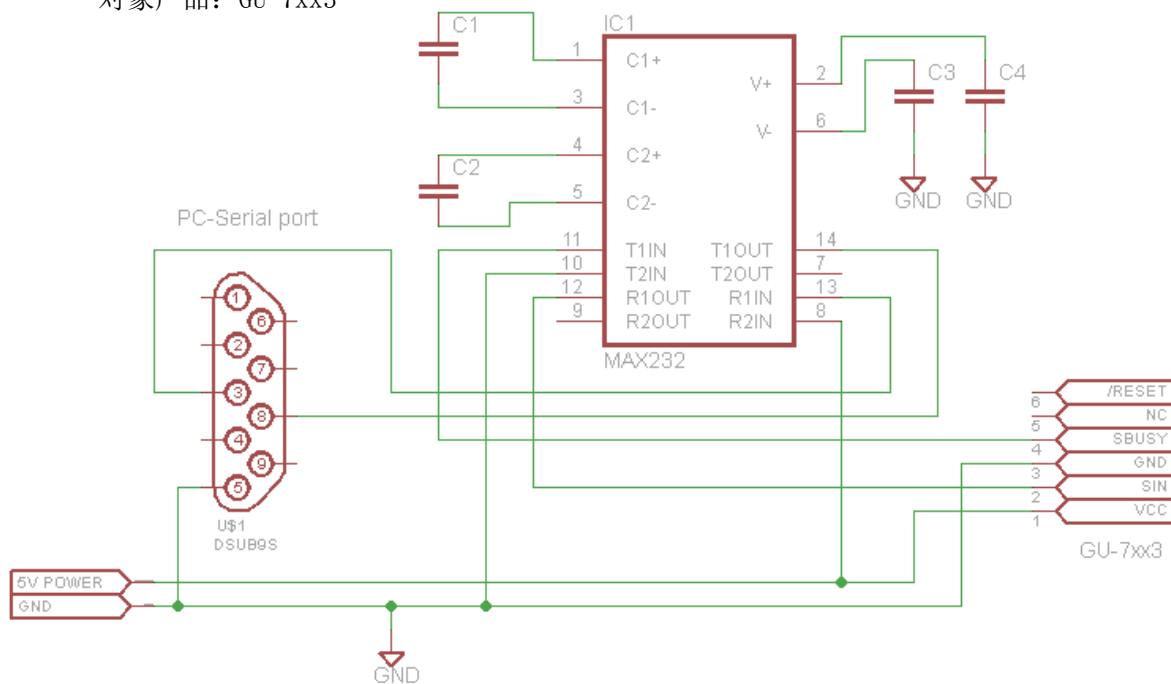
#### 4.5.9 使用 BUSY 信号输出和硬件 Reset 信号输入功能

对象产品： GU-7xxx

并口接口的第 3 引脚可以设定作为 BUSY 信号的输出端或者作为 /RESET 信号的输入端使用。该设定通过 PCB 板上跳线器（JBR）来进行。

#### 4.5.10 C-MOS 串口（异步）和 PC 的连接示例

对象产品：GU-7xx3



## 5 软件

### 5.1 初始设定及通信协议

通信协议为简单的硬件握手协议。

系统默认的内部初始化动作自动完成通信准备，用户无需特别的初始化操作即可使用基本的显示功能，只需要在加电后等待适当的时间（初始化所需时间）即可。显示数据采用以 ASCII 为基础的通用代码体系，控制指令则采用以 ESC 及其他代码引导的控制序列，对模组写入时请使用硬件握手信号进行通信控制。

GUD10K 是我们提供的免费的客户支持软件，可以帮助客户准确快速地理解 GU-7000 系列的各种功能，客户可以在我公司网站下载该软件。

### 5.2 显示内存（RAM）

从端口写入的数据在模组内部即时被处理为图形数据并存储在显示内存 RAM 中，由于 RAM 的容量要大于显示画面的点阵数，所以实际显示出来的只是 RAM 中的部分数据。也就是说，RAM 分为可见区域和非可见区域两个部分。

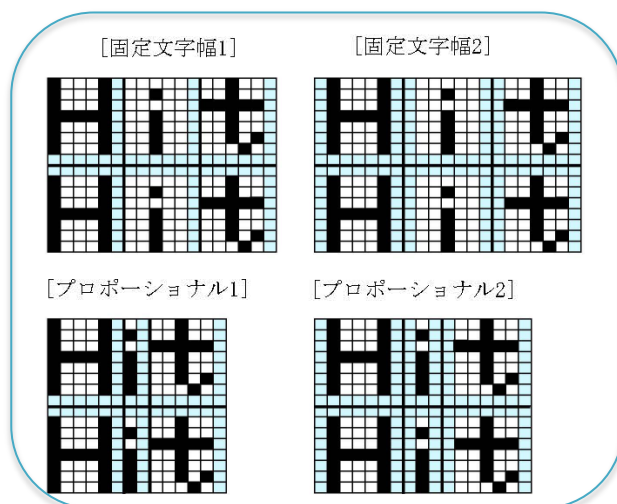


非可见区域可以用来存储不希望被用户看到的中间数据，或者用做滚动显示图像的存储区域。

### 5.3 自动字符间距

自动字符间距功能是通过自动压缩字符间距，从而在同样的显示区域容纳更多显示信息的功能。此时光标的移动距离也会随之自动调整。

该功能仅在单字节的英文及数字字符时有效。



#### 5.4 搭载字库 （末尾带 B 的型号和不带 B 的型号相同）

型号	搭载字库
GU112X16G-7000	5x7dot ANK & International
GU112X16G-7003	5x7dot ANK & International
GU112X16G-7900	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU128X32D-7000	5x7dot ANK & International
GU128X32D-7003	5x7dot ANK & International
GU128X32D-7900	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU128X32D-7901	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU128X64D-7000	5x7dot ANK & International
GU128X64D-7003	5x7dot ANK & International
GU128X64D-7900	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU128X64F-7000	5x7dot ANK & International
GU128X64F-7003	5x7dot ANK & International
GU128X64F-7900	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU128X128D-7203	6x8, 8x16, 12x24, 16x32dot ANK & International
GU140X16G-7040A	5x7dot ANK & International
GU140X16G-7000	5x7dot ANK & International
GU140X16G-7003	5x7dot ANK & International
GU140X16G-7900	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU140X16G-7903	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU140X16J-7000	5x7dot ANK & International
GU140X16J-7000A	5x7dot ANK & International
GU140X16J-7003	5x7dot ANK & International
GU140X32F-7000	5x7dot ANK & International
GU140X32F-7003	5x7dot ANK & International
GU140X32F-7900	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU140X32F-7903	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU140X32F-7032	5x7dot ANK & International
GU140X32F-7050	5x7dot ANK & International
GU140X32F-7053	5x7dot ANK & International
GU140X32F-7950	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU256X64C-7000	5x7dot ANK & International
GU256X64C-7900	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU256X64D-7000	5x7dot ANK & International
GU256X64D-7900	5x7dot ANK & International, 16x16dot JIS, 简体字, 繁体字, 韩国语
GU280X16G-7000	5x7dot ANK & International
GU280X16G-7003	5x7dot ANK & International

## 5.5 字库基础

下面是模组字库的基本框架，具体请参考字库规格书。

### 5.5.1 5x7dot ANK （单字节字符）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
0H	指令	通用 ASCII 字符区							扩充字符区 使用 ESC t n 指令设定							
1H																
2H																
3H																
4H																
5H																
6H																
7H																
8H																
9H																
AH																
BH																
CH																
DH																
EH																
FH																

单字节字符共划分为 3 个区域：

- 00Hex~1FHex： 指令区域，该区域字符被赋予执行特定动作的功能。例如写入 0DHex (CR) 模组即会执行光标移动至画面左端的动作。
- 20Hex~7FHex： 通用 ASCII 字符区域。
- 80Hex~FFHex： 扩充字符区，可使用 ESC t n 指令从系统提供的多种扩充字符集中指定要使用的字符集。变更扩充字符集的操作不影响已经显示的内容。

n	文字種
0	PC437(USA: Standard Europe)
1	日文片假名
2	PC850(Multilingual)
3	PC860(Portuguese)
4	PC863(Canadian-French)
5	PC865(Nordic)
16	WPC1252
17	PC866(Cyrillic #2)
18	PC852(Latin 2)
19	PC858

