

蛍光表示管モジュール

「GU-3000 シリーズ モジュール」

アプリケーションノート

APF300 R1.00

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。 GU-3000シリーズは、連続ドットのグラフィックタイプ蛍光表示管を使用したビットマップ画像とキャラクタ表示を可能とした蛍光表示管モジュールです。

マクロプログラムやデータを内蔵のFlash Romに記憶し自律動作を行うことも可能です。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。

May 13, 2011 Copy rights reserved.

目次

蛍光表示管モジュール.....	- 1 -
「GU-3000 シリーズ モジュール」	- 1 -
1 概要.....	- 6 -
1.1 文字大きさについて.....	- 6 -
2 蛍光表示管モジュールラインナップ.....	- 7 -
2.1 品名について.....	- 7 -
2.2 標準シリーズ系統図.....	- 8 -
3 対象品種 GU-7000 シリーズ (2011 年 2 月現在 予定を含む).....	- 9 -
4 ハードウェア	- 10 -
4.1 ブロック図	- 10 -
4.1.1 ブロック図 (GU-3xx0)	- 10 -
4.1.2 ブロック図 (GU-3901)	- 10 -
4.2 コネクタ	- 11 -
4.3 ホストインターフェイス	- 11 -
4.4 入出力部等価回路	- 12 -
4.4.1 シリアルインターフェイス入出力部等価回路	- 12 -
4.4.2 パラレルインターフェイス入力部等価回路例 (GU256X64D-3900B)	- 13 -
4.4.3 汎用ポート回路.....	- 14 -
4.4.4 汎用 IO ポート使用例 1	- 15 -
4.4.5 汎用 IO ポート使用例 2	- 15 -
4.5 ホストインターフェイス 接続例.....	- 16 -
4.5.1 パラレル接続回路例	- 16 -
4.5.2 RS-232C シリアル の PC 接続例.....	- 16 -
4.5.3 PC のシリアルポートについて	- 17 -
4.5.4 シリアルポート経由での多数接続について.....	- 17 -
4.5.5 LAN アダプタと RS232C 分岐ボード組み合わせ例	- 17 -
5 ソフトウェア	- 18 -
5.1 動作モード	- 18 -
5.2 ノーマルモードでのデータ書き込みプロトコル.....	- 19 -
5.2.6 ダイレクトモード.....	- 19 -
5.2.7 電文モード.....	- 19 -
5.3 マクロとプログラムマクロについて	- 19 -
5.4 初期設定と書き込みプロトコル	- 20 -
5.5 表示用メモリ (RAM)	- 20 -

5.6	搭載文字種	- 21 -
5.6.1	GU-31xx シリーズ搭載文字種	- 21 -
5.6.2	GU-39xx シリーズ搭載文字種	- 21 -
5.6.3	GU-39xxB シリーズ搭載文字種	- 21 -
5.7	フォントテーブル	- 22 -
5.7.1	ANK 文字 (1 バイト文字)	- 22 -
5.7.2	キャラクターコード指定 (ESC t n)	- 22 -
5.7.3	International 国際文字セット指定 (ESC R n)	- 23 -
5.7.4	JIS 漢字,ハングル,簡体字,繁体字 (2 バイト文字)	- 23 -
5.7.5	フォント例	- 25 -
5.8	コマンドテーブル	- 26 -
5.9	カーソル移動と表示モード	- 26 -
6	パラレルインターフェイスの場合のプログラム例	- 27 -
7	表示デバイスとしての使用例 (ホスト接続)	- 28 -
7.1	Windows PC を使った Microsoft 製 Visual Studio 2010 プログラム例	- 28 -
7.1.1	Visual C# 2010 を使ってシリアルポートに接続	- 28 -
7.2	プログラム例	- 30 -
7.2.1	画面のクリア	- 30 -
7.2.2	カーソル移動	- 31 -
7.2.3	拡大文字	- 31 -
7.2.4	テストプログラム	- 31 -
7.2.5	プロポーショナル英数字	- 32 -
7.2.6	漢字表示を設定する	- 33 -
7.2.7	漢字表示の使い方	- 33 -
7.2.8	グラフィックを表示する	- 34 -
7.2.9	グラフィック表示実行例	- 35 -
7.2.10	グラフィックをスクロールさせる	- 36 -
7.2.11	グラフィックのスクロール実行例	- 36 -
7.2.12	裏画面を表示する	- 37 -
7.2.13	文字をスクロール表示させる	- 37 -
7.2.14	文字スクロール実行例	- 38 -
7.2.15	ユーザーウィンドウで画面を区別してつかう。	- 38 -
7.2.16	ユーザーウィンドウ使用例	- 39 -
7.3	LAN アダプタの TCP/IP ソケットによる接続	- 40 -
8	グラフィック DMA モード	- 40 -
9	自律動作用途での使い方 (1)	- 40 -
9.1	アクション付の表示を繰り返す。(名称: ME-021 itron 楽画記帳)	- 41 -
9.2	汎用 IO ポートの状態に対応して表示内容を変える (名称 PortLinkerDx)	- 41 -
9.3	シリアル及びパラレルで書き込まれたデータに対応した表示を行う。	- 41 -

9.4	TCP/IP ポートよりのデータに対応した表示を行う (名称 ME-141:TcpPortLinkerDX)	- 42
9.5	自分でマクロプログラムを作成する。(名称 ME-023 : Macro 操師)	- 42 -
9.6	RSS リーダー (名称 ME-136)	- 42 -
10	いろいろな操作 (プチツール)	- 42 -
10.1	使用方法 ほか。	- 42 -
10.2	利用条件	- 42 -
10.3	著作権	- 43 -
10.4	プチツール一覧	- 43 -
10.4.1	プチ・マクロ起動ツール : 「M 起動隊」 ME-030	- 43 -
10.4.2	プチ・スイッチャ ツール : 「スイッチ賛」 ME-031	- 43 -
10.4.3	プチ・メモリ検視ツール : 「MR I」 ME-032	- 43 -
10.4.4	プチ・バージョンチェッカー : 「管定士」 ME-033	- 43 -
10.4.5	プチ・クリーナ : 「クリンちゃん」 ME-034	- 43 -
10.4.6	I/O ポート確認ツール : 「お出入 GU」 ME-035	- 44 -
10.4.7	楽画記帳停止ツール : 「画記封じ」 ME-104	- 44 -
10.4.8	表示画面キャプチャツール : 「GuCapture」 ME-111	- 44 -
11	トラブルシューティング	- 45 -
11.1	READY 信号について	- 45 -
11.2	リセットについて	- 45 -
11.3	プログラムマクロが終了できないときの回復方法	- 45 -
11.4	まったく点灯しないとき---セルフテストモード	- 46 -
11.5	セルフテストモー設定方法	- 47 -
11.5.1	GU128X128D-3900B	- 47 -
11.5.2	GU256X128C-3900B	- 47 -
11.5.3	GU256X128D-3900B	- 47 -
11.5.4	GU256X128E-3100、GU256X128E-3900	- 48 -
11.5.5	GU256X128E-3910	- 48 -
11.5.6	GU256X16M-3900、GU256X16M-3900B	- 48 -
11.5.7	GU256X32D-3100、GU256X32D-3900、GU256X32D-3900B	- 48 -
11.5.8	GU256X32L-3900	- 49 -
11.5.9	GU256X64C-3100、GU256X64C-3900、GU256X64C-3900B	- 49 -
11.5.10	GU256X64D-3100、GU256X64D-3900、GU256X64D-3900B	- 49 -
11.5.11	GU256X64D-3101	- 49 -
11.5.12	GU256X64E-3100、GU256X64E-3900、GU256X64E-3900B	- 49 -
11.5.13	GU256X64E-3101	- 50 -
11.5.14	GU256X64F-3100、GU256X64F-3900	- 50 -
11.5.15	GU256X64F-3101	- 50 -
11.5.16	GU320X32D-3900	- 50 -

11.5.17	GU384X32L-3100、GU384X32L-3900、GU384X32L-3900B.....	- 51 -
11.5.18	GU384X32L-3950B.....	- 51 -
11.5.19	GU512X32H-3100、GU512X32H-3900、GU512X32H-3900B.....	- 51 -
11.5.20	GU512X32H-3940、GU512X32H-3940B.....	- 51 -
12	支援 TOOL	- 52 -
12.1	UNIDEMO (ユニデモ)	- 52 -
12.2	COMTEST (コムテスト)	- 52 -
13	環境対応.....	- 53 -
13.1	RoHS 指令対応	- 53 -
14	安全規格.....	- 53 -
15	免責と制限事項について	- 53 -
16	問い合わせ先 :	- 53 -

1 概要

GU-3000 シリーズ蛍光表示管モジュールは、高い視認性と信頼性を有する蛍光表示管に電源回路、制御用 CPU、Flash メモリを搭載した表示サブシステムです。CPU 搭載によりホスト制御器からコマンドを送ることによって表示を制御することができます。JIS や中国の漢字、ハングル文字フォントを搭載品しています。

Flash ROM には、画像データやマクロプログラムを記憶でき、I/O ポートと組み合わせてスタンダアローン動作を可能にしています。

PHOTO.1 GU256X64D-3900



1.1 文字大きさについて

蛍光表示管は蛍光体を使用した自発光型のディスプレイです。このため反射/透過光を使用する LCD と比べて視認性がたかくなります。同じ文字寸法であればより遠くから読むことができ、同じ視認距離を得るための文字寸法は小さくて済みます。寸法を検討の際には実機での確認をお願いします。

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：GU256X64D-3900B

GU	↓		: グラフィック表示可能
256X64	↓		: 256 x 64 画素
D	↓		: 画素の高さ 0.4~0.49mm/画素
			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
		-3900B	: 3900B シリーズ

インターフェイス等仕様を表します。

31xx= JIS 漢字のみを搭載した品種です。新規採用には
-3900B シリーズをご検討ください。

39xx= (JIS 漢字、中国漢字、ハングル対応)

39x0=RS232C とパラレルインターフェイス

39x1= USB インターフェイス

39x2=C-MOS シリアルとパラレルインターフェイス

39xxB=仕様書上位互換の世代交代版です。

同じシリーズでは、コマンドや主な仕様が共通になっています。上記テーブルは一般的な設定です。各品種の仕様につきましては各々の製品仕様書でご確認ください。

2.2 標準シリーズ系統図

GU-3000 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
- ┆ 標準 GU シリーズ
 - | グラフィック可能なシリーズ
 - ┆ GU-300 シリーズ、GU9300 シリーズ
 - |
 - ┆ GU-800 シリーズ
 - | ┆ GU-800, GU-800B
 - | | グラフィック専用
 - | ┆ GU-8000, GU-8000B
 - | 漢字表示機能追加
 - ┆ GU-7000 シリーズ 外形が LCD 互換。
 - | ┆ GU-7000
 - | | 小型。グラフィック可能な文字表示モジュール。
 - | ┆ GU-7900
 - | 小型。7000 の漢字表示追加品。
 - ┆ GU-3000 シリーズ
 - ┆ **GU-3100**
 - | グラフィック可能な文字表示モジュール。
 - | 24Dot 日本語漢字フォント搭載。
 - | FROM 搭載でマクロ機能を搭載しスタンドアローン動作が可能。
 - ┆ **GU-3900**
 - | グラフィック可能な文字表示モジュール。
 - | JIS,繁体、簡体漢字、ハングルフォント搭載。
 - | FROM 搭載でマクロ機能を搭載しスタンドアローン動作が可能。
 - ┆ **GU-3900B**
 - | グラフィック可能な文字表示モジュール。
 - | 3100&3900 の機能上位互換品の後継品種。

3 対象品種 GU-7000 シリーズ (2011 年 2 月現在 予定を含む)

このアプリケーションノートの対象品種は以下の通りです。

表に無い標準品種につきましても基本的な機能は共通です。

GU128X128D-3900B

GU256X128C-3900B

GU256X128D-3900B

GU256X128E-3100、 GU256X128E-3900、 GU256X128E-3910

GU256X16M-3900、 GU256X16M-3900B

GU256X32D-3100、 GU256X32D-3900、 GU256X32D-3900B

GU256X32L-3900

GU256X64C-3100、 GU256X64C-3900

GU256X64D-3100、 GU256X64D-3101、 GU256X64D-3900、 GU256X64D-3900B

GU256X64E-3100、 GU256X64E-3101、 GU256X64E-3900、 GU256X64E-3900B

GU256X64F-3100、 GU256X64F-3101、 GU256X64F-3900

GU320X32D-3900

GU384X32L-3100、 GU384X32L-3900、 GU384X32L-3900B、 GU384X32L-3950B

GU512X32H-3100、 GU512X32H-3900、 GU512X32H-3900B

GU512X32H-3940、 GU512X32H-3940B

3100 と 3900 は順次 3900B に置き換えとさせていただきます。新プロジェクトには 3900B の採用を検討ください。

最新の品種 list につきましては、[インターネットの弊社サイト](http://www.noritake-itron.jp/)をご覧ください。営業担当者または最寄りの営業所までお問い合わせください。

