# 漢字･簡易グラフィック端末

# GT100

# 取扱説明書

Cyber MELON

株式会社　インターネット

# ユーザーズマニュアル履歴

| Rev. | 改訂日付 | 内　容 |
|---|---|---|
| 1.00 | 2009/ 2/11 | 初版リリース |
| 1.10 | 2011/ 1/ 5 | 各種コマンド追加 |
| 1.11 | 2012/ 2/19 | 描画コマンド追加・整理 |

☆本マニュアルの最新版は当社ホームページからダウンロードいただけます。

# 目次

## ５．動作テストとサンプルプログラム



## ６．その他

# 1. はじめに

このたびはサイバーメロン 漢字・簡易グラフィック端末ボード GT100 をお買いあげいただきまして、誠に有り難うございます。
GT100 は文字、グラフィックスの STN 液晶表示に各種 I/O 機能を備え製品開発、組み込みなどに最適な汎用製品です。
ご使用に当たりましては本書を良くお読みいただき正しい取り扱い方法をご理解の上ご使用いただきますようお願い致します。

## ご注意事項

1. 本製品の仕様、および本書の内容に関して事前の予告なく変更することがありますのでご了承ください。
2. 本製品の使用によるお客様の損害、および第三者からのいかなる請求につきましても当社はその責任を負いかねますので予めご了承ください。
3. 本製品に付属のサンプルプログラムはその動作を完全保証するものではありません。製品に組み込んで使用される場合にはユーザ様にて十分なテストと検証をお願いします。
4. 本製品および本書に関し、営利目的での複製、引用、配布は禁止されています。

# ご使用に当たって

1. 梱包品の内容をまずご確認ください。
2. ご使用になる前に下記の安全についての注意を必ずお読みください。
3. 通電する前に、本製品の使い方を十分ご確認いただき、正しい接続と設定をご確認ください。

# 安全についてのご注意

1. 本製品を医療機器など人命に関わる装置や高度な信頼性・安全性を要求される装置へ搭載することはご遠慮ください。
   その他の装置に搭載する場合でもユーザー様にて十分な信頼性試験・評価をおこなった上で搭載してください。
   また非常停止や緊急時の制御は外部の独立した回路にておこなってください。
2. 本製品の改造使用は発熱、火災などの原因となり危険ですのでご遠慮ください。
3. 本製品のマニュアル記載環境以外でのご使用は故障、動作不良などの原因になりますのでご遠慮ください。
4. 本製品は導電部分が露出しておりますので、金属パーツなどショートの可能性のあるもの、液体のこぼれる可能性のある場所の近くでの使用はお控えください。
   また装置に組み込む場合も絶縁に関しては十分な注意を払ってください。
5. 電源は必ず本製品専用（指定）のものをお使いください。
   電圧、極性、プラグ形状など異なるものをご使用になりますと故障の原因となるばかりでなく、火災など重大事故に繋がる危険性があります。
6. 本製品に触れる前には体から静電気を除去してください。
7. 本製品には落下など強い衝撃を与えないでください。

# 使用環境

■ 以下の環境でのご使用はお控えください。

・強い電磁界や静電気などのある環境
・直射日光の当たる場所、高温になる場所
・氷結や結露のある場所、湿度の異常に高い場所
・薬品や油、塩分などのかかる場所
・可燃性の気体、液体などに触れる場所
・振動の多い場所、本製品が静止できない場所
・基板のショートを引き起こす可能性のある場所

# 規格取得など

■ 本製品は UL CSA 規格、CCC 認証など取得しておりません。
装置に組み込む場合は各安全規格への適合性をユーザー様で
ご確認いただき、対応して頂きますようお願いします。

■ また本製品は RoHS(特定有害物質の使用制限指令)に対応しておりません。

# 製品保証と修理

・本製品の保証は商品到着後１０日以内の初期不良のみ無償交換
とさせていただきます。
本マニュアルに記載するテスト手順にて正しく動作しない場合は
ただちに電源を切って、当社ホームページのサポートからご連絡
ください。　折り返し交換手順をご案内いたします。
・保証期間中であってもユーザー様の責となる故障（落下や電源の
誤接続など）は有料修理になります。
・その他の故障やクレームにつきましても当社ホームページ
http://www.cyber-melon.com　サポートコーナーよりご連絡ください。

# 梱包内容をご確認ください 

| | | |
|---|---|---|
| 1. GT100 ボード本体 | 1 |  |
| 2. 説明書 CD-ROM | 1 |  |
| 3. 電源アダプター | 1 |  |
| 4. RS232C ケーブル | 1 |  |
| 5. SD カード | 1 |  |