### 5.5.2 指定国际字符集（特殊字符集）

指定国际字符集，是将通用字符区域（20Hex~7FHex）的部分特殊字符设定为特定国家地区通用格式的操作。例如设定为日本(08H)的话，反斜线符号（字符代码 5CHex）“\”的显示被设定为日本格式的“/”。

设定国际字符集使用指令 ESC R n。

n	区域	00H	01H	02H	03H	04H	05H	06H	07H	08H	09H	0AH	0BH	0CH	0DH
0	美国														
1	法国	23H	#	#	#	#	#	#	#	#	#	#	#	#	#
2	德国	24H	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$
3	英国	40H	@	@	@	@	@	@	@	@	@	@	@	@	@
4	丹麦 I	5BH	[	°	A	[	A	°	i	[	A	A	i	i	[
5	瑞典	5CH	\	ö	\	ö	\	ö	\	ö	\	ö	\	ö	\
6	意大利	5DH	]	ö	]	A	A	ö	ö	]	A	A	ö	ö	]
7	西班牙 I	5EH	^	^	^	^	^	^	^	^	^	^	^	^	^
8	日本	60H	^	^	^	^	^	^	^	^	^	^	^	^	^
9	挪威	60H	^	^	^	^	^	^	^	^	^	^	^	^	^
10 (0AH)	丹麦 II	7BH	^	^	^	^	^	^	^	^	^	^	^	^	^
11 (0BH)	西班牙 II	7CH	^	^	^	^	^	^	^	^	^	^	^	^	^
12 (0CH)	拉丁美洲	7DH	^	^	^	^	^	^	^	^	^	^	^	^	^
13 (0DH)	韩国	7EH	^	^	^	^	^	^	^	^	^	^	^	^	^

执行该指令不影响既有的显示内容。



## 5.5.3 16x16dot JIS, 简体字, 繁体字, 韩国语 (仅限 GU-79xx)

(双字节文字)

GU-79xx 系列搭载有 16x16 的点阵字库, 每个文字可通过由 2 个字节构成的字符代码来调用显示。在使用 16x16 点阵字库前首先需要做如下的系统设定, 这个设定不会影响既有的显示内容。根据需要随时改变设定, 就可以实现在一屏画面上同时显示多种语言文字。

使用双字节 (除字库选择以外其他操作相同)

设定为显示日文

1FH, 28H, 67H, 01H, 02H ‘ 选择 8x16 点阵字符  
1FH, 28H, 67H, 02H, 01H ‘ 指定双字节模式  
1FH, 28H, 67H, 0FH, 00H ‘ 指定日文  
88H, A2H ‘ 显示 “阿”

设定为显示韩文

1FH, 28H, 67H, 01H, 02H ‘ 选择 8x16 点阵字符  
1FH, 28H, 67H, 02H, 01H ‘ 指定双字节模式  
1FH, 28H, 67H, 0FH, 01H ‘ 韩国语  
写入 2 个字节的字符代码

设定为显示中文简体

1FH, 28H, 67H, 01H, 02H ‘ 选择 8x16 点阵字符  
1FH, 28H, 67H, 02H, 01H ‘ 指定双字节模式  
1FH, 28H, 67H, 0FH, 02H ‘ 简体字  
写入 2 个字节的字符代码

设定为显示中文繁体

1FH, 28H, 67H, 01H, 02H ‘ 选择 8x16 点阵字符  
1FH, 28H, 67H, 02H, 01H ‘ 指定双字节模式  
1FH, 28H, 67H, 0FH, 03H ‘ 繁体字  
写入 2 个字节的字符代码

各字库的规格及代码范围如下:

字库	规格	双字节代码范围
JIS 日文汉字	JISX208 (Shift-JIS)	8140H~9FF0H, E040~EFFCH
韩文	KSX5601-87	A1A1H~FEFEH
简体中文	GB2312-80	A1A1H~FEFEH
繁体中文	Big-5	A140H~FEFEH

JIS 日文字库例

825x	1	2	3	4	5	6	7	8	9						
826x	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
827x	Q	R	S	T	U	V	W	X	Y	Z					
828x		a	b	c	d	e	f	g	h	i	j	k	l	m	n
829x	p	q	r	s	t	u	v	w	x	y	z				あ
82Ax	あ	い	う	え	お	か	き	く	け						
82Bx	げ	こ	さ	し	す	せ	そ	た	だ	ち					
82Cx	ち	っ	つ	づ	て	と	ど	な	に	ぬ	の	は	ば	ぱ	
88Ax	啞	娃	阿	哀	愛	挨	始	逢	葵	茜	穉	惡	握	渥	旭
88Bx	芦	鱒	梓	庄	幹	扱	宛	姐	蛇	飴	絢	綾	鮎	或	粟
88Cx	安	庵	按	暗	案	闇	鞍	杏	以	伊	位	依	偉	困	夷
88Dx	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃	萎
88Ex	謂	違	遺	医	井	亥	域	育	郁	磯	一	壹	溢	逸	稻
88Fx	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭		

韩文字库例

B0Ax		가	각	간	갈	갈	감	감	감	갇	갇	갇	갇	갇	갇
B0Bx	갈	갇	갈	개	객	객	객	객	객	갇	갇	갇	갇	갇	갇
B0Cx	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇
B0Dx	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇
B0Ex	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇
B0Fx	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇
B1Ax		감	감	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇
B1Bx	감	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇
B1Cx	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇
B1Dx	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇
B1Ex	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇
B1Fx	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇	갇

简体中文例

B0Ax		啊	阿	埃	挨	哎	唉	哀	皑	癌	蔼	矮	艾	碍	爱
B0Bx	鞍	氨	安	俺	按	暗	岸	胺	案	肮	昂	盎	凹	敖	熬
B0Cx	袄	傲	奥	懊	澳	芭	捌	扒	叭	吧	芭	八	疤	巴	拔
B0Dx	靶	把	耙	坝	霸	罢	爸	柏	百	摆	佰	败	拜	裨	斑
B0Ex	班	搬	扳	般	颁	板	版	扮	拌	伴	瓣	半	办	绊	邦
B0Fx	梆	榜	膀	绑	棒	磅	蚌	傍	谤	苞	胞	包	褒	剥	
B1Ax		薄	雹	保	堡	饱	宝	抱	报	暴	豹	鲍	爆	杯	碑
B1Bx	卑	北	辈	背	贝	钡	倍	狈	备	惫	焙	被	奔	苯	笨
B1Cx	崩	绷	甬	泵	蹦	迸	鼻	比	鄙	笔	彼	碧	蓖	蔽	毕
B1Dx	毙	恣	币	庇	痹	闭	蔽	弊	必	辟	臂	避	陛	鞭	边
B1Ex	编	贬	扁	便	变	卞	辨	辩	辨	遍	标	彪	膘	表	鳖
B1Fx	别	瘪	彬	斌	濒	滨	宾	宾	兵	冰	柄	丙	秉	饼	炳

繁体中文例

A74x	作	你	伯	低	伶	余	尙	佈	佚	兌	克	兔	兵	治	冷
A75x	判	利	刪	創	劫	助	努	劬	匣	即	卵	吝	吭	吞	否
A76x	呎	吧	呆	呃	吳	呈	呂	君	盼	告	吹	吻	吸	吮	訥
A77x	吠	吼	呀	吱	舍	吟	听	囟	困	囹	圉	坊	坑	址	圉
A7Ax		均	坎	圾	坐	坏	圻	壯	夾	妝	妒	妨	妞	妣	妙
A7Bx	妍	妤	妓	妊	妥	孝	孜	孚	幸	完	宋	宏	尪	局	尿
A7Cx	尾	岐	岑	岔	岌	巫	希	序	庇	床	廷	弄	弟	彤	尿
A7Dx	役	忘	忌	志	忍	忱	快	忸	忸	戒	我	抄	抗	抖	扶
A7Ex	抉	扭	把	扼	找	批	扳	抒	扯	折	扮	投	抓	抑	技
A7Fx	攻	攸	旱	更	束	李	杏	材	村	杜	杖	杞	杉	杆	杠
A84x	杓	宋	步	每	求	汞	沙	沁	沈	沉	沅	沅	沅	沅	沅
A85x	沌	汨	冲	沒	汽	沃	汲	汾	沅	沅	沅	沅	沅	沅	沅