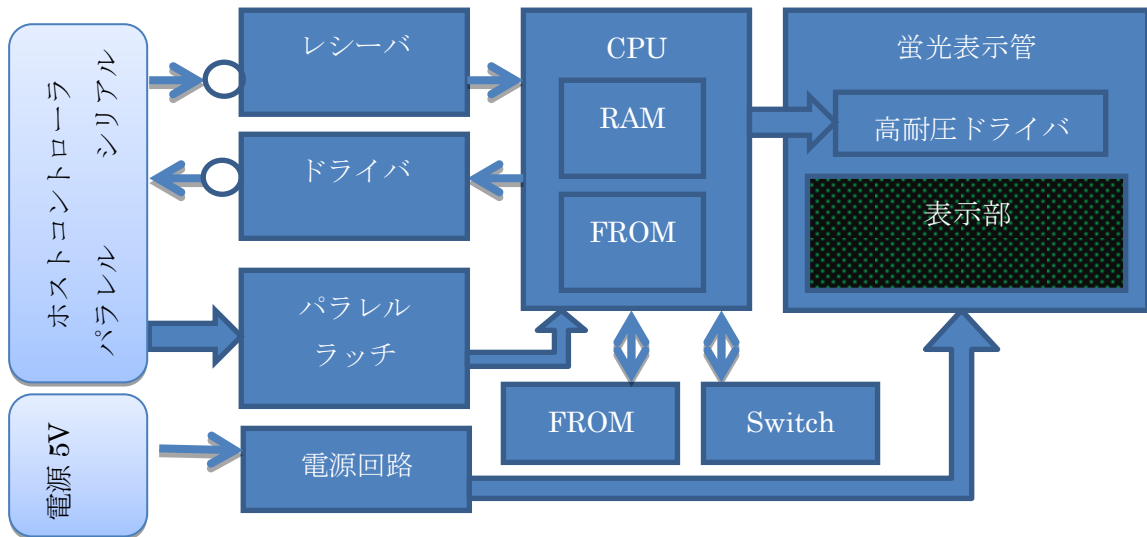
<http://www.noritake-itron.jp/>

4 ハードウェア

4.1 ブロック図

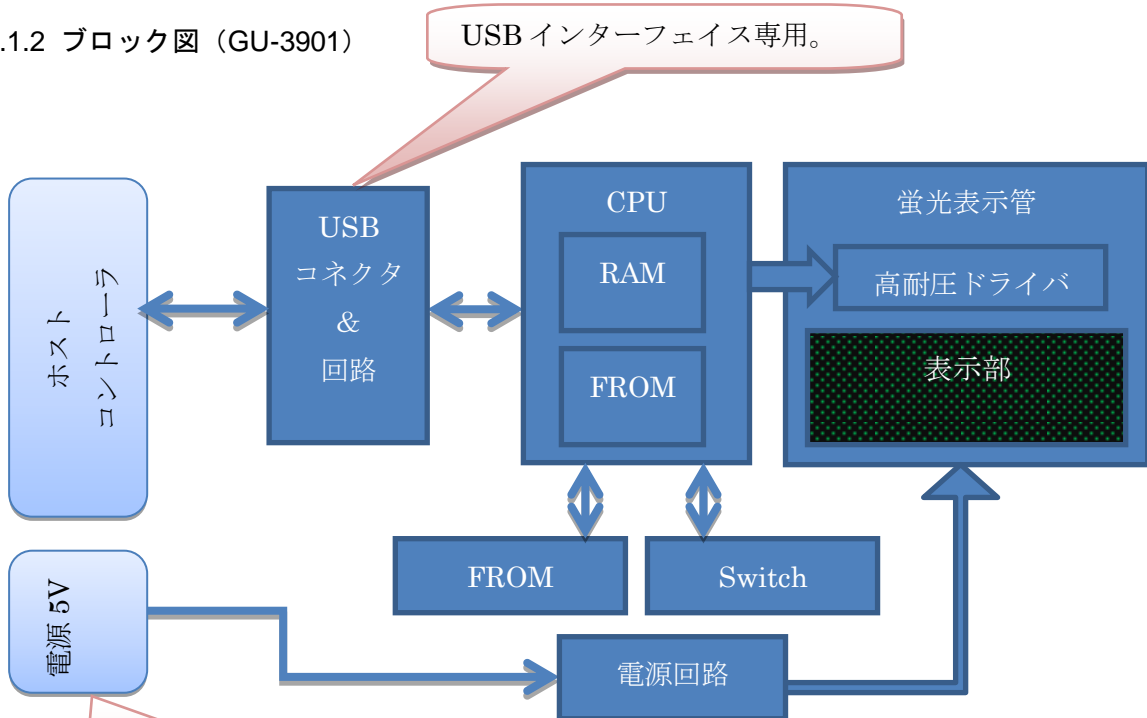
モジュールは、入出力部、CPU（制御回路）、電源回路、蛍光表示管から構成されています。

4.1.1 ブロック図（GU-3xx0）



5V 以外の電源の品種もあります。仕様書でご確認下さい。

4.1.2 ブロック図（GU-3901）

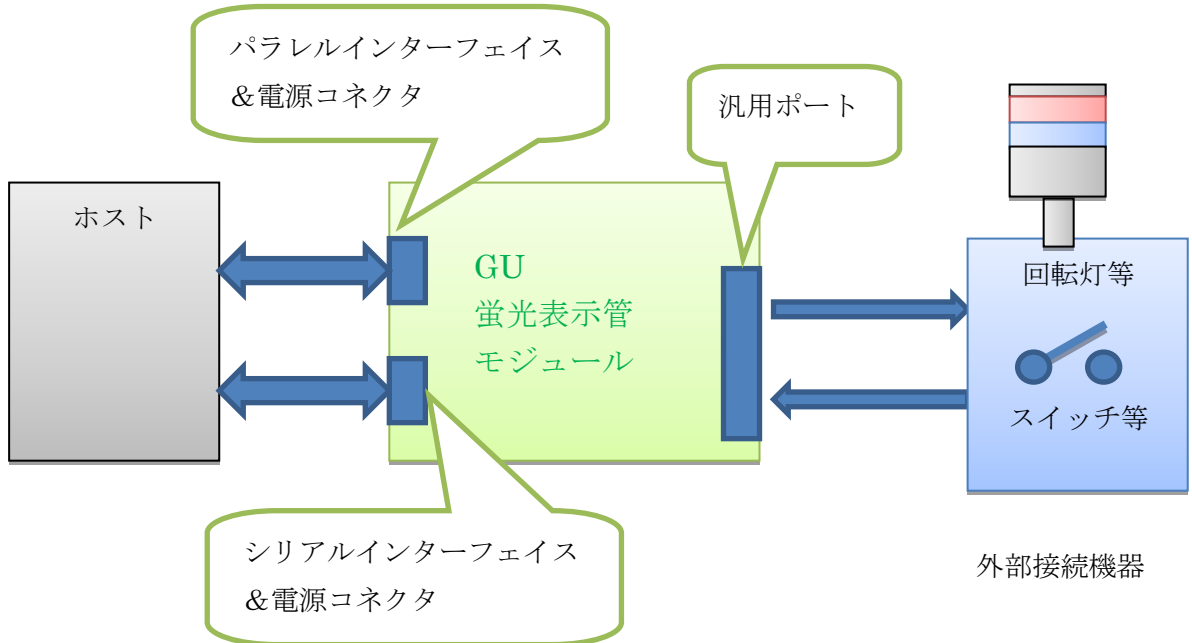


表示管用電源は独立コネクタより供給となります。

5V 以外の電源の品種もあります。仕様書でご確認下さい。

4.2 コネクタ

表示器には3個のコネクタが搭載されています。



	表示器搭載コネクタ	適合コネクタ	備考
パラレルインターフェイス&電源コネクタ	IRISO:IMSA-9032B-16P または同等品	2.54mm ピッチフラットケーブルコネクタ	
シリアルインターフェイス&電源コネクタ	JST:B7B-XH-A または同等品	JST 社製適合品 JQ シリーズ NR シリーズ NRD シリーズ XH シリーズ	2.5mm ピッチ 7ピンコネクタ。 詳しくはJST社にお問い合わせください。
汎用ポート	直径 1mm スルーホール	位置と並びは各品種のハードウェア仕様書にてご確認ください。	