# 動作チェック

付属の 5V 電源アダプターを GT100 本体に接続して通電してください。

・まずボード上２カ所の赤い LED が２回点滅してボードの初期化が完了したことを知らせます。

・液晶パネルにフォントとグラフィックのデモを５秒ずつ交互に表示します。

・ 付属の SD カードをパソコンで FAT32 にフォーマットし、付属 CD-ROM の Font フォルダーから “kfont12.fnt” ファイルを SD カードのルートにコピーして、GT100 カードソケットに挿入してください。
その時点でリセットボタン（黒）を押すと、こんどは漢字のデモを含めた画面（2.3 の内容）が液晶パネルに５秒ずつ表示されます。

・とりあえず RS-232C での動作の感触を試してみるには **5.1 ターミナルソフトからの描画テスト** から始めてください。
その後 **4.1** から 4 章に進んでいただくのが良いと思います。

もし動作異常が認められた場合は電源をはずして当社ホームページ
http://www.cyber-melon.com のサポートから症状をご連絡ください。
対処方法をメールまたは電話でご連絡いたします。

# 2. 概要

GT100 はマイコンやパソコンと RS-232C *1)で接続して文字（英数、かな、漢字）や簡易グラフィックの表示をおこなう モノクロ 128x64dot の STN 液晶端末ですが RS-232C から SD/MMC カードのファイル読み書きをおこなう機能や、汎用ポート制御など有用な機能も持っております。
組み込みの制御マイコンやＰＣから簡単なコマンド（電文）を送ることで下記の機能を使うことができます。

*1) 正式には EIA-232-D ですが本マニュアルでは通称の RS-232C で統一します。

## 2.1 特長

GT100 は以下の機能と特長を有しています。

- ◆ 漢字を含む文字列の描画。
- ◆ 直線や矩形、点、円、塗りつぶしなどの簡易グラフィック描画。
- ◆ SD/MMC カードのファイルの読み書き。 RS-232C から SD カードを利用可能。
- ◆ PS2 キーボードの読み込み。
- ◆ 32 ビットの汎用 I/O ポート制御（ブザーや音声ボード、リレーの制御、スイッチ読み込みなど)。
- ◆ C 言語によるサンプルソースプログラムを参考に組み込み機器などから GT100 の呼び出しが簡単におこなえます。
  可能なコマンドの種類は 4.2.5 のコマンド一覧表を参照してください。

## 2.2 主な用途

本製品の主な用途として以下のアプリケーションが考えられます。

・各種産業用マイコン制御システムへの組み込み
・生産工程管理システムや工作機械などの入出力端末
・測定機器への組み込み
・通信システムの端末

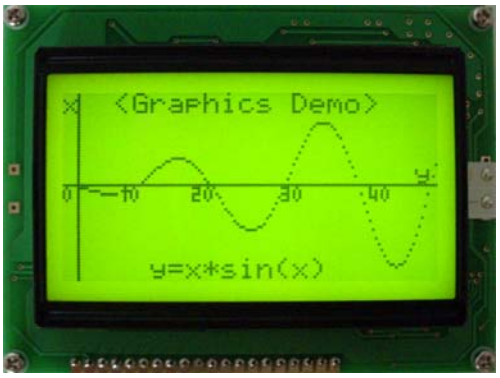
# 2.3 ボード仕様

| 項 目 | 仕　様 |
|---|---|
| CPU | PIC32MX360F512L |
| | Flash ROM 容量 512KB, 内蔵 RAM 容量 32KB |
| 動作クロック | 内部 75.6MHz, 周辺バスクロック 18.9MHz |
| | |
| 電源電圧 | DC5V 入力 (CPU 周辺動作 3.3V ) |
| 入出力 | RS232C コネクター |
| | PS2 コネクタ (キーボード読み込み専用) |
| | MMC/SD カードソケット |
| STN 液晶表示器 | モノクロ 128 x 64 dot |
| 液晶発光面 | 72 x 40mm |
| 入出力 | タクトスイッチ（２個） |
| | LED（２個） |
| | リセットスイッチ |
| | 汎用 I/O コネクター１( 2mm ピッチ 50 ピン) |
| | 汎用 I/O コネクター２( 2mm ピッチ 14 ピン) |
| 基板寸法 | 99.1(W) x 79.4(D)x 16.8(H) mm (基板のみ) |
| | 99.1(W) x 83.9(D)x 37.4(H) mm (突起部、液晶含) |
| 重量 | 138g　(液晶表示器含む) |
| | |
| | |
| | |