完整的字库信息请参考相应规格书。

## 5.6 指令集

### 5.6.1 指令集 1 GU-7xxx 共通

名称	代码	功能
BS	08h	光标向左移动 1 个字符位置。
HT	09h	光标向右移动 1 个字符位置。
LF	0Ah	光标位置向下移动 1 行。
HOM	0Bh	光标位置移动到画面左上角。
CR	0Dh	光标位置移动到同一行左端。
CLR	0Ch	清除当前显示画面，并将光标移动到画面的左上角。
ESC	1Bh, ,	扩充指令集引导符
US	1Fh	扩充指令集引导符

### 5.6.2 指令集 2 扩充指令集 GU-7xxxx 共通

名称	代码	功能
初始化	1Bh 40h	软件方式的系统重置操作。 该指令不会重新读取跳线的设置状态。
设定光标	1Fh 24h xL xH yL yH	将光标移动到(x, y)位置。
光标 ON/OFF	1Fh 43h n	打开/关闭光标显示。 0: 不显示。(初始值) 1: 显示。
字符显示间距	1Fh 28h 67h 03h w	指定 1 字节字符的显示宽度。 w=0: 固定字间距 1、字符间隔 1dot 1: 固定字间距 2、字符间隔 2dot 2: 自动字间距 1、字符间隔 1dot 3: 自动字间距 2、字符间隔 2dot
放大字符	1Fh 28h 67h 40h x y	纵向、横向放大显示字符。

用户自定义字符	1Bh 26h a C1 C2 [data]	最大支持 16 个 RAM 用户自定义字符。在开始使用用户自定义字符之前，要首先通过【指定用户自定义字符】指令使定义完成的字符有效，这个操作不影响已经显示的内容。如果需要重新定义用户字符，需要首先清除当前的自定义字符。
清除用户自定义字符	1Bh 3Fh a c	清除当前的用户自定义字符。
指定用户自定义字符	1Bh 25h n	设定用户自定义字符的有效或者无效。该操作不影响已经显示的内容，只对执行该指令后的写入数据有效。 n=0: 用户自定义字符无效。 =1: 用户自定义字符有效。
指定国际字符集	1B 52h n	将字符代码 20h~7Fh 中的一部分特殊字符根据需要替换为不同国家地区版本。该指令不影响既有的显示内容。
指定扩充字符集	1Bh 74h n	指定字符代码 80h~FFh 显示用的扩充字符集。该指令不影响既有的显示内容。
指定显示模式-覆盖原文	1Fh 01h	设定显示模式为覆盖原文。
指定显示模式-纵向滚动	1Fh 02h	设定显示模式为纵向滚动。
指定显示模式-横向滚动	1Fh 03h	设定显示模式为横向滚动。
指定横向滚动速度	1Fh 73h n	指定横向滚动使用的时间 每次移动需要的时间为 $(T * (n-1))$ ，n=0 时速度最快，n=1 时滚动速度为 T/2。T 为时间常量，大约为 10~20mSec，根据模组型号不同而不同。
指定・解除反转显示模式	1Fh 72h n	指定・解除反转显示模式，对文字及图像显示均有效。该指令不影响既有的显示内容，仅对执行该指令以后的显示动作有效。
指定显示画面合成模式	1Fh 77h n	指定写入字符及图像时新写入内容和既有显示内容的逻辑运算方式（画面合成）。 n=0: 显示新写入内容。 1: OR 运算并显示运算结果。 2: AND 运算并显示运算结果。 3: EX-OR 运算并显示运算结果。
设定显示亮度	1Fh 58h n	设定显示画面的亮度。 亮度 = $n/8$ 其中 $n=1\sim 8$ ，8 为最大亮度 100%。

延时等待指令	1Fh 28h 61h 01h t	延时等待指令。 延时等待大约 $t / 2$ 秒，延时结束前不接受新指令。 $t = 0 \sim 255$
滚屏显示特效	1Fh 28h 61h 10h wL wH cL cH s n	按照指定的次数执行移动显示画面的动作。 指定适当的移动次数（画面竖向字节数的整数倍） 即可实现滚屏显示。
闪烁显示特效	1Fh 28h 61h 11h p t1 t2 c	显示画面的闪烁显示特效。
屏幕保护指令	1Fh 28h 61h 40h p	控制显示用电源的 ON/OFF 以及启动屏幕保护显示， 执行该指令后的屏幕保护状态将持续到新的数据写入为止。 p=0: 显示用电源 OFF 1: 显示用电源 ON 2: 全点亮 3: 全熄灭 4: 重复反转显示。
定义用户窗口	1Fh 28h 77h n a b [窗口坐标]	定义・解除用户窗口，对既有显示内容无有影响。
设定当前窗口	1Fh 28h 77h 01h a	选择一个用户窗口作为当前活动窗口。 a = 0: 指定基础窗口为当前活动窗口 1~4: 指定用户窗口 1~4 为当前活动窗口
设定当前窗口快捷指令	10h or 11h or 12h or 13h or 14h	快速设定当前窗口的快捷指令，单字节指令。
设定基础窗口模式	1Fh 28h 77h n a	指定基础窗口可以使用的显示内存范围。 a = 0: 设定显示内存 RAM 的可见区域作为基础窗口的可用范围。 1: 设定全部显示内存 RAM 作为基础窗口的可用范围。 光标移动范围受限于这里设定的基础窗口范围。
实时位图图像显示	1Fh 28h 66h 11h xL xH yL yH g d1...dk	以当前光标位置为起点显示写入的图像。

## 5.6.3 指令集 3 扩充指令集 FROM 操作 GU-79xxx

注意：FROM 有擦写次数的限制，请避免在日常使用中进行频繁擦写操作。在擦写状态时不可断开模组电源，否则有可能造成 FROM 数据的丢失。

FROM 重写数据指令以及 FROM 擦除数据指令为工厂保留指令，用户不可使用。

名称	代码	功能
进入 FROM 写模式	1Ch 7Ch 4Dh D0h d1...d6	进入 FROM 写模式
定义 FROM 位图	42h k n d1...d32768	在 FROM 内定义位图图像。 该指令必须在 FROM 写模式下使用。
FROM 校验和比较指令	53h k d1...d4 dm	在执行 FROM 重写数据以及定义 FROM 位图指令后对实际写入的数据和指令中的数据参数 d1...d4 进行校验和比较。不一致的话在画面上显示错误信息并延长 BUSY 时间到 2 秒左右。
结束 FROM 写模式	45h k	结束 FROM 写模式并执行初始化。
显示已定义的位图	1Fh 28h 66h 10h m aL aH aE ySL ySH xL xH yL yH g	在当前光标位置显示 FROM 中已定义的位图图像。

## 5.6.4 指令集 4 扩充指令集 双字节文字操作 GU-79xx

名称	代码	功能
选择字符大小	1Fh 28h 67h 01h m	选择单字节基本字符大小。 m=1: 5x7 点阵 2: 8x16 点阵
指定・解除双字节字符模式	1Fh 28h 67h 02h m	指定・解除双字节字符模式。 m=0h: 解除双字节字符模式 1h: 指定双字节字符模式
指定双字节字符种类	1Fh 28h 67h 0Fh m	指定双字节字符种类。 m=0h: 日语 1h: 韩国语 2h: 简体字 3h: 繁体字

### 5.6.5 指令集 5 GU-7xxxB 追加指令 点单位任意位置显示 (GU-7000B、7900B 系列追加指令)

此前的模组产品，其纵向的位置指定都是以字节（8dot）为单位的，新的 GU-7000B 系列开始支持以 1dot 为单位的位置指定。

名称	代码	功能
显示已定义位图一点单位任意位置	1Fh 28h 64h 20h xPL xPH yPL yPH m aL aH aE ySL ySH x0L x0H y0L y0H xL xH	GU-7000B: 显示定义在显示内存区域（非可见区域等）的位图，可以以点单位指定任意位置作为显示起始位置。 GU-7900B: 显示定义在显示内存区域（非可见区域等）或 FROM 内的位图，可以以点单位指定任意位置作为显示起始位置。
实时位图显示一点单位任意位置	1Fh 28h 64h 21h xPL xPH yPL yPH xL xH yL yH g d(1)···d(k)	实时显示位图，可以以点单位指定任意位置作为显示起始位置。
字符显示一点单位任意位置	1Fh 28h 64h 30h xPL xPH yPL yPH m bLen d(1)···' d(bLen)	字符显示，可以以点单位指定任意位置作为显示起始位置。 位置指定为 (xP) =FFFFh 的话则自动接着前面的位置显示。 字符放大以及粗体字指定无效