パラレルインターフェイスとシリアルインターフェイスはいずれか一方で使用可能です。

GU 蛍光表示管モジュールの基板には外部接続機器を実装できるフリーエリアはありません。

4.3 ホストインターフェイス

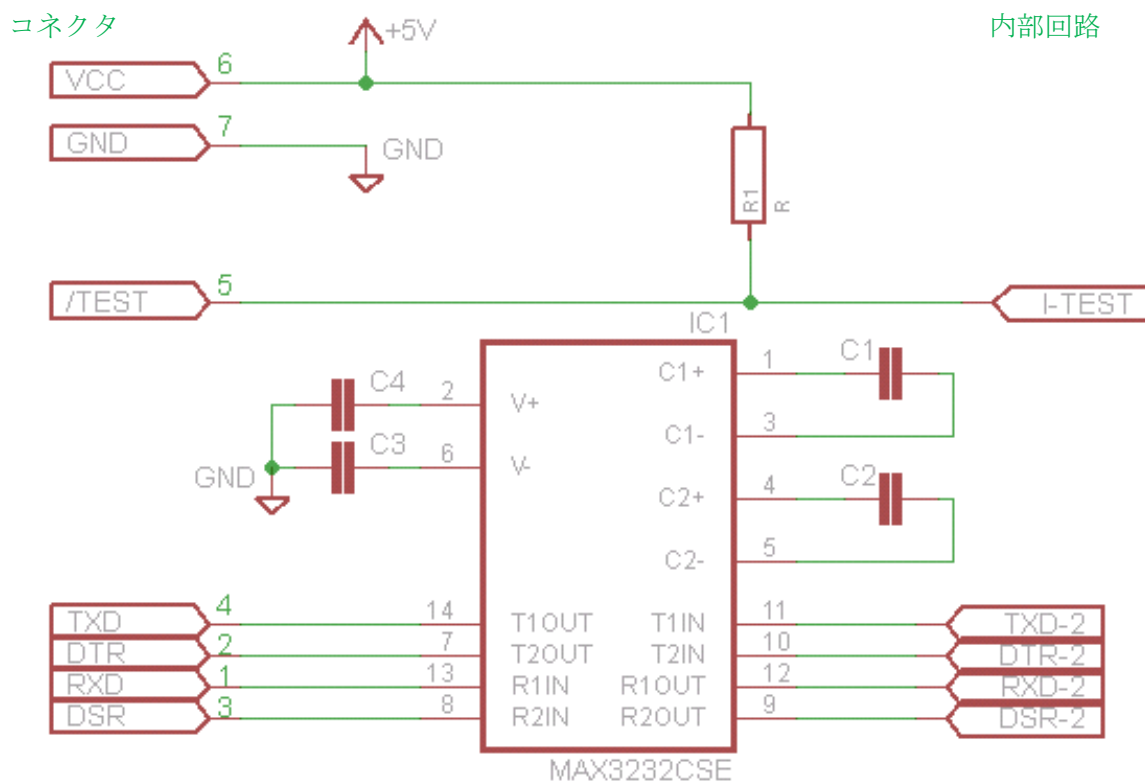
品名の末尾の1桁でインターフェイス仕様を示します。

0 : RS-232C レベル非同期シリアル入力 + 8ビットパラレル入力

1 : USB

4.4 入出力部等価回路

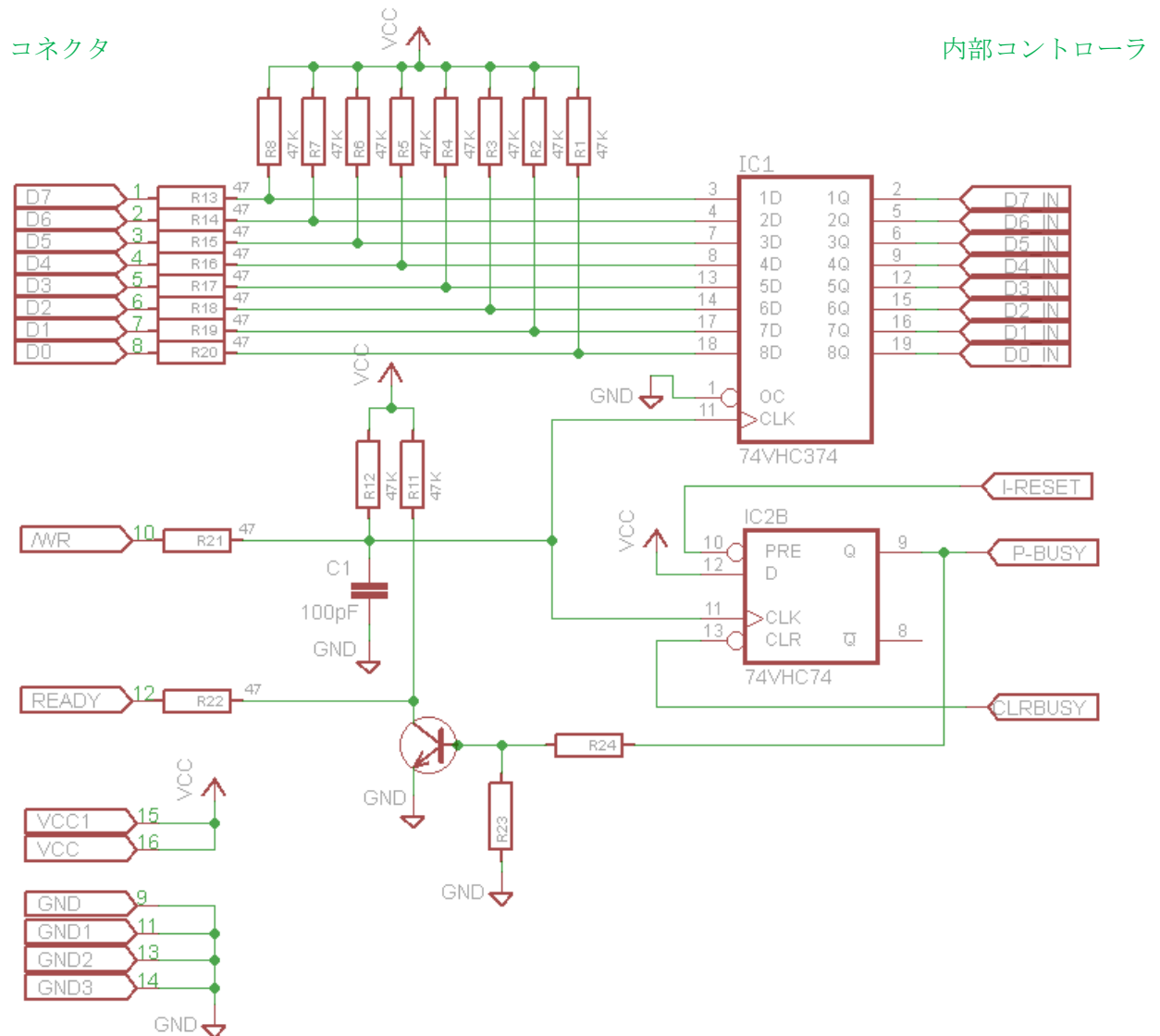
4.4.1 シリアルインターフェイス入出力部等価回路



入出力をご理解頂くための等価回路であり使用部品をふくめて製品と同じであることを保証するものではありません。部品番号は実機とは異なります。

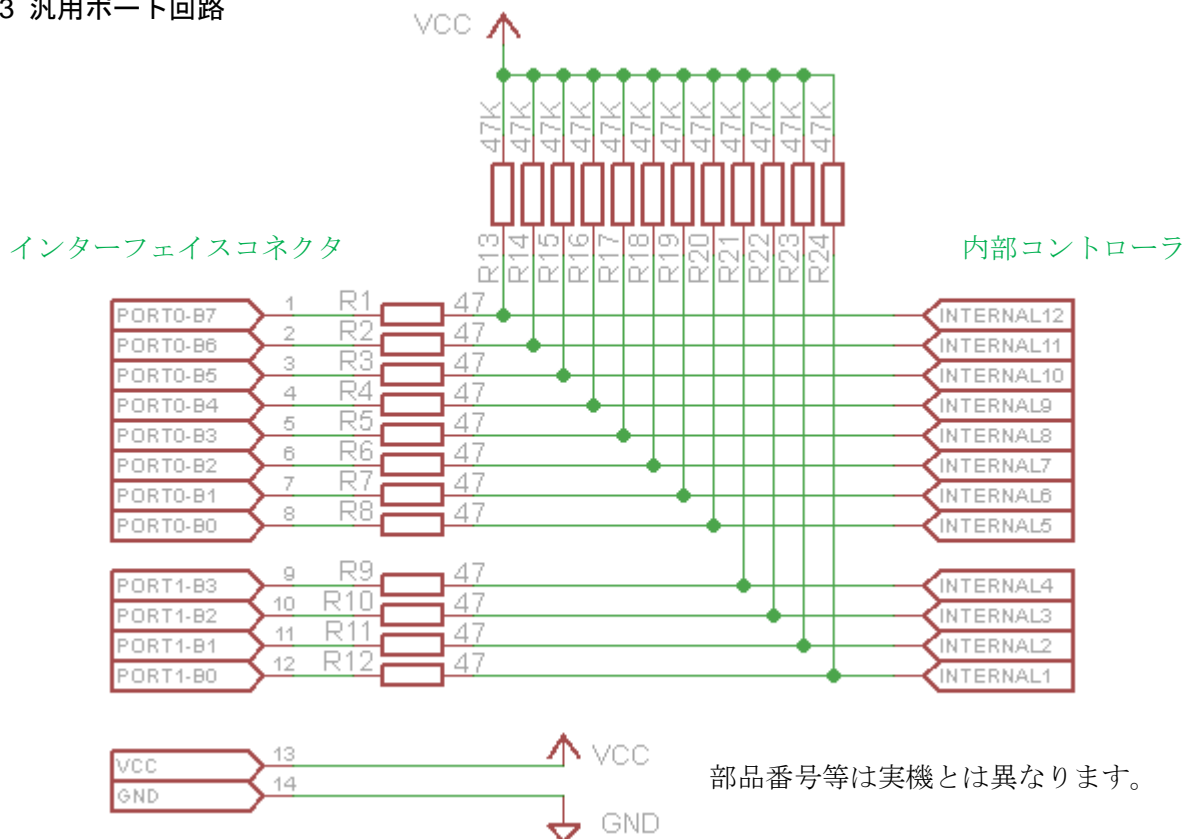
4.4.2 パラレルインターフェイス入力部等価回路例 (GU256X64D-3900B)

パラレルインターフェイスは、書き込みのみ可能です。データ読み出しがありませんので、データ読み出しを必要とするコマンドは扱えませんのでご注意ください。



入出力をご理解頂くための等価回路であり使用部品をふくめて製品と同じであることを保証するものではありません。部品番号は実機とは異なります。

4.4.3 汎用ポート回路



汎用ポートは、47Ω抵抗を介して内部のコントローラに直結されています。配線を延長する場合はバッファの追加や、フォトアイソレータによる絶縁分離などを検討してください。

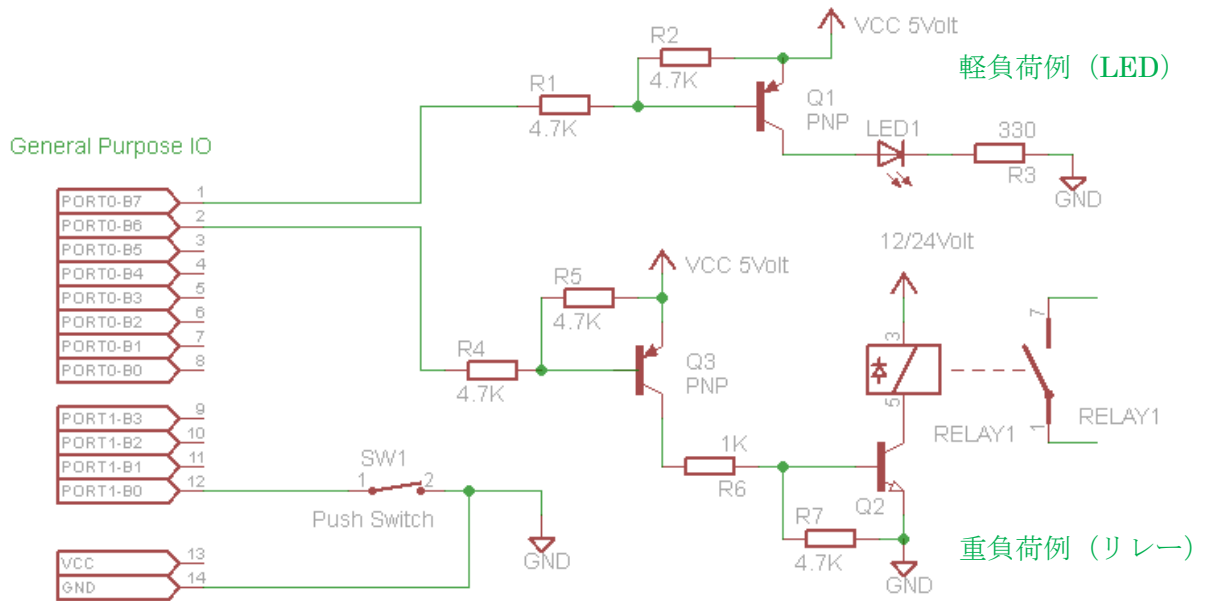
汎用 IO ポートは、リセット直後は入力となっています。従ってプルアップ抵抗により端子の電位は (VCC=5V) になります。このため、端子電位が 5V の時接続された周辺機器が動作停止となるような接続を推奨いたします。

5V で機器が動作する設計の場合、電源投入後初期化完了までの間機器が動作してしまいます。

次に掲げる例では、0V を出力するとリレーが ON し LOAD に電圧が加わる論理設計となっています。

4.4.4 汎用 IO ポート使用例 1

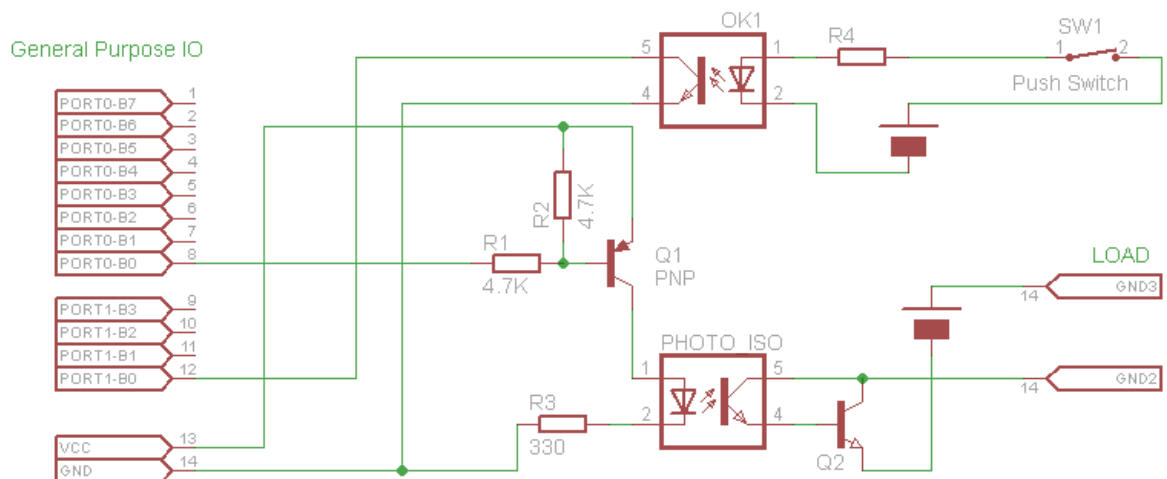
小型機器に使用し筐体内で配線が短く完了しノイズ源がない場合の使用例です。



汎用 IO ポートにスパイク状のノイズ等が侵入すると表示器の永久破壊の原因になりますので十分に検証のうえお使いください。

4.4.5 汎用 IO ポート使用例 2

配線長を長くする場合やノイズの多い環境で使用する場合は、フォトアイソレータによる絶縁を行ってください。



4.5 ホストインターフェイス 接続例

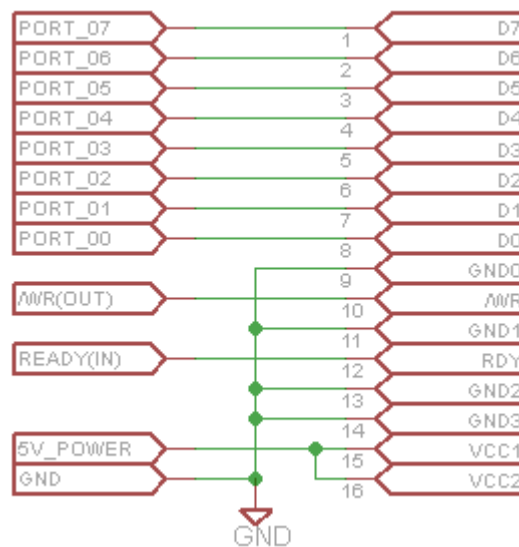
スクロール、時間待ちコマンドなどを有効に使用するために **READY** 信号を使用したハードウェアハンドシェイクを行ってください。