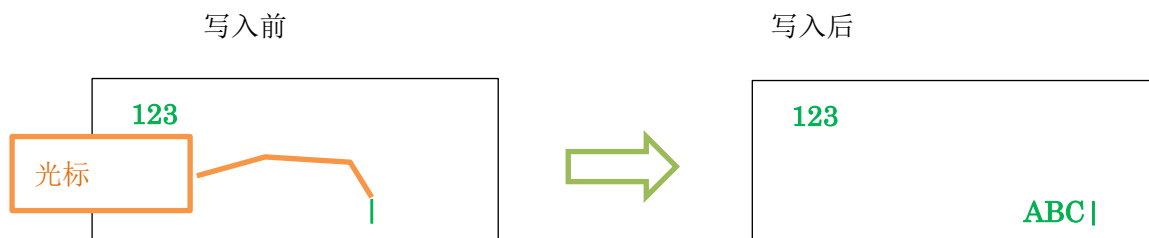
## 5.7 光标移动及显示模式

指定显示模式实际上就是指定当光标移动到了画面末尾时的下一步的移动方式。

1Fh 01h	指定显示模式—覆盖原文
1Fh 02h	指定显示模式—纵向滚动
1Fh 03h	指定显示模式—横向滚动

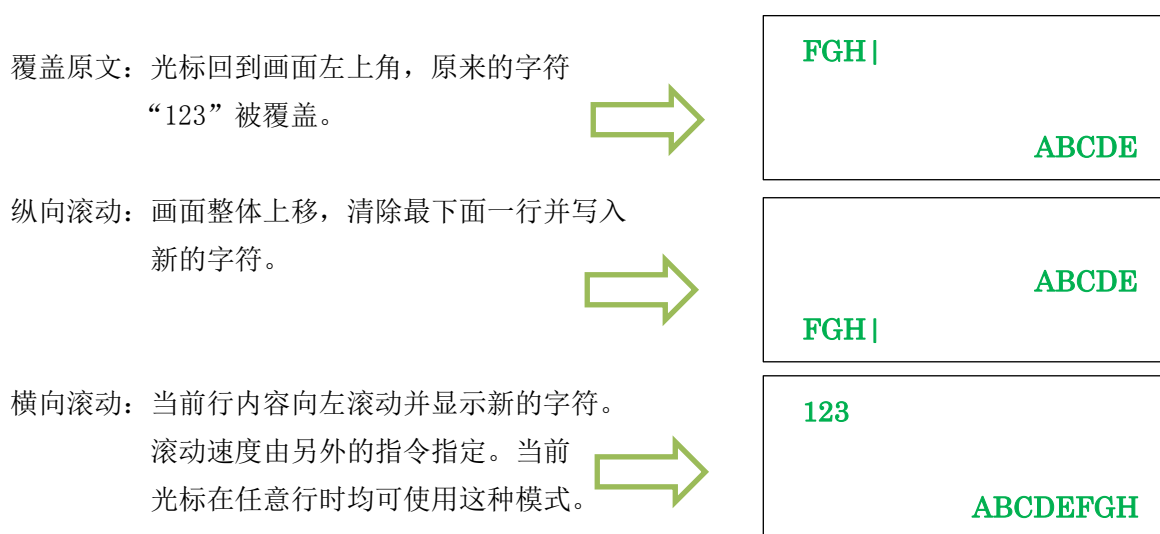
当写入字符数据时在当前光标位置显示写入的字符，同时光标向前移动一个字符位置。

下图示例为写入字符“ABC”时光标的移动情形:



根据指定的显示模式不同此后的显示效果随之变化。

下图示例为接着写入字符“DEFGH”时在 3 种显示模式下的不同的显示效果:





## 5.8 Windows PC 环境下使用 Microsoft 产品 Visual Studio 2010 的程序制作举例

### 5.8.1 使用 Visual C# 2010 集成开发环境，通过串口连接 GU-7000 模组并显示字符。

该示例程序可在 C#2010 Express Edition(免费版)环境下制作运行。

首先从 Microsoft 网站下载并安装 C#2010。

选择新项目→Windows 窗体应用程序，这样就打开一个新的应用程序窗口。

双击工具菜单中的 SerialPort 工具，然后双击 Button 工具，并把创建的 Button 移动到合适的位置。

请参考右图示例 Form1.cs。

双击窗口中 Button 以外的位置打开载入窗体事件处理器 (Form1\_Load) 并输入以下程序内容。其中程序开始处的串口号 COM3 请更改为用户实际使用的串口号。

----- 以下为程序内容 -----

```
private void Form1_Load(object sender, EventArgs e)
{
    this.serialPort1.PortName = "COM3";
    this.serialPort1.BaudRate = 38400;
    this.serialPort1.DataBits = 8;
    this.serialPort1.StopBits = System.IO.Ports.StopBits.One;
    this.serialPort1.Parity = System.IO.Ports.Parity.None;
    this.serialPort1.DiscardNull = false;
    this.serialPort1.Handshake = System.IO.Ports.Handshake.RequestToSend;
    if (this.serialPort1.IsOpen) { this.serialPort1.Close(); }
}
```

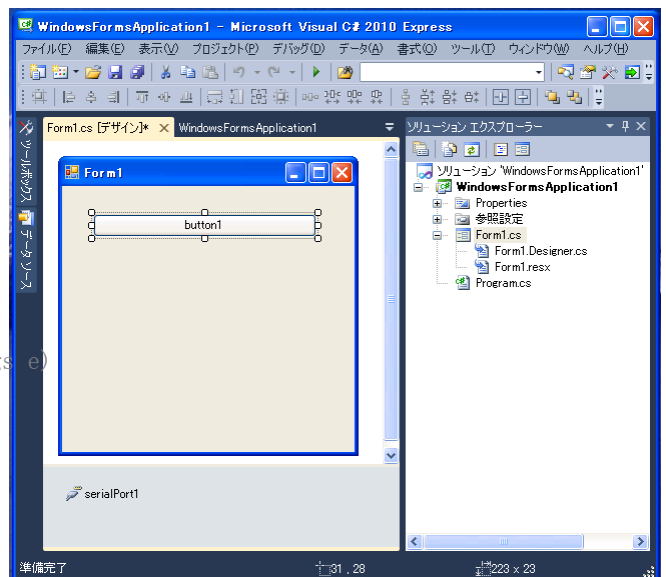
----- 程序结束 -----

点击 Form1.cs[Design]标签返回窗体设计画面。

双击 Button1 打开按钮点击事件处理器 (button1\_Click) 并输入下面的程序：

----- 以下为程序内容 -----

```
private void button1_Click(object sender, EventArgs e)
{
    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write("Hello World");
        this.serialPort1.Close();
    }
}
```



----- 程序结束-----

整体程序如下：

#### LIST C# 示例程序

```
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 WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            this.serialPort1.PortName = "COM3"; //输入实际使用的端口号。
            this.serialPort1.BaudRate = 38400;
            this.serialPort1.DataBits = 8;
            this.serialPort1.StopBits = System.IO.Ports.StopBits.One;
            this.serialPort1.Parity = System.IO.Ports.Parity.None;
            this.serialPort1.DiscardNull = false;
            this.serialPort1.Handshake = System.IO.Ports.Handshake.RequestToSend;
            if (this.serialPort1.IsOpen) { this.serialPort1.Close(); }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            this.serialPort1.Open();
            if (this.serialPort1.IsOpen)
            {
                this.serialPort1.Write("Hello World");
                this.serialPort1.Close();
            }
        }
    }
}
```

请参考注释

----- 程序结束-----

注：此行的 COM3 请修改为实际使用的端口名称。

```
this.serialPort1.PortName = "COM3";
```

全部输入完成后按 F5 执行编译，编译完成会自动打开窗体 Form1，请点击 Button1 观察程序运行效果。  
VFD 模組上显示 “Hello World” 即为运行成功。

下面的章节中我们将在该示例程序的基础上追加部分其他的 VFD 模组驱动示例程序。

## 5.9 示例程序

上面章节中使用 C# 示范了显示 “Hello World” 字符的过程，下面我们来看其他各种功能的 C# 示范例。

### 5.9.1 清除画面

```
private void ClearScreen()
{
    byte[] bb = new byte[1];
    bb[0] = (byte) 0x0c;

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen) {
        this.serialPort1.Write(bb, 0, 1);
        this.serialPort1.Close(); }
}
```