4.5.1 パラレル接続回路例

パラレルインターフェイスは、書き込みのみ可能です。データ読み出しがありませんので、データ読み出しを必要とするコマンドは扱えませんのでご注意ください。

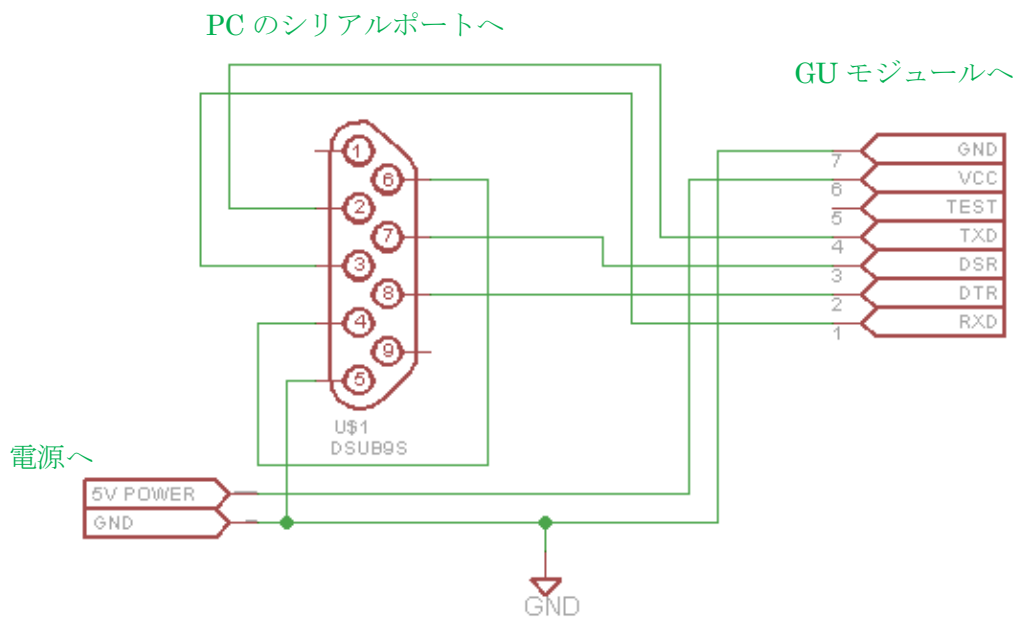
ホスト回路

ディスプレイコネクタ



4.5.2 RS-232C シリアル の PC 接続例

シリアル入力はソフトウェア **FIFO** バッファを持っておりますがオーバーランエラーのリスクに対応するためのものです。書き込み時間を最小にするためには、1 バイト毎のハンドシェイクによる書き込みを行ってください。(仕様書記載の方法です。)



4.5.3 PC のシリアルポートについて

シリアルポートを持たない PC も多くなっています。このような場合は、市販の USB シリアルアダプタがご使用いただけます。動作環境等につきましては、アダプタの販売元に問い合わせさせて頂くようお願いいたします。

また Ethernet 経由で接続する LAN アダプタもご用意させて頂いております。



LAN アダプタキット
SCK-BCXPST01-A

左の写真は、Lantronix 社製の XPORT アダプタを搭載した弊社製の Ethernet-RS232C 接続アダプタです。ご利用頂ければ幸いです。(2011 年 2 月現在)

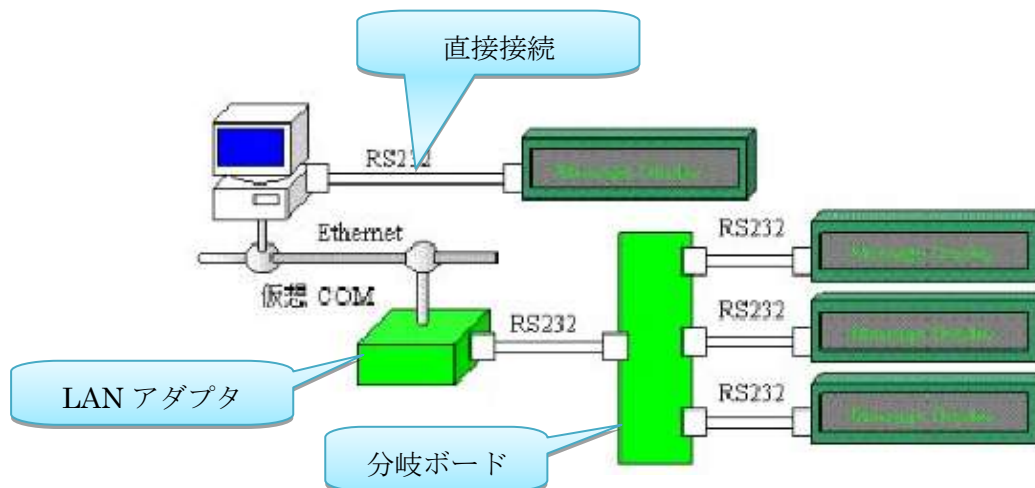
4.5.4 シリアルポート経由での多数接続について

複数台の表示器を PC につなぐ場合に便利な RS232C 信号の分岐ボードもご用意させて頂いております。本ボード 2 枚を使うと、ひとつのシリアルポートに 16 台の表示器を接続できます。電文モードを使用して表示器にアドレスを設定することで分岐ボードに接続された表示器を個別に制御します。



RS232C 分岐ボード SCP-BD09ST01

4.5.5 LAN アダプタと RS232C 分岐ボード組み合わせ例



5 ソフトウェア

5.1 動作モード

ノーマルコマンドモード

通常使用時の動作状態です。ホストから受け取ったコマンドやデータを処理し表示したり、マクロとプログラムマクロを実行したりします。

グラフィック DMA モード

通常使用時の動作状態です。パラレルインターフェイスを使いコマンドを扱わないことによりグラフィックデータをより高速に扱います。

ユーザー設定モード

Flash ROM 上のメモリスイッチや各種データの保存と行うためのモードです。メモリスイッチによりリセット直後の設定をデフォルトから変更できます。

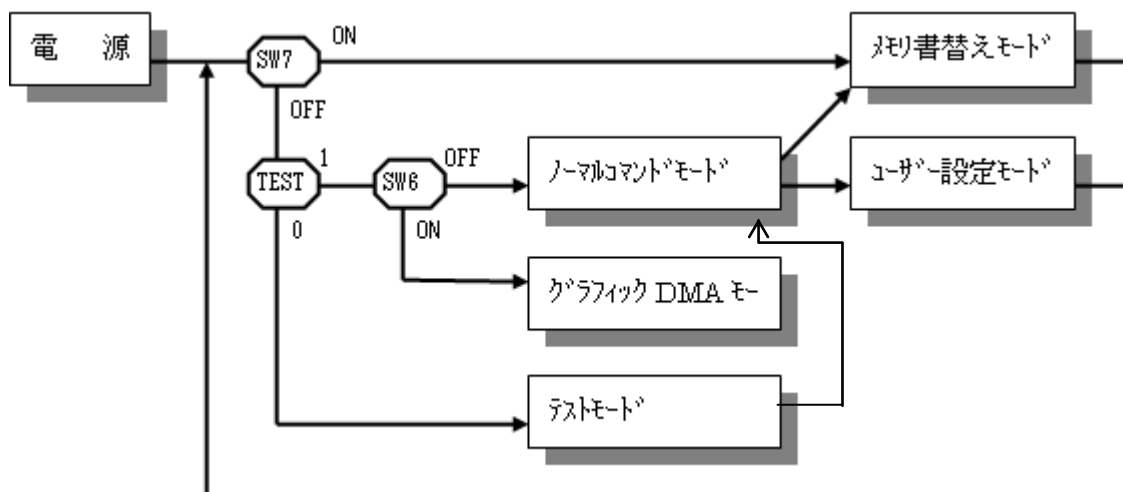
メモリ書替えモード

ファームウェア、フォントデータなどを書き換えるためのモードです。電源 OFF などによる中断でファームウェアが書き換わりますと工場ではしか修復できなくなりますので、機器組み込み後の通常運用中の使用は避けてください。

ファームウェア書き換えコマンドは弊社工場専用のコマンドです。

テストモード

外部インターフェースや Flash ROM 設定によらず、テストパターンを表示します。プログラムマクロを終了できなくなった場合に、テストモードを経由することでノーマルモードに移行することができます。(方法は [11.5 参照](#))



表示器の起動シーケンスとモード遷移

5.2 ノーマルモードでのデータ書き込みプロトコル

ノーマルモードでは、シリアルまたはパラレルポートを使ってコマンドとデータの通信を行います。2種類の通信プロトコルをサポートしています。

5.2.6 ダイレクトモード

コマンドとデータをそのまま書き込みます。表示器は受け取ったデータをそのまま処理します。

5.2.7 電文モード

アドレス情報を付加したパケット化して通信します。

USB 接続モデルには電文モードはありません。

表示器は、パケットのアドレスと DIP スイッチで設定された自分のアドレスの一致したパケットのみを処理対象とします。アドレス範囲は次の通りです。

- GU-31xx, GU-39xx の場合：

アドレス設定用スイッチは4ビットで0~15の16台までを個別に制御できます。
パケットのアドレス=FFhとすると、すべての表示器が対応します。

- GU-3900B の場合：

アドレスは8ビットで、FFHは同報用アドレスですべての表示器が対応します。従い最大255台まで接続が可能です。正常にパケットを受信した表示器はACKを返送します。アドレス=FFhのパケットに対しては、アドレス=00hの表示器のみがACKを送り返します。

BCCエラーを検出した場合、パケットは破棄されレスポンスは返されません。

5.3 マクロとプログラムマクロについて

表示を行うための一連のコマンドや文字コードをRAMやFlash ROMに書き込んでおき、それを繰り返し実行・表示させることができます。マクロには次の2種類があります。

- マクロ (または 通常マクロ)

エスケープシーケンスなどのコマンドや文字コードの組み合わせを順番に並べたマクロです。リセットコマンドなど一部の例外を除きソフトウェア仕様書記載のコマンドを使用できます。最後まで表示が終わると最初に戻って繰り返します。

- プログラムマクロ

オリジナルのスクリプト言語によるマクロです。汎用IOやシリアルポートでの入出力や条件分岐、ループなどが使用可能となります。「マクロの操師」というコンパイラを用意しております。

Flash ROM に書かれたマクロとプログラムマクロは電源投入時に自動起動させることができますので、スタンドアローン表示器としてご利用いただけます。

通常マクロは、マクロ実行中に外部からデータが書き込まれると中断し入力されたデータを処理します。待ち時間表示などには RAM にマクロを書き込んで実行することで、ホストの負担なしに表示を変え続けることができます。

プログラムマクロの注意点

プログラムマクロは、外部からのデータ書き込みで自動的には中断しません。定期的に外部データを確認し終了するようにプログラムしておいて頂く必要があります。これがないと、別のマクロに書き換えるなども出来なくなります。プログラムマクロが終了できなくなった場合の強制終了方法は、[トラブルシューティング](#)をご覧ください。

5.4 初期設定と書き込みプロトコル

デフォルト設定による内部初期化機能により特別な初期化無しで基本的な表示機能を使用することができます。電源投入後は初期化が終了すると **READY** がアクティブになりますので、単に表示データを書き込んでください。データは、ASCII を基本とした一般的なコード体系となっています。各種機能を使用するためのコマンドは ESC コード他で始まる拡張シーケンスを使用します。書き込みの際は、ハードウェアハンドシェイクによる書き込み制御を行ってください。メモリスイッチの変更により初期化後の表示器の各種設定をデフォルトとは異なった状態にすることができます。

5.5 表示用メモリ (RAM)

書き込まれたデータは、随時処理されてグラフィックイメージに内部変換され表示用 RAM に保存されます。RAM の容量は表示画面のドット数より大きいため、RAM データの一部が表示されることとなります。つまり、RAM は表示されるエリアと非表示エリアに分かれることとなります。

表示用 RAM



非表示エリアは、描画途中のデータをユーザに見せないための作業領域に使用したり、スクロールで表示させる画像を準備する領域に使用できます。

5.6 搭載文字種

文字コードとして、1バイトの英数文字と2バイトの漢字フォントを扱うことができます。
搭載フォントは次の通りです。

5.6.1 GU-31xx シリーズ搭載文字種

文字コード長	フォント寸法	フォントデータ
1バイト	6x8 ドット	インターナショナルフォント
	8x16 ドット	
	16x32 ドット	
2バイト	16x16 ドット	日本語
	32x32 ドット	日本語

5.6.2 GU-39xx シリーズ搭載文字種

文字コード長	フォント寸法	フォントデータ
1バイト	6x8 ドット	インターナショナルフォント
	8x16 ドット	
2バイト	16x16 ドット	日本語
		中国簡体語
		中国繁体語
		韓国語

5.6.3 GU-39xxB シリーズ搭載文字種

文字コード長	フォント寸法	フォントデータ
1バイト	6x8 ドット	インターナショナルフォント (DS-1600-0004-XX 参照)
	8x16 ドット	
	12x24 ドット	
	16x32 ドット	
2バイト	16x16 ドット	日本語 (DS-906-0002-XX 参照)
		中国簡体語 (DS-954-0006-XX 参照)
2バイト	16x16 ドット	中国繁体語 (DS-954-0007-XX 参照)
		韓国語 (DS-954-0008-XX 参照)
2バイト	32x32 ドット	日本語 (DS-906-0003-XX 参照)

5.7 フォントテーブル

概念のみを示します。具体的なフォントデータは仕様書を参照してください。

5.7.1 ANK 文字 (1 バイト文字)

	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	フ ン ト コ ド	共通フォント領域 ASCII 準拠							拡張フォント領域 ESC t n で選択							
_1H																
_2H																
_3H																
_4H																
_5H																
_6H																
_7H																
_8H																
_9H																
_AH																
_BH																
_CH																
_DH																
_EH																
_FH																

1 バイト文字は 3 つの領域に分割されています。

00Hex~1FHex: 機能が割り当てられています。例えば 0DHex(CR)を書き込むとカーソル位置が画面の左端に移動します。

20Hex~7FHex: ASCII 準拠の英数字フォント領域です。

80Hex~FFHex: 拡張フォント領域で、ESC t n コマンドを書き込むことで複数のセットから選択できます。選択変更は書き込み済みの文字には影響しません。

5.7.2 キャラクターコード指定 (ESC t n)

n	文字種
0	PC437(USA: Standard Europe)
1	カタカナ
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
255 (FF h)	ユーザーテーブル

255(FFhex)は、FROM にユーザーが登録したフォントテーブルを指定します。

5.7.3 International 国際文字セット指定 (ESC R n)

国際文字セット指定は、基本フォント領域（20Hex~7FHex）の文字を一部変更する指定です。
例えば、日本(08H)を指定すると、“ ”（文字コード 5CHex）が円マーク “¥” に変わります。

ESC R n コマンドで指定します。

n	言語	00H	01H	02H	03H	04H	05H	06H	07H	08H	09H	0AH	0BH	0CH	0DH
0	USA														
1	フランス	23H	##	##	##	##	##	##	##	##	##	##	##	##	##
2	ドイツ	24H	\$\$	\$\$	\$\$	\$\$	\$\$	\$\$	\$\$	\$\$	\$\$	\$\$	\$\$	\$\$	\$\$
3	イギリス	40H	@@	@@	@@	@@	@@	@@	@@	@@	@@	@@	@@	@@	@@
4	デンマーク I	5BH	[°	A	[A	°	i	[A	A	i	i	[
5	スウェーデン	5CH	\	ö	\	ö	\	ö	\	ö	ö	ö	ö	ö	ö
6	イタリア	5DH]S	O]A	A	e	¿]A	A	¿	¿	¿	¿]S
7	スペイン	5EH	^	^	^	^	^	^	^	^	^	^	^	^	^
8	日本	60H	^	^	^	^	^	^	^	^	^	^	^	^	^
9	ノルウェー	60H	^	^	^	^	^	^	^	^	^	^	^	^	^
10(0AH)	デンマーク II	7BH	C	e	a	C	a	a	a	~	C	a	e	i	i
11(0BH)	スペイン II	7CH		ö		ö	ö	ö	ö	ö	ö	ö	ö	ö	ö
12(0CH)	ラテンアメリカ	7DH	>	e	ö	>	a	a	e	>	a	a	ö	ö	>
13(0DH)	韓国	7EH	~	ö	~	ö	ö	ö	ö	ö	ö	ö	ö	ö	ö

指定の変更は書き込み済みの文字には影響しません。

5.7.4 JIS 漢字,ハングル,簡体字,繁体字（2バイト文字）

2バイト文字で漢字を表示させるには方法を説明します。

次のコマンドを書き込んでから文字コードを書き込んでください。設定を変えても書き込み済みの文字には影響しませんので、設定を順次変えながら書き込みことで複数のフォントを同時表示することができます。

2バイト文字表示手順（文字タイプ選択以外は同じです。）

日本語表示（16x16 ドット）の設定例

- 1FH, 28H, 67H, 01H, 02H ‘ 8x16 フォントサイズ選択
- 1FH, 28H, 67H, 02H, 01H ‘ 2バイト文字モード指定
- 1FH, 28H, 67H, 03H, 00H ‘ 日本語
- 88H, A2H ‘ “阿” を表示

日本語表示（32x32 ドット）の設定例

1FH, 28H, 67H, 01H, 04H	‘ 16x32 フォントサイズ選択
1FH, 28H, 67H, 02H, 01H	‘ 2 バイト文字モード指定
1FH, 28H, 67H, 03H, 00H	‘ 日本語
88H, A2H	‘ “阿” を表示

韓国語表示（16x16 ドット）の設定例

1FH, 28H, 67H, 01H, 02H	‘ 8x16 フォントサイズ選択
1FH, 28H, 67H, 02H, 01H	‘ 2 バイト文字モード指定
1FH, 28H, 67H, 03H, 01H	‘ ハングル

文字コード書き込み

中国簡体字表示（16x16 ドット）の設定例

1FH, 28H, 67H, 01H, 02H	‘ 8x16 フォントサイズ選択
1FH, 28H, 67H, 02H, 01H	‘ 2 バイト文字モード指定
1FH, 28H, 67H, 03H, 02H	‘ 簡体字

文字コード書き込み

中国繁体字表示（16x16 ドット）の設定例

1FH, 28H, 67H, 01H, 02H	‘ 8x16 フォントサイズ選択
1FH, 28H, 67H, 02H, 01H	‘ 2 バイト文字モード指定
1FH, 28H, 67H, 03H, 03H	‘ 繁体字

文字コード書き込み

32x32 ドットフォントは、JIK 漢字のみで使用可能です。他のフォントで 32x32 ドット表示を行う場合は、16x16 ドット文字を縦横各倍に拡大して使用してください。

各フォントの準拠規格とコード範囲は次の通りです。

フォント	準拠規格	2 バイトコード範囲
JIS 漢字	JISX208 (Shift-JIS)	8140H~9FF0H, E040~EFFCH
ハングル	KSX5601-87	A1A1H~FEFEH
簡体字	GB2312-80	A1A1H~FEFEH
繁体字	Big-5	A140H~FEFEH

5.7.5 フォント例

JIS 16x16 ドット

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	P
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	o
829x	p	q	r	s	t	u	v	w	x	y	z					あ
82Ax	あ	い	い	う	う	え	え	お	お	か	が	き	ぎ	く	く	け
82Bx	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た	だ	ち
82Cx	ち	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は	ば	ば
88Ax	啞	娃	阿	哀	愛	挨	始	逢	葵	茜	種	悪	握	渥	旭	葦
88Bx	芦	鱒	梓	庄	幹	扱	宛	姐	虻	飴	絢	綾	鮎	或	粟	裕
88Cx	安	庵	按	暗	案	闇	鞍	杏	以	伊	位	依	偉	困	夷	委
88Dx	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃	萎	衣
88Ex	謂	違	遺	医	井	亥	域	育	郁	磯	一	志	溢	逸	稻	茨
88Fx	芋	鱒	允	印	咽	員	因	姻	引	飲						

JIS 32x32 ドット

825x	1	2	3	4	5	6	7	8	9						
826x	A	B	C	D	E	F	G	H	I	J					
827x	Q	R	S	T	U	V	W	X	Y	Z					
828x		a	b	c	d	e	f	g	h	i					
829x	p	q	r	s	t	u	v	w	x	y					
82Ax	あ	い	い	う	う	え	え	お	お	か					
82Bx	げ	こ	ご	さ	ざ	し	じ	す	ず	せ					
82Cx	ち	っ	つ	づ	て	で	と	ど	な	に					
88Ax	啞	娃	阿	哀	愛	挨	始	逢	葵	茜					
88Bx	芦	鱒	梓	庄	幹	扱	宛	姐	虻	飴					
88Cx	安	庵	按	暗	案	闇	鞍	杏	以	伊					
88Dx	威	尉	惟	意	慰	易	椅	為	畏	異					
88Ex	謂	違	遺	医	井	亥	域	育	郁	磯					
88Fx	芋	鱒	允	印	咽	員	因	姻	引	飲					

ハングルフォント例 16x16 ドット

B0Ax		가	각	간	갈	갈	갈	갈	갈	갈	갈	갈	갈	갈	갈	갈
B0Bx	갈	갈	갈	개	객	객	객	객	객	객	객	객	객	객	객	객
B0Cx	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓
B0Dx	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓
B0Ex	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓	갓
B0Fx	곤	곤	곤	곤	곤	곤	곤	곤	곤	곤	곤	곤	곤	곤	곤	곤
B1Ax		관	관	관	관	관	관	관	관	관	관	관	관	관	관	관
B1Bx	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관
B1Cx	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관
B1Dx	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관
B1Ex	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관
B1Fx	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관	관

簡体字フォント例 16x16 ドット

B0Ax		啊	阿	埃	挨	哎	唉	哀	皑	癌	蔼	矮	艾	碍	爰	隘
B0Bx	鞍	氨	安	俺	按	暗	岸	胺	肮	昂	盎	凹	敖	熬	翱	
B0Cx	袄	傲	奥	懊	澳	芭	捌	扒	叭	吧	芭	八	疤	巴	拔	跋
B0Dx	靶	把	耙	坝	霸	罢	爸	白	柏	百	摆	佰	败	拜	裨	帮
B0Ex	班	搬	扳	般	颁	板	版	扮	伴	瓣	半	办	绊	邦	帮	
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	悲
B1Bx	卑	北	辈	背	贝	钡	倍	狈	备	惫	焙	被	奔	笨	本	笨
B1Cx	崩	绷	甬	泵	蹦	迸	逼	鼻	鄙	笔	彼	碧	蔽	蔽	毕	
B1Dx	毙	悖	币	庇	痹	闭	敝	弊	必	辟	壁	臂	避	陛	鞭	边
B1Ex	编	贬	扁	便	变	卞	辨	辩	辨	遍	标	彪	膘	表	瞥	憋
B1Fx	别	瘕	彬	斌	濒	滨	宾	宾								

繁体字フォント例 16x16 ドット

A74x	作	你	伯	低	伶	余	尙	佈	供	兌	克	免	兵	治	冷	別
A75x	判	利	刪	剗	劫	助	努	劬	匡	卽	卵	吝	吭	吞	吾	否
A76x	呎	吧	呆	呃	吳	呈	呂	君	吩	告	吹	吻	吸	吮	吵	訥
A77x	吠	吼	呀	吱	含	吟	听	函	困	囫	圇	坊	坑	址	坍	
A7Ax		均	坎	圾	坐	坏	折	壯	爽	妝	妒	妨	妞	妣	妙	妖
A7Bx	妍	妤	妓	妊	妥	孝	孜	孛	孛	完	宋	宏	趁	局	屁	尿
A7Cx	尾	岐	岑	岔	峯	巫	希	序	庇	床	廷	弄	弟	彤	彤	衍
A7Dx	役	忘	忌	志	忍	忱	快	忸	忸	戒	我	抄	抗	抖	技	扶
A7Ex	扶	扭	把	扼	找	批	扳	杼	扯	折	扮	投	抓	抑	技	改
A7Fx	攻	攸	早	更	束	李	杏	材	村	杜	杖	杞	杉	杆	杠	
A84x	杓	杙	步	每	求	汞	沙	沁	沈	沉	沅	沛	汪	決	沐	汰
A85x	沌	汨	冲	沒	汽	沃	汲	汾	沅	沅	沅	沅	沅	沅	沅	沅

全フォントテーブルについては、仕様書を参照して下さい。

5.8 コマンドテーブル

使用可能なコマンドにつきましては、ソフトウェア仕様書を参照して下さい。

GU-31xx シリーズと GU39xx シリーズは、画素数毎にソフトウェア仕様書が用意されています。

GU39xxB シリーズは、全品種をカバーした仕様書となっています。

5.9 カーソル移動と表示モード

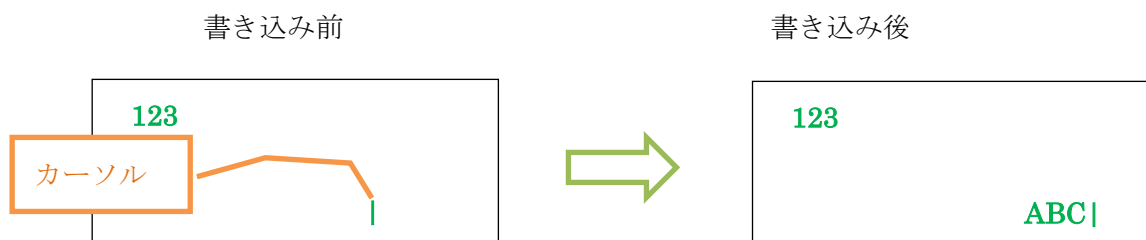
拡張シーケンスコマンドで設定する表示モードについて説明します。

表示モードは、画面端まで来たカーソルが次どうするかを指定します。

オーバーライトモード指定	1Fh 01h	表示モードをオーバーライトモードにします。
縦スクロールモード指定	1Fh 02h	表示モードを縦スクロールモードにします。
横スクロールモード指定	1Fh 03h	表示モードを横スクロールモードにします。