### 5.9.2 设定光标位置

移动光标位置到 (int X, int Y)。请注意 Y 是以 8bit 为单位的。

```
/* Move Cursot to (X, Y). Y is in Byte. */
private void moveCursor(int X, int Y)
{
    byte[] bb = new byte[6];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x24;
    bb[2] = (byte)(X % 0x100);
    bb[3] = (byte)(X / 0x100);
    bb[4] = (byte)(Y % 0x100);
    bb[5] = (byte)(Y / 0x100);

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(bb, 0, 6);
        this.serialPort1.Close();
    }
}
```

### 5.9.3 放大字符

纵横方向整数倍放大字符。

```
/* @Font Magnified */
private void fontMagnified(int X, int Y)
{
    byte[] bb = new byte[6];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x28;
    bb[2] = (byte)0x67;
    bb[3] = (byte)0x40;
    bb[4] = (byte)X;
    bb[5] = (byte)Y;
```

```

        this.serialPort1.Open();
        if (this.serialPort1.IsOpen)
        {
            this.serialPort1.Write(bb, 0, 6);
            this.serialPort1.Close();
        }
    }
}

```

#### 5.9.4 设定字符间距（英数字）

设定字符间距的模式。使用自动字符间距功能可以增加平均可显示文字数。

```

/* Set Font Size **
**
** w=0: Fixed Font Size with 1 dot space
** w=1: Fixed Font Size with 2 dot space
** w=2: Proportional Font Size with 1 dot space
** w=3: Proportional Font Size with 2 dot space
*/
private void FontWidth(int w)
{
    byte[] bb = new byte[5];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x28;
    bb[2] = (byte)0x67;
    bb[3] = (byte)0x03;
    bb[4] = (byte)(w % 0x100);

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(bb, 0, 5);
        this.serialPort1.Close();
    }
}

```

#### 5.9.5 设定亚洲字符显示模式(仅适用 7900 系列)

设定开启CJK（Chinese-Japanese-Korean）字符的显示。

```

/* Setup Kanji
**
** Set up Kanji Display mode.
**
** cjk=0:Japanese,
**      1:Korean,
**      2:Simplified Chinese
**      3:Traditional Chinese
*/

private void CJK_setup( int cjk)
{
    byte[] bb = new byte[15];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x28;

```

```
bb[2] = (byte)0x67;
bb[3] = (byte)0x01;
bb[4] = (byte)0x02;
bb[5] = (byte)0x1f;
bb[6] = (byte)0x28;
bb[7] = (byte)0x67;
bb[8] = (byte)0x02;
bb[9] = (byte)0x01;
bb[10] = (byte)0x1f;
bb[11] = (byte)0x28;
bb[12] = (byte)0x67;
bb[13] = (byte)0x0f;
bb[14] = (byte)cjk;

this.serialPort1.Open();
if (this.serialPort1.IsOpen)
{
    this.serialPort1.Write(bb, 0, 15);
    this.serialPort1.Close();
}
}
```

#### 5.9.6 使用中文汉字字符（简体）显示功能

显示中文简体汉字时显示器要求的输入代码 GB2312 代码，由于 C#内部为 Unicode 编码，所以需要修改为 GB2312 代码以便正确输入，下面是具体的示例。

```
private void button2_Click(object sender, EventArgs e)
{
    const int GB = 0;

    CJK_setup(GB); /*Setup GB Kanji*/

    string str = "中文简体";
    Encoding gbEnc = Encoding.GetEncoding("GB2312");
    int NumberOfBytes = gbEnc.GetByteCount(str);
    byte[] bytes = gbEnc.GetBytes(str);

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(bytes, 0, NumberOfBytes);
        this.serialPort1.Close();
    }
}
```

#### 5.9.7 显示图片

实时显示位图图片。写入位置在可见显示区域的话为实时显示的效果，写入位置在非可见显示区域的话可以使用滚动显示的功能使其显示在可见区域。

```
/* Realtime Bitimage Display
 *
 * image: bitmap image
 * X      : Horizontal size in Bit
 * Y      : Vertial size in Byte (8Bit)
 *
 */
```

```

*/
private void DrawBitmap(byte[] image, int X, int Y)
{
    byte[] bb = new byte[9];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x28;
    bb[2] = (byte)0x66;
    bb[3] = (byte)0x11;

    bb[4] = (byte)(X % 256);
    bb[5] = (byte)(X / 256);
    bb[6] = (byte)(Y % 256);
    bb[7] = (byte)(Y / 256);
    bb[8] = (byte)0x01;

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(bb, 0, 9);
        this.serialPort1.Close();
    }
    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(image, 0, X * Y);
        this.serialPort1.Close();
    }
}

```

#### 5.9.8 图像显示实例

下面是打开图像文件并调用函数 DrawBitmap() 进行显示的示例程序。首先在画面上追加 Button5，其点击事件为本图像显示例程。请注意 BMP 图像尺寸不可超过 VFD 显示器的限制，另外纵向 dot 数必须是 8 的整数倍否则余数部分无法显示。

```

private void button5_Click(object sender, EventArgs e)
{
    int i, j;
    byte b2;

    OpenFileDialog openDia = new OpenFileDialog();

    if (openDia.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        Bitmap bmp = new Bitmap(openDia.FileName);

        // @Show Bitmap on window
        pictureBox1.Image = bmp;

        // Transform Bitmap file into byte array
        Byte[] bb = new Byte[bmp.Width * (bmp.Height/8)];

        for (i = 0; i < bmp.Width; i++)
        {
            for (j = 0; j < (bmp.Height / 8); j++)

```

```
{
    b2 = 0;
    if (bmp.GetPixel(i, j * 8).G < 128) { b2 = 1; } b2 += b2;
    if (bmp.GetPixel(i, j * 8 + 1).G < 128) { b2++; } b2 += b2;
    if (bmp.GetPixel(i, j * 8 + 2).G < 128) { b2++; } b2 += b2;
    if (bmp.GetPixel(i, j * 8 + 3).G < 128) { b2++; } b2 += b2;
    if (bmp.GetPixel(i, j * 8 + 4).G < 128) { b2++; } b2 += b2;
    if (bmp.GetPixel(i, j * 8 + 5).G < 128) { b2++; } b2 += b2;
    if (bmp.GetPixel(i, j * 8 + 6).G < 128) { b2++; } b2 += b2;
    if (bmp.GetPixel(i, j * 8 + 7).G < 128) { b2++; }

    bb[i * (bmp.Height / 8) + j] = b2;
}

// Move Cursor to Home
moveCursor(0, 0);

// Call Realtime bitmap display
DrawBitmap(bb, bmp.Width, bmp.Height / 8);
}
openDia.Dispose();
}
```

### 5.9.9 滚动显示图像

通过移位显示内存数据从而实现滚屏显示的效果。

```
/*
 * Graphics Horizontal scroll
 *
 */
private void GraphicsHorizontalScroll(int skip, int number, int speed)
{
    byte[] bb = new byte[9];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x28;
    bb[2] = (byte)0x61;
    bb[3] = (byte)0x10;

    bb[4] = (byte)(skip % 256);
    bb[5] = (byte)(skip / 256);
    bb[6] = (byte)(number % 256);
    bb[7] = (byte)(number / 256);
    bb[8] = (byte)speed;

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(bb, 0, 9);
        this.serialPort1.Close();
    }
}
```



#### 5.9.10 滚屏显示示例程序

编程为Button6点击事件。

Xsize 和 Ysize 请替换为实际使用的显示器的纵横 dot 数。

```
private void button6_Click(object sender, EventArgs e)
{
    const int Xsize = 256; /* Horizontal screen size of display*/
    const int Ysize = 64; /* Vertical screen size of display*/
    const int speed = 1;

    /*
     * Scroll entire display
     */

    GraphicsHorizontalScroll(Ysize / 8, Xsize, speed);
}
```

#### 5.9.11 显示非可见区域画面

使用「5.9.9 滚动显示图像」的滚屏显示指令，实现显示内存非可见区域的显示。

编程为Button7的点击事件。

Xsize 和 Ysize 请替换为实际使用的显示器的纵横 dot 数。

```
private void button7_Click(object sender, EventArgs e)
{
    const int Xsize = 256; /* Horizontal screen size of display*/
    const int Ysize = 64; /* Vertical screen size of display*/
    const int speed = 1;

    /*
     * Show hidden area
     */

    GraphicsHorizontalScroll(Ysize/8 * Xsize, 1, speed);
}
```