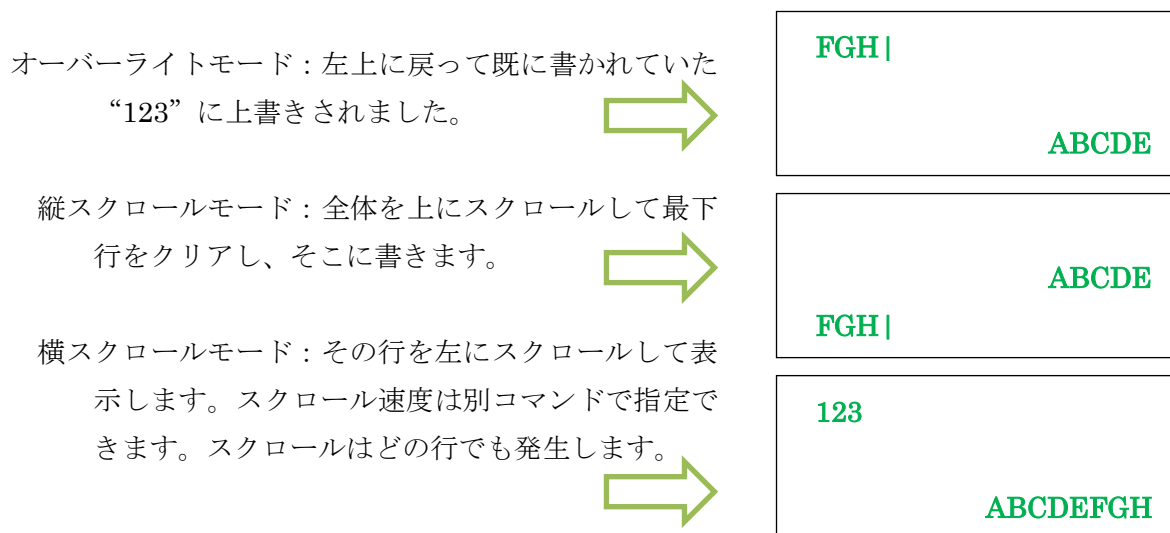
文字を書き込んだとき、カーソル位置に文字が表示されて、カーソルは1文字進みます。

例えば“ABC”と書き込んだ時のカーソルの動きはこのようになります。



表示モード指定でこの後の書き込み動作が変化します。

さらに“DEFGH”と書いたときの各モードの動作が次のようになります。



6 パラレルインターフェイスの場合のプログラム例

組み込み用のMPUをホストに使用して8 Bitパラレルインターフェイスで使用する場合の1バイトの書き込みプログラム例は次のようになります。 必要に応じて時間待ちを行ってください。

回路は 4.5.1 の場合を想定しています。

-----データ書き込みプログラム例-----

```
Void GU3000_out( char data)
{
    do { while ( READY == 0); /* Wait for ready */
        /* wait 1uSec, if necessary */
        DataLines = data;      /* Output data */

        WR = 0; /* set WR Low */
        /* wait 100nS, if necessary*/
        WR = 1; /* set WR High */
        /* wait 500nS, if necessary*/
    }
}
```

ホストが高速の場合には必要に応じて時間待ちを行ってください。

ホストが高速の場合には必要に応じて時間待ちを行ってください。

注：組み込みプログラムに関する一般的な注意ですが、次の点を考慮してください。

ポート操作に必要な最小時間により例の時間待ちは省略できます。ソフトウェアループによる時間待ちは、次の場合など期待した時間待ちにならない場合がありますのでご注意ください。

- コンパイラの最適化機構（オプティマイザ）がループを削除する。
- IOチャンネルが独立したコントローラの場合、CPUとIOが非同期で動作します。この場合はCPUの時間待ちがIOポート操作時間を遅くするとは限りません。

対策はご利用のシステムにより変わるとは思いますが、一般的に使える方法として次が考えられます。

- 時間待ちループ内にIO操作命令を加える。（「同じ出力を繰り返し出力する」「入力する」など）IO操作は最適化処理の対象外になっていると思いますが、必要に応じてVolatile指定などを行ってください。

7 表示デバイスとしての使用例（ホスト接続）

ホストに常時接続されて、受け取ったコマンドに従って表示内容を更新させる使い方です。

7.1 Windows PC を使った Microsoft 製 Visual Studio 2010 プログラム例

7.1.1 Visual C# 2010 を使ってシリアルポートに接続

GU-3xxx シリーズ表示モジュールに文字を表示させるサンプルです。

このサンプルは、C#2010 Express Edition(無償版) で実行できますのでお試しください。

シリアルポートには、市販の USB-シリアルアダプタや弊社の LAN アダプタがご利用いただけます。

まず、Microsoft 社のサイトから C#2010 をインストールします。

新しいプロジェクト→Windows フォームアプリケーションをダブルクリックします。

これであたらしいアプリケーションが作られます。

ツールボックスの中の SerialPort をダブルクリックします。

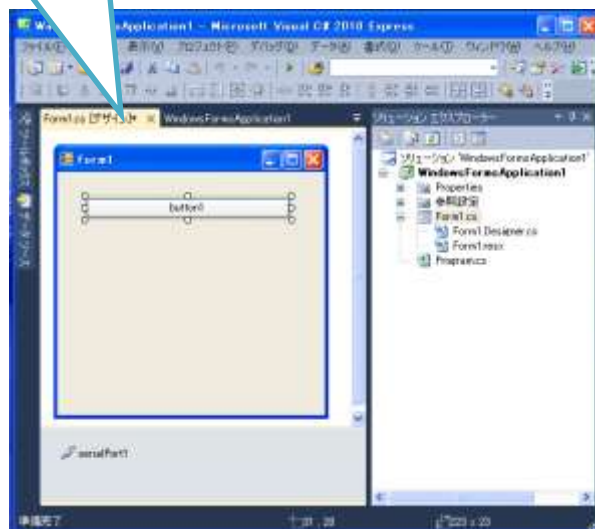
ツールボックスの中の Button をダブルクリックします。

Form1.cs をみるとウィンドウの左上にボタンがあると思いますので中央に格好よく移動しておきましょう。

右はここまでの画面イメージです。

Form1.cs[デザイン]タブ

ウィンドウのボタン以外のところをダブルクリックすると Form1_Load の定義入力ができるようになりますので、次のように入力してください。中身の最初の COM1 はシステムに合わせて適切なシリアルポート名に直してください。リスト中でグレイの部分はずでにありません。



----- ここから -----

```
private void Form1_Load(object sender, EventArgs e)
{
    this.serialPort1.PortName = "COM1";
    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(); }
}

```

----- リスト終了-----

Form1.cs[デザイン]タブをクリックしてウインドウのデザイン画面に戻ります。

Button1 ボタンをダブルクリックすると、ボタンクリックの処理入力が増加されますので次のように書き込みます。

----- ここから -----

```

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

```

----- リスト終了-----

全体はこうなります。

LIST C# サンプルプログラム

```

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 = "COM1"; //ポート名称はシステムにあわせる。

```

注を参照ください。

7.2.2 カーソル移動

(int X, int Y) に移動します。Y はバイト単位であることに注意してください。

```
/* 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();
    }
}
```

7.2.3 拡大文字

縦横整数倍に文字を拡大表示できます。

```
/* @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();
    }
}
```

7.2.4 テストプログラム

ここまでの Function のテスト用のコードです。Button2 から Button6 を作って、各々のクリックイベントを作って次のように書き込んでください。イベント処理させるには Form 1 にあるボタンをダブルクリックします。また X と Y のパラメータ設定用に TextBox を 2 個作ってください。名前は textBox1 と textBox2 から変えないでください。

```
private void button2_Click(object sender, EventArgs e)
{
    ClearScreen();
}

private void button3_Click(object sender, EventArgs e)
{
    moveCursor(int.Parse(this.textBox1.Text), int.Parse(this.textBox2.Text));
}

private void button4_Click(object sender, EventArgs e)
{
    fontMagnified(int.Parse(this.textBox1.Text), int.Parse(this.textBox2.Text));
}

private void button5_Click(object sender, EventArgs e)
{
    FontWidth(1);
}

private void button6_Click(object sender, EventArgs e)
{
    FontWidth(2);
}
```

7.2.5 プロポーショナル英数字

文字間隔を変更します。プロポーショナルフォントを使うと、表示可能な平均的字数が増えます。

```
/* 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)0x04;
    bb[4] = (byte)(w % 0x100);

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

7.2.6 漢字表示を設定する

CJK (Chinese-Japanese-Korean) フォントの設定を行います。

```
/* 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();
    }
}
```

7.2.7 漢字表示の使い方

書き込む文字コードは Shift-JIS コードになります。C#の場合内部コードが Unicode ですので S-JIS にエンコードしてから書き込む必要があります。次の例は、C#の button7 のクリックで日本語表示を行う場合の例です。

```
private void button7_Click(object sender, EventArgs e)
{
    const int JIS = 0;

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

    string str = "日本語表示します";
}
```

```
Encoding sjisEnc = Encoding.GetEncoding("Shift_JIS");
int NumberOfBytes = sjisEnc.GetByteCount(str);
byte[] bytes = sjisEnc.GetBytes(str);

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

7.2.8 グラフィックを表示する

バイト列に変換されたビットマップイメージをリアルタイムに表示管に書き込みます。表示エリアに書き込めば即時表示されます。非表示エリアに書いた場合は、グラフィックスクロールにより非表示エリアを表示エリアにスクロールさせると表示されます。

```
/* Realtime Bitmap Display
 *
 * image: bitmap image
 * X    : Horizontal size in Bit
 * Y    : Vertical 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();
    }
}
```

7.2.9 グラフィック表示実行例

グラフィックファイルを開いて `DrawBitmap()` を呼び出すサンプルプログラムを次に示します。`Button8` を追加してこのクリックイベントとして実行するように書かれています。ビットマップイメージのサイズが表示器の制限を超えると表示できませんので注意してください。またビットマップイメージの縦方向のドット数が8の倍数でない場合、端数部分は表示されません。画像確認用にツールボックスから `pictureBox` をダブルクリックして `Form` に追加しておいてください。

```
private void button8_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);

        bmp.Dispose();
    }
    openDia.Dispose();
}
```

7.2.10 グラフィックをスクロールさせる

表示用 RAM のデータを移動することで書き込み済みの画面のスクロールを行います。

```
/*
 * 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();
    }
}
```

7.2.11 グラフィックのスクロール実行例

Button9のクリックイベントとして実装した呼び出しプログラム例です。

Xsize と Ysize は、使用する表示器の縦横ドット数に書き換えてください。

```
private void button9_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);
}
```

7.2.12 裏画面を表示する

「7.2.8」のスクロールコマンドを使って、RAMの非表示エリアを表示エリアに移動できます。

Button10のクリックイベントとして実装した呼び出しプログラム例です。

Xsize と Ysize は、使用する表示器の縦横ドット数にあわせて書き換えてください。

```
private void button10_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);
}
```

7.2.13 文字をスクロール表示させる

文字のスクロールは、文字の書き込み時に行われます。

つまり、文字の横スクロールは次のような動作を行います。

- 1 : 横スクロールモードを設定したのち
- 2 : 文字書き込み位置が、画面右端に達すると、
- 3 : 次の文字書き込みで文字スクロール動作が行われる。

スクロール速度の設定もあわせて行ってください。

```
/*
 * 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();
    }
}

```

7.2.14 文字スクロール実行例

コマンドの使用例です。Button11のクリックイベントとしてプログラムしています。

```

private void button11_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();
    }
}

```

7.2.15 ユーザーウィンドウで画面を区別してつかう。

ユーザーウィンドウを4個まで定義し、そのなかで文字表示などの表示を行うことができます。ユーザーウィンドウの設定、削除、選択コマンドを使用します。

```

//ユーザーウィンドウ定義
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();
        }
    }
}
//ユーザーウインドウ削除
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();
    }
}
//ユーザーウインドウ選択
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();
    }
}
}
```

7.2.16 ユーザーウインドウ使用例

Button12のクリックイベントに割り当てたプログラム例です。

画面の左上に50x16画素のユーザーウインドウを設定し、そこに“Window”と表示します。

引き続き文字スクロールやグラフィック表示を行うとこのユーザーウインドウ内に書き込まれます。

```
private void button12_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();
    }
}
```

ユーザーウィンドウを使わなくするときは `SelectCurrentUserWindow(0);`としてください。

7.3 LAN アダプタの TCP/IP ソケットによる接続

接続例 7.1 では、COM ポートを使ったプログラムを示しました。LAN アダプタ（4.5.3 参照）の仮想 COM ポートを使ってサンプルプログラムを使うことができます。

LAN アダプタに搭載している XPORT は、TCP/IP Socket を使用する方法でも接続が可能です。この場合のサンプルは EXCEL の VBA を使って用意しておりますので弊社 Web サイトよりダウンロードしてご利用ください。