#### 5.9.12 字符的滚动显示

首先需要设定显示模式为横向滚屏模式，这样当光标位置到达显示画面的最右端后，在写入下一个字符代码时系统自动执行滚动显示动作。

请同时设定好滚屏的速度。

```
/*
 * Horizontal Scroll Mode
 */
```

```
private void HorizontalScrollMD3()
{
    byte[] bb = new byte[2];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x03;

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(bb, 0, 2);
        this.serialPort1.Close();
    }
}

/*
 * Horizontal Scroll Speed
 */
private void HorizontalScrollSpeed(int speed)
{
    byte[] bb = new byte[3];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x73;
    bb[2] = (byte)(speed % 32);

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(bb, 0, 3);
        this.serialPort1.Close();
    }
}
```

### 5.9.13 字符滚动显示的示例程序

编程为Button8的点击事件。

```
private void button8_Click(object sender, EventArgs e)
{
    const int speed = 2;

    HorizontalScrollMD3();
    HorizontalScrollSpeed(speed);

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write("Horizontal Scroll Mode Test.....");
        this.serialPort1.Close();
    }
}
```

### 5.9.14 使用用户自定义窗口功能。

用户自定义窗口是由用户在显示画面上自定义的相对独立的显示窗口，使用时首先定义然后选择作为当

前活动窗口即可激活，此后的所有显示动作都在该指定窗口内进行。系统最多支持 4 个这样的用户自定义窗口。

创建用户自定义窗口

```
private void DefineUserWindow(int a, int X, int Y, int W, int H)
{
    byte[] bb = new byte[14];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x28;
    bb[2] = (byte)0x77;
    bb[3] = (byte)0x02;
    bb[4] = (byte)a;
    bb[5] = (byte)1;
    bb[6] = (byte)(X % 256);
    bb[7] = (byte)(X / 256);
    bb[8] = (byte)(Y % 256);
    bb[9] = (byte)(Y / 256);
    bb[10] = (byte)(W % 256);
    bb[11] = (byte)(W / 256);
    bb[12] = (byte)(H % 256);
    bb[13] = (byte)(H / 256);

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(bb, 0, 14);
        this.serialPort1.Close();
    }
}
```

删除用户自定义窗口

```
private void CancelUserWindow(int a)
{
    byte[] bb = new byte[6];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x28;
    bb[2] = (byte)0x77;
    bb[3] = (byte)0x02;
    bb[4] = (byte)a;
    bb[5] = (byte)0;

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write(bb, 0, 6);
        this.serialPort1.Close();
    }
}
```

指定自定义窗口为当前窗口

```
private void SelectCurrentUserWindow(int a)
{
    byte[] bb = new byte[5];

    bb[0] = (byte)0x1f;
    bb[1] = (byte)0x28;
```

```
bb[2] = (byte)0x77;
bb[3] = (byte)0x01;
bb[4] = (byte)a;

this.serialPort1.Open();
if (this.serialPort1.IsOpen)
{
    this.serialPort1.Write(bb, 0, 5);
    this.serialPort1.Close();
}
}
```

#### 5.9.15 使用用户自定义窗口示例程序

编程为Button9的点击事件。

在显示画面左上角定义一个50x16点阵的用户窗口并在该窗口内显示“Window”字样。

继续进行文字的滚动显示以及图像显示，可以看到显示动作范围被限定在指定的用户窗口内。

```
private void button9_Click(object sender, EventArgs e)
{
    const int UserWindow1 = 1;

    DefineUserWindow(UserWindow1, 0, 0, 50, 2);
    SelectCurrentUserWindow(UserWindow1);

    this.serialPort1.Open();
    if (this.serialPort1.IsOpen)
    {
        this.serialPort1.Write("Window");
        this.serialPort1.Close();
    }
}
```

需要跳出当前指定的自定义窗口的话执行 SelectCurrentUserWindow(0);即可。

## 5.10 使用并口通信的示例程序

下面是在嵌入式应用中以 MPU 作为 Host 使用 8bit 并口通信方式向 VFD 模组写入 1 个字节的程序示例。使用时请根据具体情况加入适当的延时。

硬件回路连接请参考 4.5.5 并口输入 连接示例 2 使用 BUSY 信号

-----数据写入程序示例-----

```
Void GU7000_out( char data)
{
    do {} while ( BUSY == 1); /* Wait for ready */
    /* wait 1.5uSec, if necessary */
    DataLines = data;        /* Output data */

    WR = 0; /* set WR Low */
    /* wait 100nS, if necessary */
    WR = 1; /* set WR High */
}
```

作为 Host 的  
MPU 速度很高的  
话请根据需要加  
入适当延时。

作为 Host 的  
MPU 速度很高的  
话请根据需要加  
入适当延时。

## 6 故障排查

### 6.1 关于 BUSY 信号

并口用的 BUSY 信号和串口用的 SBUSY 是不同的两种信号，不可混为一谈。另外使用并口用的 BUSY 信号的话，需要将跳线 JRB 的 C-B 端子短接，该端子出厂时默认为 OPEN 状态。

### 6.2 关于 RESET

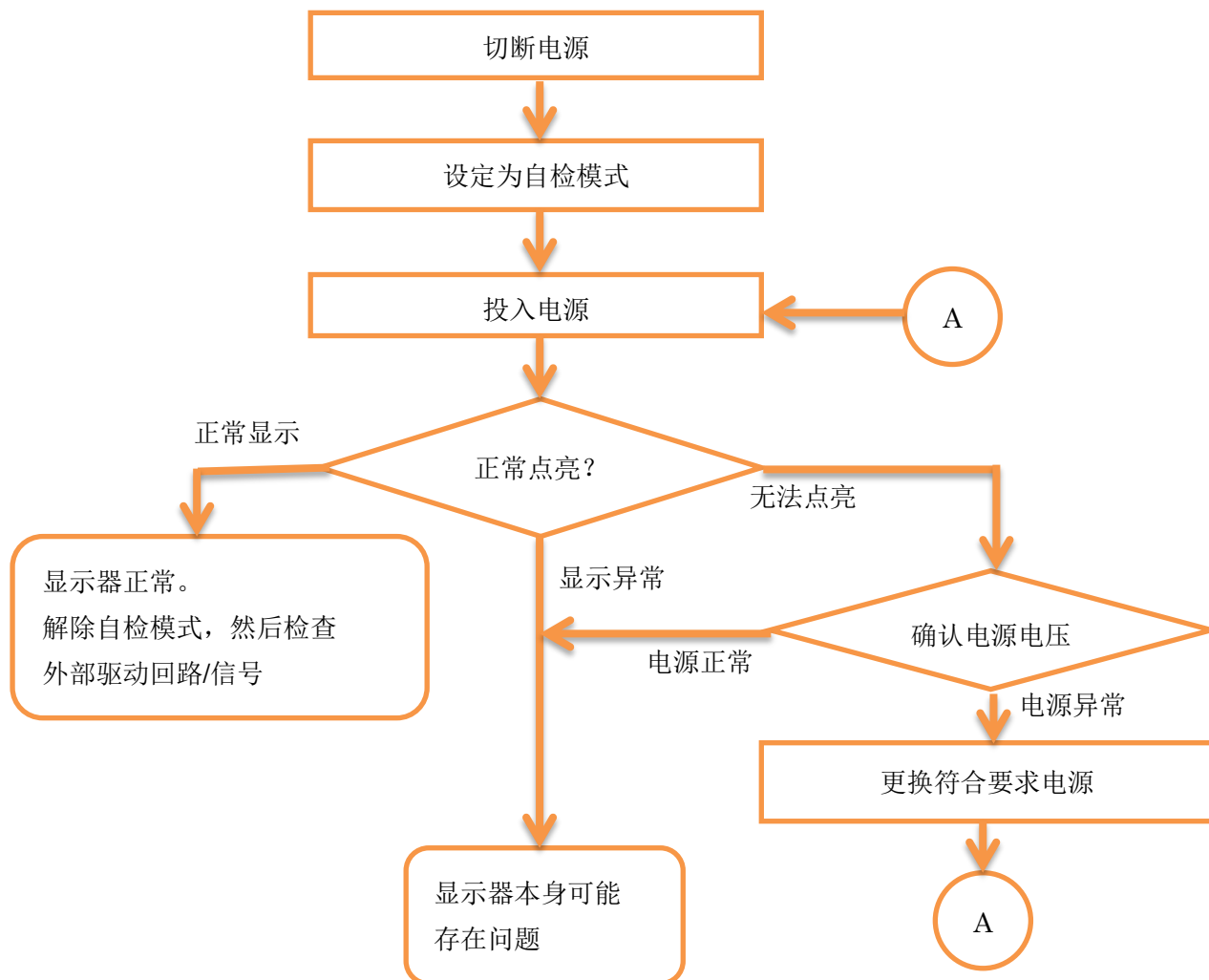
电源投入时显示器内部控制器自动执行初始化处理，在该处理完成前系统保持并输出 BUSY 状态，无法接受外部操作指令，所以请不要在此期间向显示器写入任何数据。

### 6.3 显示器无法点亮---自检模式

正常情况下向显示器写入 ASCII 代码即可显示英文及数字而无需任何特别的设定。如果完全无法点亮，则需要判断是外部驱动回路的问题或者是显示器本身的问题，为此该系列显示器内置了加电自检模式。

自检模式的设定方法请参考「エラー！参照元が見つかりません。自检模式的设定方法」。

自检模式

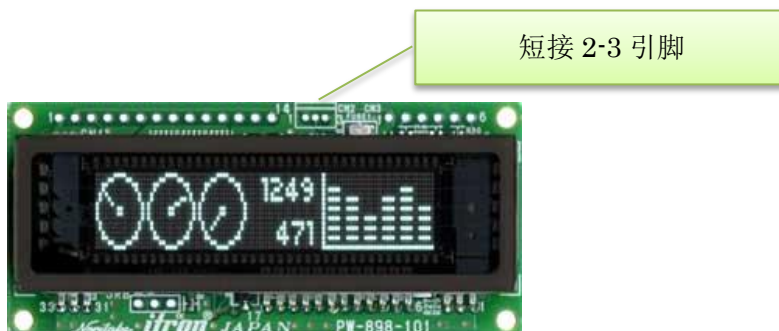


#### 6.4 自检模式的设定方法

自检模式的设定方法因显示器型号而异。如果是搭载有电源专用的 3 引脚插座的型号，则将中央引脚对地短接然后重新加电即可；没有搭载该电源插座的型号，需要通过跳线或跳线开关 DIP-SW 来完成设置。

\*以下照片非等比例缩放，仅供参考。

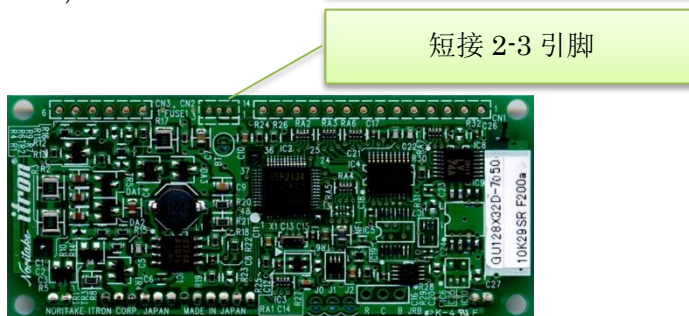
##### 6.4.1 GU112X16G-7000, -7003, -7900, -7000B, -7003B, -7900B



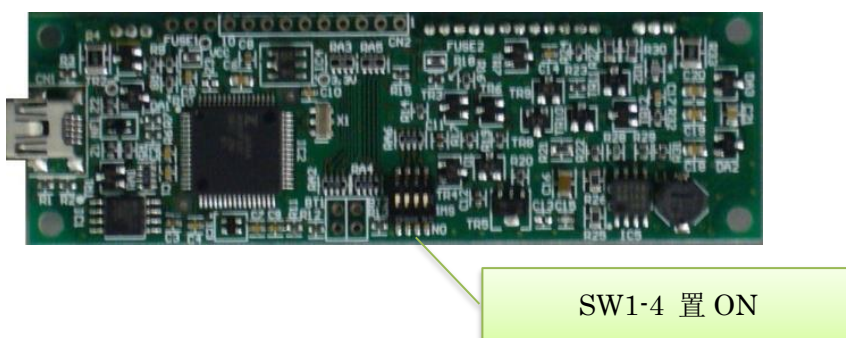
##### 6.4.2 GU128X32D-7000, -7003, -7900, -7000B, -7003B, -7900B