名称 ME-131:Socket Driver Sheet

8 グラフィック DMA モード

□章でコマンドを使用した操作につき説明しましたが、コマンド解析に時間がかかるため応答に時間がかかります。グラフィック DMA モードでは、多くのコマンドに対応しないことで応答速度を速くしグラフィック書き込みの高速化を行います。

パラレルインターフェイスで使用できます。

9 自律動作用途での使い方（1）

FlashROM にマクロおよびプログラムマクロを登録することで、自律的に表示を行うことができます。電源投入での自動起動ができます。プログラムマクロでは汎用 IO ポートやインターフェイスポートを使って表示内容を変えたり、特定の条件でデータを送信したりできます。

自律動作によりご使用頂きやすくするため、FlashROM データ作成書込み Windows ソフトウェアを用意しております。以下の使用形態に対応しておりますので選んでご利用ください。

9.1 アクション付の表示を繰り返す。(名称：ME-021 itron 楽画記帳)

文字とビットマップ画像をアクション付で組み合わせて表示させるメッセージを作り Flash ROM に登録します。登録後は電源のみで表示を繰り返します。表示を更新するときのみ PC と接続する必要があります。



9.2 汎用 IO ポートの状態に対応して表示内容を変える (名称 PortLinkerDx)

Microsoft-EXCEL で記述されたアプリケーションで、シートへの入力内容を表示器の Flash ROM に書き込みます。汎用 IO ポートをつかい、入力 9 本、出力 3 本まで対応しています。



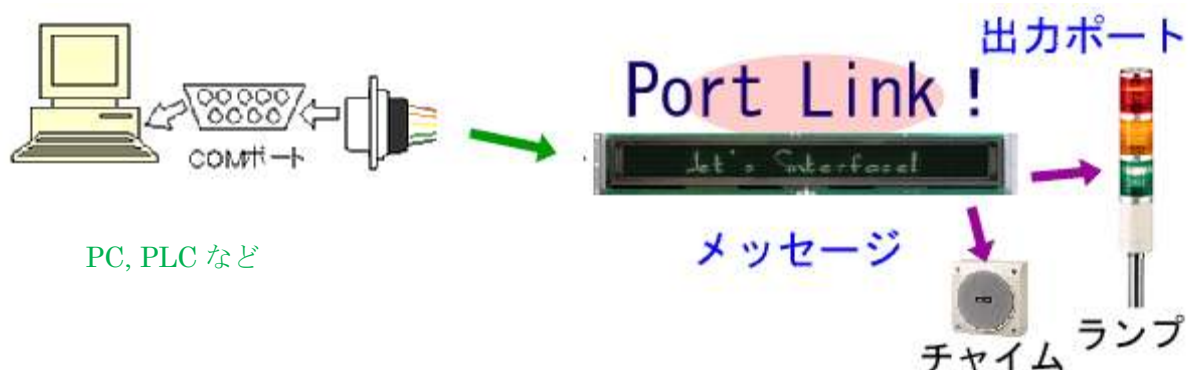
◆例えば、
人が近づいたら質問メッセージと選択肢を表示しチャイムで注目させ、
押されたボタンに対応のメッセージをランプと共に表示、
一定時間後に終了メッセージを流したあと表示を消灯。。。
こんな動作をセルへの入力だけで創り込むことができます。

9.3 シリアル及びパラレルで書き込まれたデータに対応した表示を行う。

(名称 ComPortLinkerDX)

Microsoft-EXCEL の VBA で作られておりシートへの入力内容を表示器の Flash ROM に書き込みます。インターフェイスパラレルポートからも入力できます。

PortLinkerDX と類似機能ですが、こちらはインターフェイス入力により制御されます。



9.4 TCP/IP ポートよりのデータに対応した表示を行う（名称 ME-141:TcpPortLinkerDX）



9.5 自分でマクロプログラムを作成する。（名称 ME-023 : Macro 操師）

プログラムマクロのコンパイラです。

弊社 Web サイトよりダウンロードしてご利用ください。

9.6 RSS リーダー（名称 ME-136）

LAN アダプタを接続した GU-3900 シリーズにマクロプログラムを書き込みインターネットの RSS リーダーとしてカスタマイズします。

用途によっては著作権上の許諾が必要な場合があります。お客様の責任の下で十分にご注意してご利用下さい。

10 いろいろな操作（プチツール）

10.1 使用方法 ほか。

ダウンロードした *.zip ファイルを解凍し、その中の *.exe ファイルを任意の場所にコピーしてお使い下さい。不要になった場合は、*.exe ファイルの削除でアンインストールできます。

Windows2000、XP で動作確認していますが、USB-RS232 アダプタなど PC とアダプタの相性などの要因で動作しない場合があるかもしれません。あらかじめご了承ください。また Windows98SE、Me でも動作可能な場合もあるようです。ご確認の上、ご活用頂ければ幸いです。

10.2 利用条件

本ソフトは、ノリタケ伊勢電子製の表示モジュール GU-3000 シリーズ用の開発支援ソフトです。その目的の範囲で無償でご利用いただけますが、他への二次利用はご遠慮ください。

これらのソフトウェアを使用したことによって生じたいかなる結果についても当社は責任を負いません。損害の補償も致しません。ご意見、ご要望、ご質問、不具合（バグ）報告等はメールにてお願いします。

10.3 著作権

本ソフトウェアの著作権は ノリタケ伊勢電子株式会社に帰属します。

10.4 プチツール一覧

プチ・マクロ起動ツール：「M 起動隊」	ME-030	マクロ起動設定解除
プチ・スイッチャ ツール：「スイッチ賛」	ME-031	メモリスイッチを設定
プチ・メモリ検視ツール：「MR I」	ME-032	表示 RAM 内容確認
プチ・バージョンチェッカー：「管定士」	ME-033	表示器のプロパティ確認
プチ・クリーナ：「クリンちゃん」	ME-034	メモリ内容の消去
I/O ポート確認ツール：「お出入 GU」	ME-035	I/O ポートを設定、制御
楽画記帳停止ツール：「画記封じ」	ME-104	itron 楽画記帳のマクロ停止
表示画面キャプチャツール：「GuCapture」	ME-111	表示 RAM 内容を PC に保存

10.4.1 プチ・マクロ起動ツール：「M 起動隊」 ME-030

マクロの起動を支援する道具です。

表示モジュールに登録されたマクロやプログラムマクロを起動させたり、またそれらの自動起動設定を表示モジュールのメモリスイッチに登録します。

10.4.2 プチ・スイッチャ ツール：「スイッチ賛」 ME-031

メモリスイッチを設定する道具です。表示モジュールのメモリスイッチの状態を確認したり、変更したりするときに便利なツールです。

10.4.3 プチ・メモリ検視ツール：「MR I」 ME-032

ビットマップメモリを覗き見る道具です。

表示モジュールに画像パターンが期待通り書き込まれたかどうかを確認するときに便利なツールです。表示モジュールには画像領域のデータを読み出すコマンドは無いので、マウスカーソルをメモリーマップの上にかざすだけという簡単な操作で、診たい領域を表示モジュールに表示させて確認します。

10.4.4 プチ・バージョンチェッカー：「管定士」 ME-033

表示モジュールのプロパティを鑑定する道具です。

表示モジュールのバージョンやその他のプロパティを確認したいときに便利なツールです。

10.4.5 プチ・クリーナ：「クリンちゃん」 ME-034

表示モジュールの内部を掃除する道具です。

注意：USB 接続タイプ（GU -3101 シリーズ等）では使用できません。

デモなどで表示モジュール内部にいろいろな画像パターンを登録したり、特殊なメモリーSW の設定をしたりした後に、元通りのきれいな状態に戻りたいときに便利なツールです。

10.4.6 I/O ポート確認ツール : 「お出入 GU」 ME-035

表示モジュールの I/O ポートを設定、制御する道具です。

開発作業の中で、表示モジュールの I/O ポートの状態を確認したり、試しに変更したりするときに便利なツールです。

10.4.7 楽画記帳停止ツール : 「画記封じ」 ME-104

itron 楽画記帳で登録されたプログラムマクロ (PMacro) を停止する道具です。

メッセージエディター「itron 楽画記帳」で表示登録したモジュールに対し、プログラムマクロを停止させノーマルコマンドモードにしたい場合に便利なツールです。必要に応じて自動起動設定の解除もできます。

10.4.8 表示画面キャプチャツール : 「GuCapture」 ME-111

表示された画面を、PC に吸上げ、WindowsBmp 画像として保存する開発支援ソフトです。操作マニュアル作成時などに重宝します。(吸上げ処理の実行に指定した Macro を 1 つ使用します。通常は RAM をご指定下さい。それ以外では自動起動設定がされているとテストモード以外での解除ができなくなりますのでご注意下さい。)

11 トラブルシューティング

11.1 READY 信号について

パラレルインターフェイス用の **READY** 信号とシリアルインターフェイス用の **DTR** は異なる信号です。スクロールなど内部処理に時間のかかるコマンドが多くあります。書き込みに関してはハードウェアハンドシェイクを行ってください。

11.2 リセットについて

表示器のコントローラは電源投入時に内部の初期化処理を行い、これが終了するまではデータの受付ができません。初期化中は **READY** 信号が **BUSY** 状態になりますので、この期間はデータやコマンドを書き込まないようにしてください。

初期化完了直後の内部状態はメモリスイッチの設定により変更できます。

メモリスイッチ変更ツールを用意しております。(10.4.2 プチ・スイッチャ ツール ME-031)

11.3 プログラムマクロが終了できないときの回復方法

プログラムマクロは、何らかの条件で終了するようにプログラムされていないと終了しません。このため、デバッグ中などでプログラムマクロが終了できず、自動起動が設定されていると新しいマクロを書き込むこともできなくなります。このような場合は、次の手順でマクロを止めてください。

STEP1 電源を OFF します。

STEP2 テストモードを設定します。

方法は”[11.5 セルフテストモード設定方法](#)”を参照してください。

STEP3 電源を投入します。(これでテストモードが開始します。)

STEP4 テストモード設定を解除します。

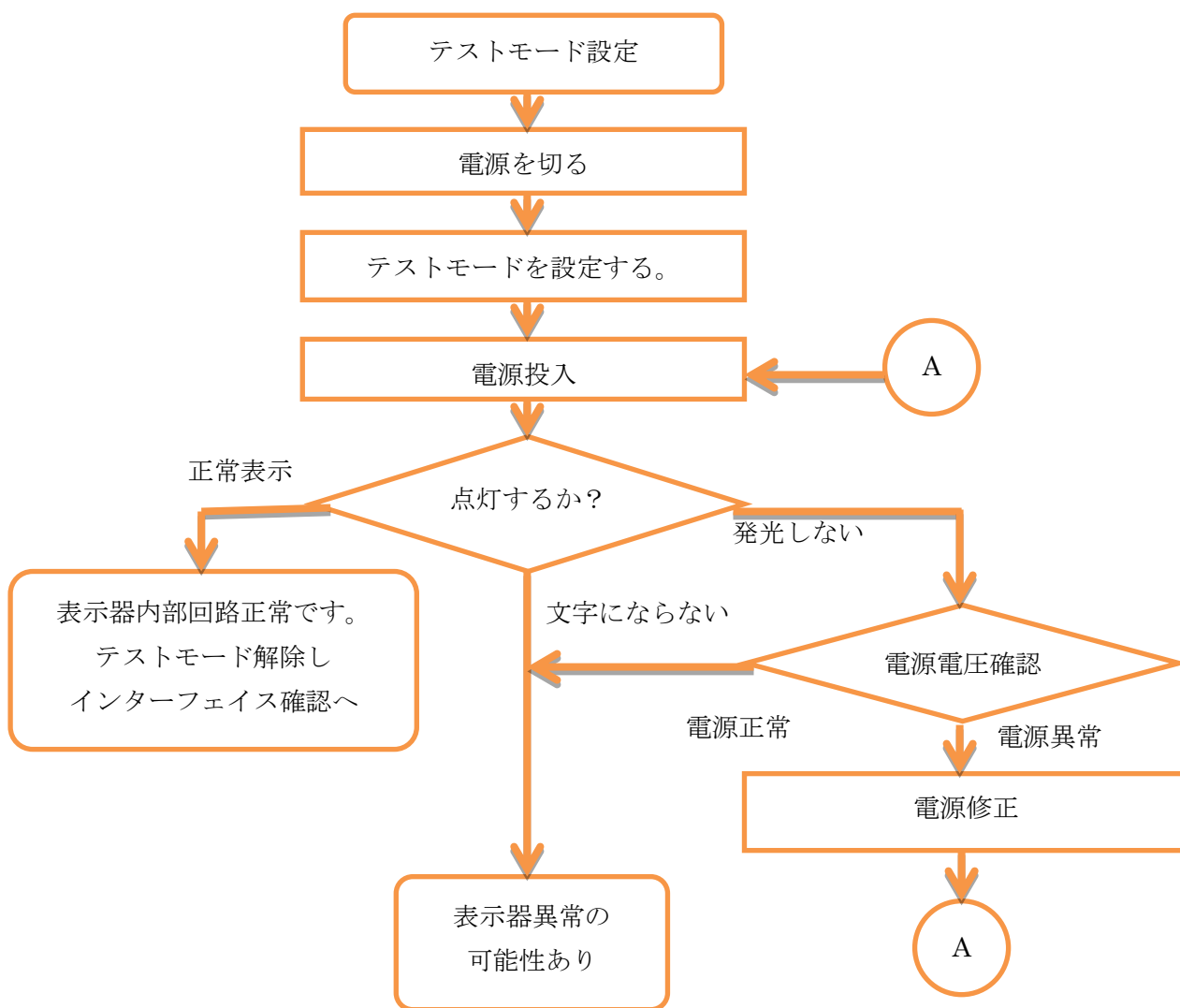
STEP5 これでコマンドを受け付けるようになりますのでマクロの自動起動を解除してください。起動解除には 9.6 プチツールをご利用下さい。