##### 6.4.3 GU128X32D-7050, -7950

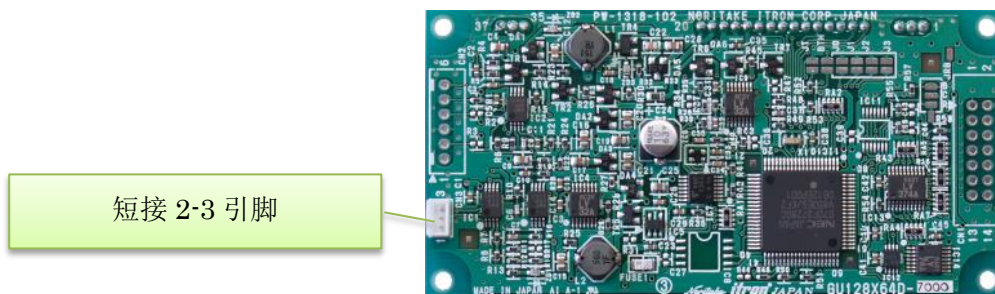


##### 6.4.4 GU128X32D-7901

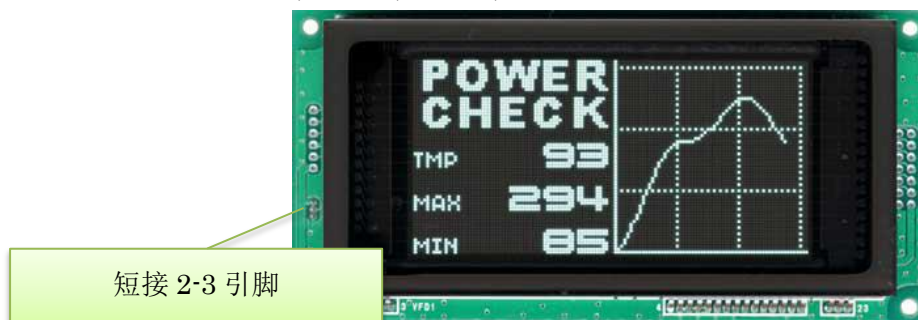




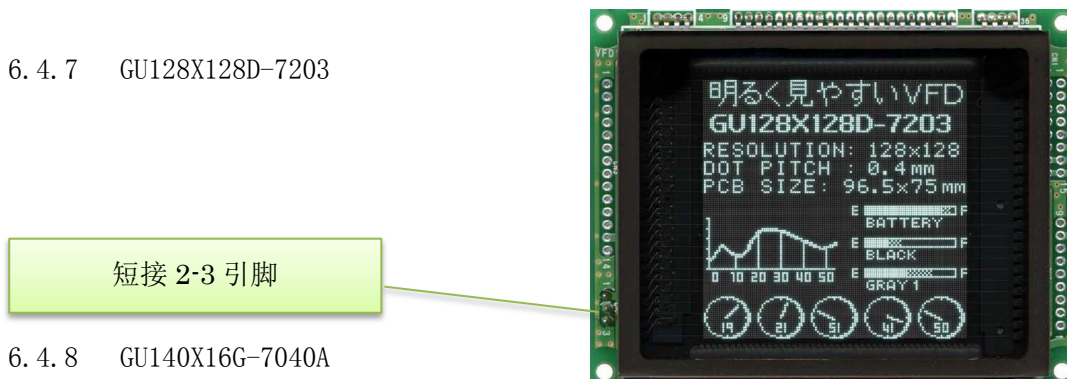
6.4.5 GU128X64D-7000, -7003, -7900



6.4.6 GU128X64F-7000, -7003, -7900, -7900BX



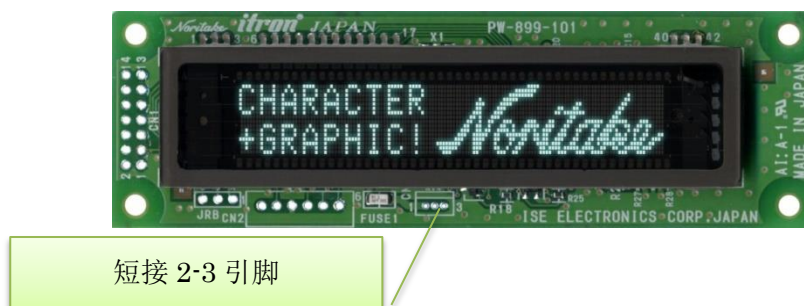
6.4.7 GU128X128D-7203



6.4.8 GU140X16G-7040A



6.4.9 GU140X16G-7000, -7003, -7900, -7903, -7042, -7000B, -7003B



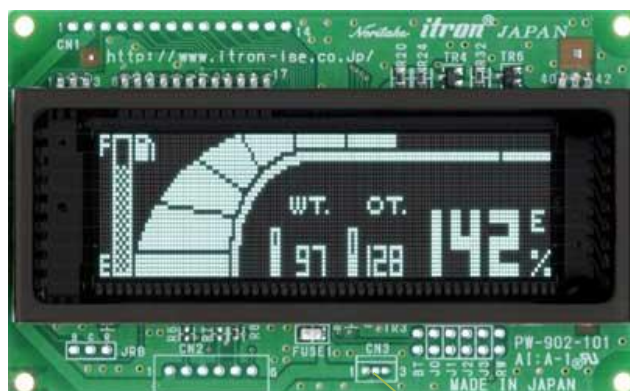


6.4.10 GU140X16J-7000, -7003, -7000B, -7003B, -7900B



短接 2-3 引脚

6.4.11 GU140X32F-7000, -7003, -7900, -7903, -7032, -7000B, -7003B, -7900B



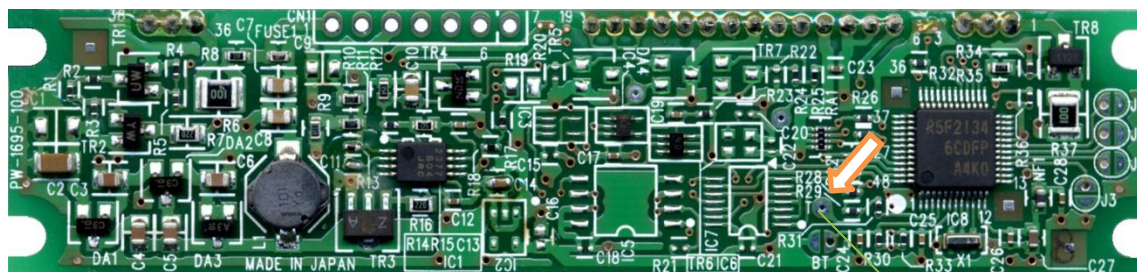
短接 2-3 引脚

6.4.12 GU140X32F-7050, GU140X32F-7053, GU140X32F-7950



短接 2-3 引脚

6.4.13 GU144X16D-7053B

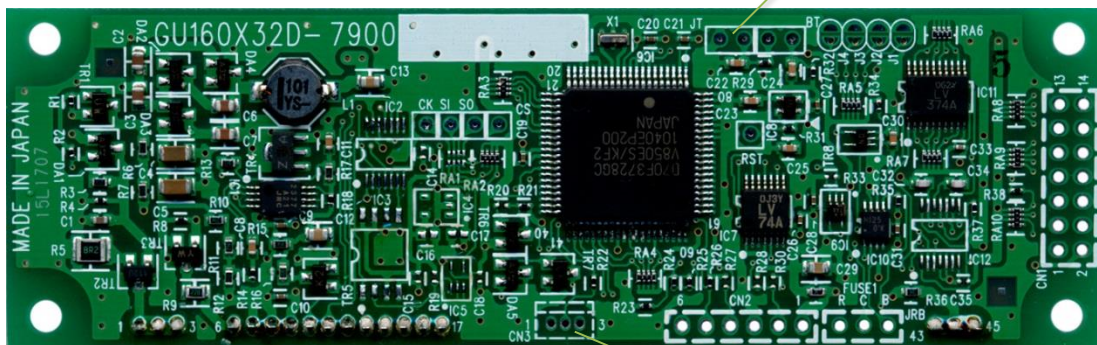


短接焊盘 T 和  
GND

#### 6.4.14 GU160X32D-7050, -7950

两种方式任取一种。

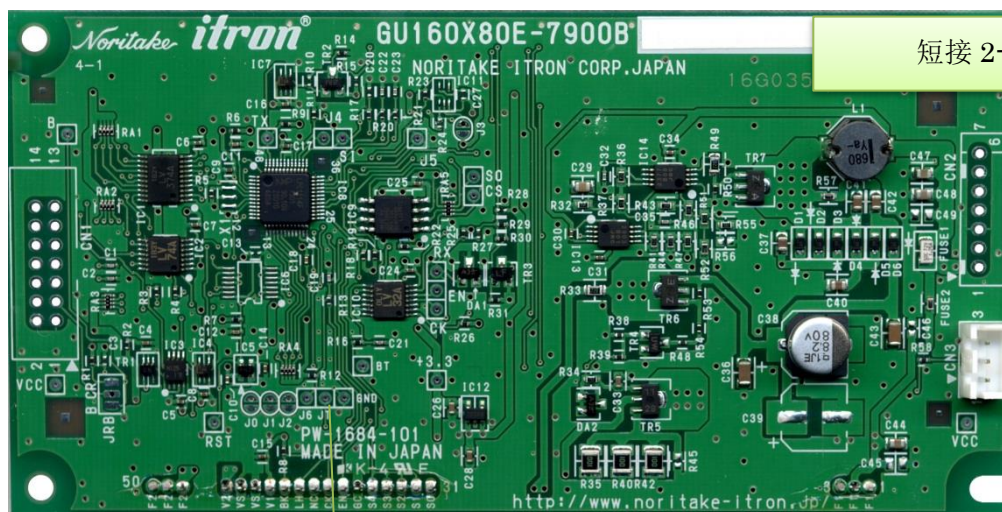
短接焊盘 JT



#### 6.4.15 GU160X80E-7900B

两种方式任取一种。

短接 2-3 引脚



短接 JT 和 GND

#### 6.4.16 GU256X64C-7000, -7003, -7900

短接 2-3 引脚

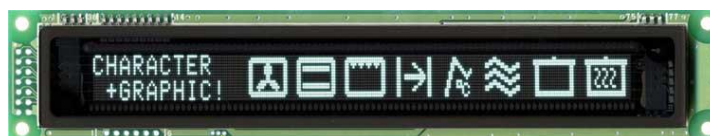


6.4.17 GU256X64D-7000, -7900



短接 2-3 引脚

6.4.18 GU280X16G-7000, -7003



短接 2-3 引脚



## 7 技术支持

我司网站提供有初期评测以及数据生成用支援工具软件，欢迎使用。如需其他支持请咨询营业人员或我司技术支持人员。

我司网站：

<http://www.noritake-itrn.sh.cn/>

技术支持：

E-MAIL: [vfd@noritake.sh.cn](mailto:vfd@noritake.sh.cn)

### C 言語用库函数

帮助客户在 C 语言环境中更加顺利地使用 GU-7000 系列各种指令。

<http://www.noritake-elec.com/codeLibrary.php>

## 8 Environment

We are committed to the understanding of environmentally hazardous substances established by the Green Procurement Guideline.

We have also certification of ISO14000 and are keeping striving for the correspondence to an environmental issue.

### 8.1 RoHS Compliance

All standard module products including GU-7000 series are compatible with the RoHS Directive.

GU-7000 series is exempt from lead-containing component RoHS as follows:

- Lead oxide in glass used in Vacuum Fluorescent display.
- Lead oxide contained in the ceramic and glass in electronic components.
- Lead in high melting point solder components used to connect the internal semiconductor products.

## 9 Safety standard

Printed circuit board materials are certified UL :94-V0 flame retardant grade.

The UL certification number is indicated on the printed circuit board.

## 10 Disclaimers and limitations

The contents provided in this document are carefully created and managed by our Company, but we cannot guarantee to use these contents on all platforms. If you have any problems or questions, [please consult us](#).

Noritake sample codes are only for the use of Noritake products. Please carry out operating verification for final products such as application software on your responsibility. Support tools provided in a form of the installer may contain programs that are used under license. These must not be reproduced, modified, or integrated, etc. as reverse engineered, reverse compiled, or reverse assembled.

## 11 Contact us

If you have any questions or requests, please consult our sales office or customer support desk ([vfd@noritake.sh.cn](mailto:vfd@noritake.sh.cn))

End of document