11.4 まったく点灯しないとき---セルフテストモード

表示モジュールは、特別な初期設定なしで ASCII コードを入力するだけで英数字の表示が可能ですが、まったく点灯しない場合は、駆動回路の問題なのか表示モジュール自体に問題があるかの切り分けが必要になります。このために、電源投入により自動でテストパターンの表示を行う機能を標準搭載しております。

テストモードの設定方法は次章を参照してください。

テストモード利用フロー

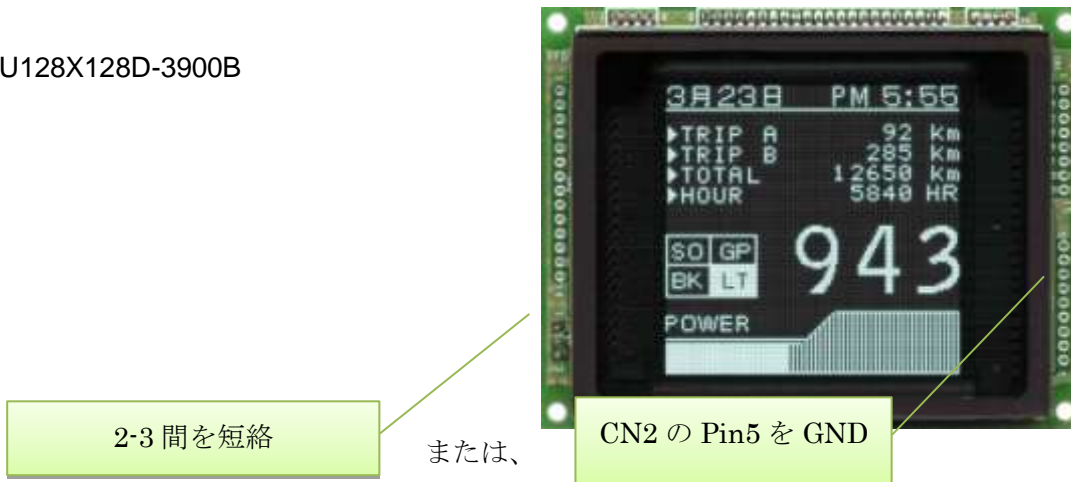


11.5 セルフテストモード設定方法

品種ごとに異なりますが、電源コネクタ搭載品では、TEST ピンを GND に接続して電源を投入することでテストモードとなります。写真の縮尺は品種によって異なります。

プログラムマクロを終了するときは、電源投入しテストモード開始後にジャンパを外してください。

11.5.1 GU128X128D-3900B

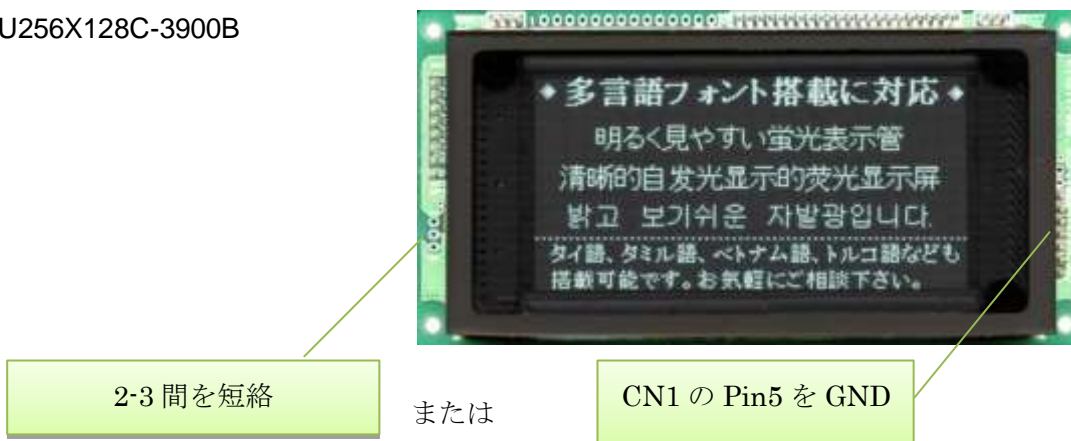


2-3 間を短絡

または、

CN2 の Pin5 を GND

11.5.2 GU256X128C-3900B

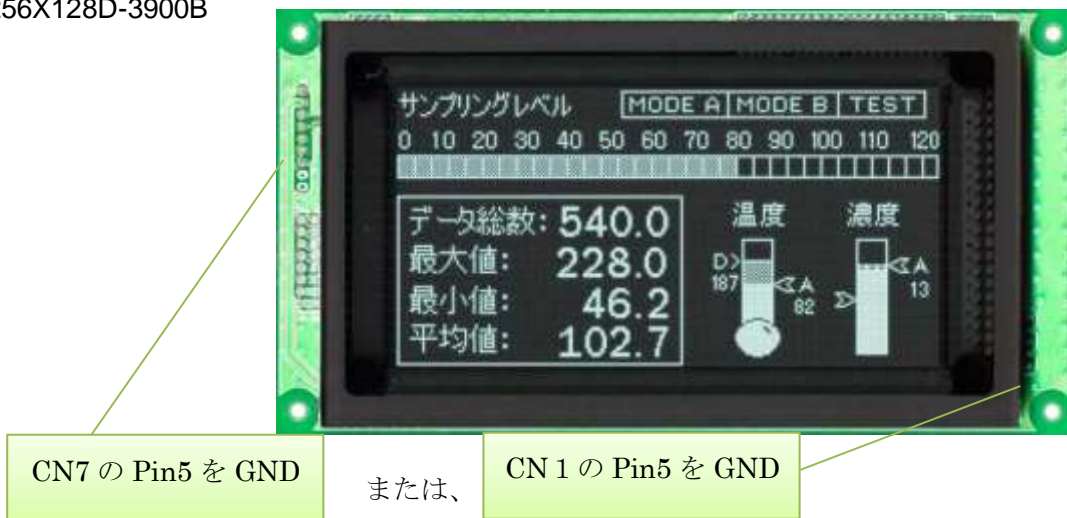


2-3 間を短絡

または

CN1 の Pin5 を GND

11.5.3 GU256X128D-3900B



CN7 の Pin5 を GND

または、

CN 1 の Pin5 を GND

11.5.4 GU256X128E-3100、GU256X128E-3900

CN 1 の Pin8 を GND



11.5.5 GU256X128E-3910

CN 1 の Pin5 を GND

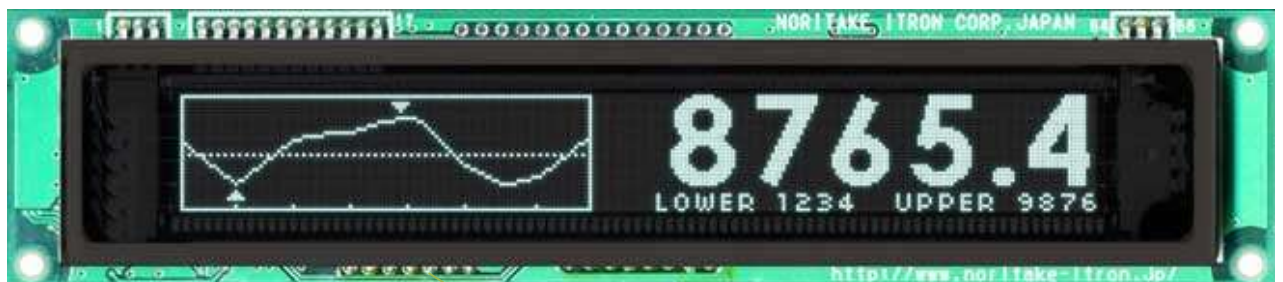


11.5.6 GU256X16M-3900、GU256X16M-3900B

CN6 の Pin5 を GND



11.5.7 GU256X32D-3100、GU256X32D-3900、GU256X32D-3900B



CN3 の Pin5 を GND

11.5.8 GU256X32L-3900



CN5 の Pin5 を GND

11.5.9 GU256X64C-3100、GU256X64C-3900、GU256X64C-3900B

CN3 の Pin5 を GND



11.5.10 GU256X64D-3100、GU256X64D-3900、GU256X64D-3900B

CN3 の Pin5 を GND



11.5.11 GU256X64D-3101

CN3 の Pin2,Pin3 間を短絡する。

11.5.12 GU256X64E-3100、GU256X64E-3900、GU256X64E-3900B



CN1 の Pin5 を GND

11.5.13 GU256X64E-3101



CN1 の Pin3 を GND

11.5.14 GU256X64F-3100、GU256X64F-3900



CN2 の Pin5 を GND

11.5.15 GU256X64F-3101



CN5 の Pin5 を GND

11.5.16 GU320X32D-3900



CN2 の Pin5 を GND

11.5.17 GU384X32L-3100、GU384X32L-3900、GU384X32L-3900B



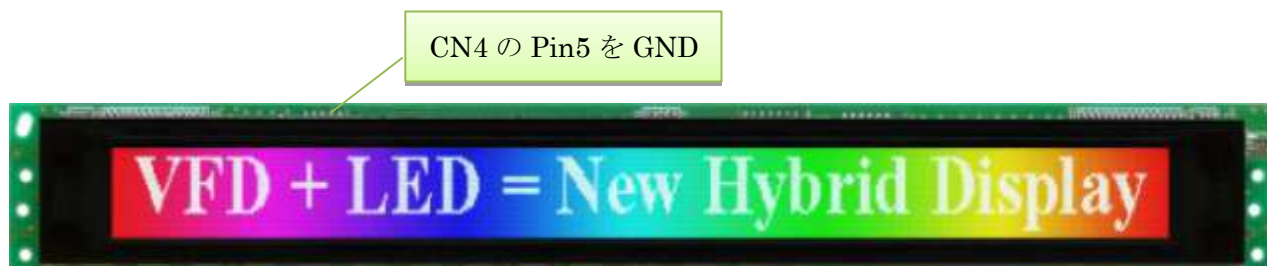
11.5.18 GU384X32L-3950B

3900B のスリム外形品です。テストモード起動方法は 3900B と同じです。

11.5.19 GU512X32H-3100、GU512X32H-3900、GU512X32H-3900B



11.5.20 GU512X32H-3940、GU512X32H-3940B



12 支援 TOOL

初期評価やデータ作成を支援する TOOL を弊社 Web で配布しておりますのでご利用ください。Web にない TOOL もありますのでお気軽に営業担当並びに弊社カスタマーサポートまでお問い合わせください。

弊社ホームページ：

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

技術サポートページ：

<http://www.noritake-itron.jp/cs/index.htm>

12.1 UNIDEMO (ユニデモ)

動作環境：Windows PC

- ・簡易言語記述により、ループや条件分岐などを使ってモジュールの動作 確認プログラムが作成できます。
- ・パソコンの通信ポートとモジュールの間でやりとりをして、実際的な動作 確認デモができる通信機能があります。

12.2 COMTEST (コムテスト)

動作環境：Windows PC

- ・蛍光表示管モジュールの動作テスト用通信ソフトです。
- ・送信データをテキスト形式、またはコード形式で入力でき、簡単にデータ通信が可能です。

13 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。

13.1 RoHS 指令対応

GU-3100, 3900, 3900B 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。
- 電子部品中のセラミックとガラスに含有する酸化鉛。
- 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。

14 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。

15 免責と制限事項について

本文書に記載の情報並びに紹介している TOOL 類は、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。

解説内容はシリーズを網羅的に取り扱っておりますので、一部の特定の品種では例外的に当てはまらない可能性があります。最終的には個々の品種の仕様書にてご確認のうえ御使用頂きますようお願いいたします。

サンプルコードは弊社製品を使用するためのみに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いたします。

インストーラの形で提供する支援 TOOL にはライセンスを受けて使用しているプログラムを含む場合があります。解析、逆コンパイル、逆アセンブルなどのリバースエンジニアリングに類する行為は行わないでください。

16 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。

以上