# 2.4 液晶表示サンプル

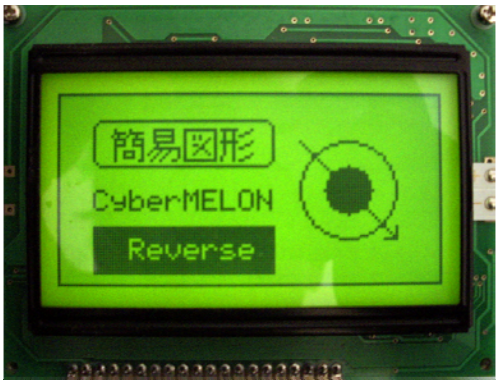
様々な表示フォーマットに対応します。


ASCII 文字( 21 文字 x 8 行 )




簡易グラフィック表示( 128 x 64 dot )


漢字表示 ( 全角 10.5 文字 x 5 行 )




簡易グラフィック表示 -2


白黒反転文字

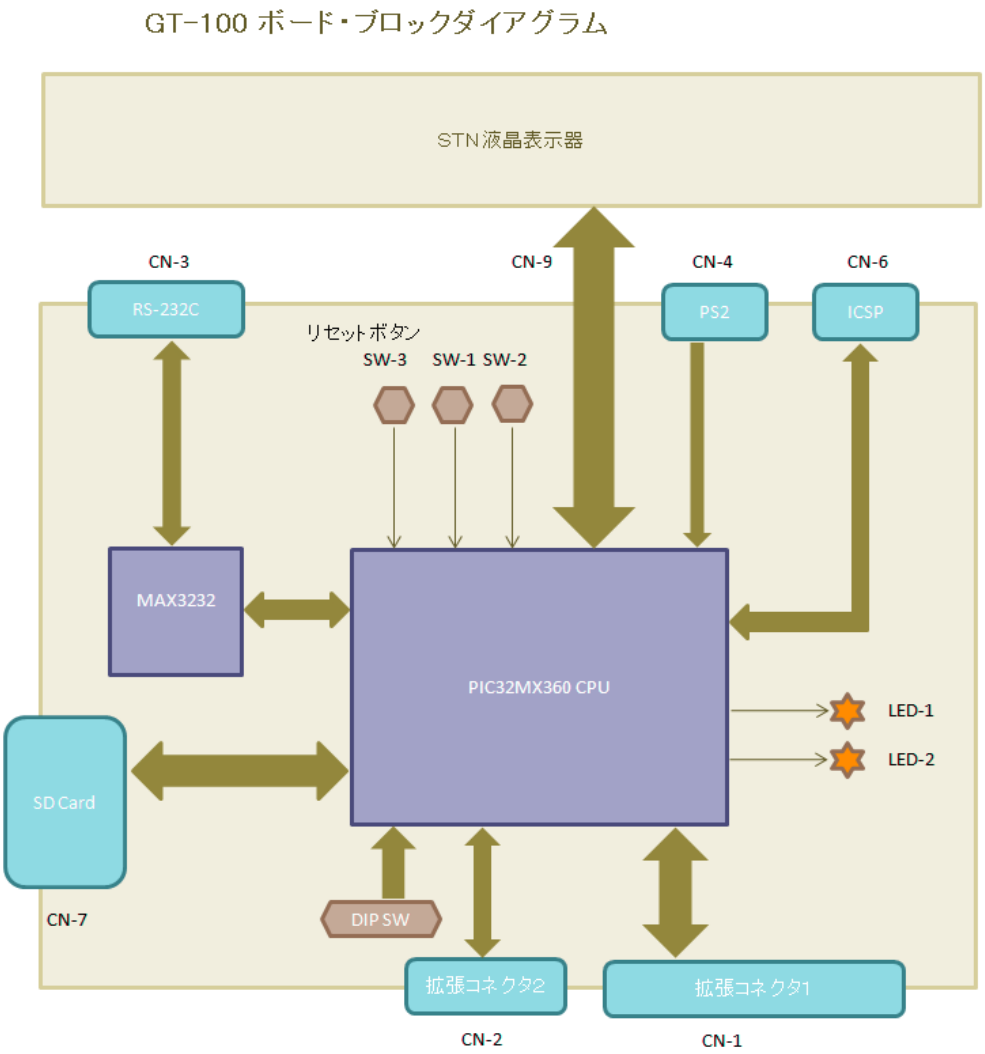

ボタンの移動サンプル

フォントサイズは ASCII は 8x6 dot, 漢字は 12x12 dot 混在も可能です。

DIP スイッチ - 4 を ON にした状態で電源を入れるとこのデモ画面が表示されます。

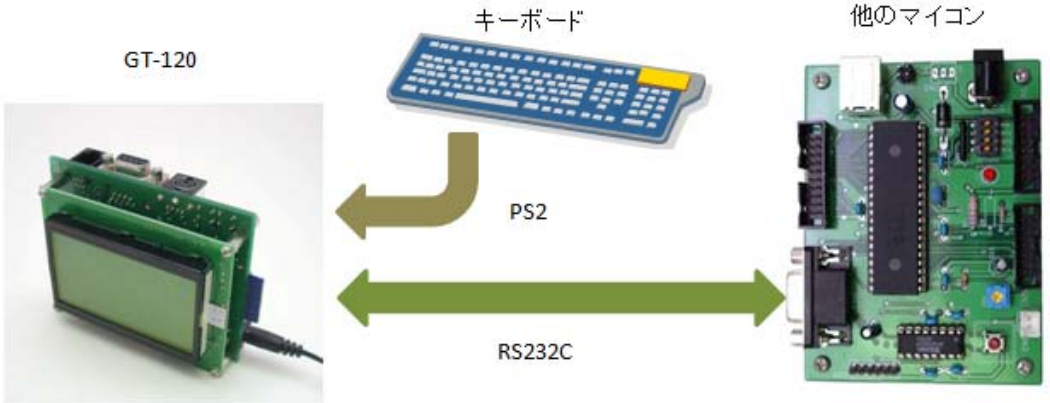
# 3．ハードウェア

## 3.1 ブロック図

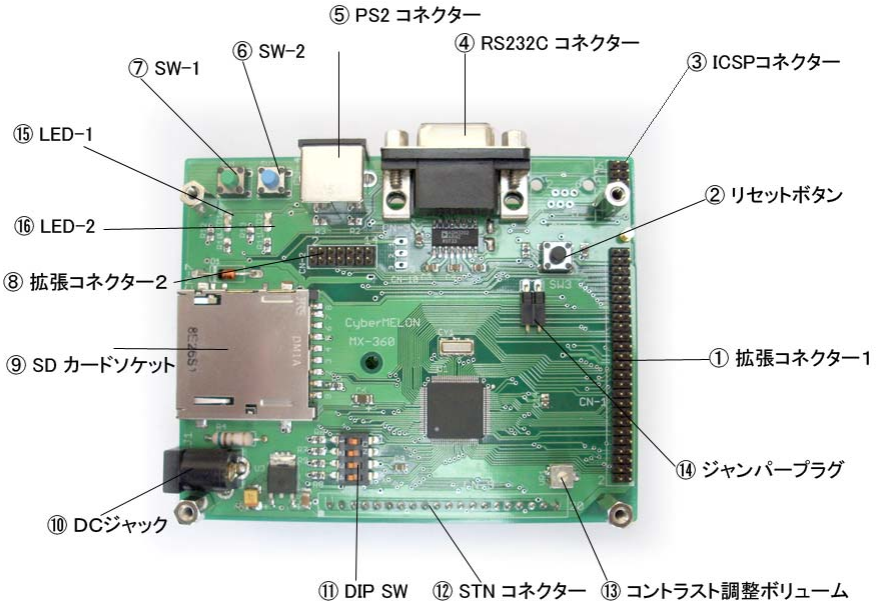


## 3.2 外観・接続図

GT100 の外観

# 3.3 各部名称と機能




① 拡張コネクター１　　ポート入出力コネクターです。(詳細は 3.4)
② リセットボタン　　CPU をリセットします。
③ ICSP コネクター　　外部プログラマー/デバッガー用です。(未使用)
④ RS232C コネクター　　９ピン D-SUB メスのコネクターです。
⑤ PS2 コネクター　　キーボード入力専用です。
⑥ SW-2　　テスト用汎用スイッチ入力です。
⑦ SW-1　　テスト用汎用スイッチ入力です。
⑧ 拡張コネクター２　　ポート入出力コネクターです。(詳細は 3.4)
⑨ SD カードソケット　　MMC/SD カードを挿入します。
⑩ DC ジャック　　付属の DC+5V スイッチング電源を接続します。
⑪ DIP SW　　通信のボーレートやデモモードを設定します。
⑫ STN コネクター　　STN 液晶表示器が接続されています。
⑬ コントラスト調整ボリューム　液晶表示器のコントラストを調整します。
⑭ ジャンパープラグ　　RS-232C の受信をイネーブルします。
⑮ LED-1　　ボードの状態を表示します。
⑯ LED-2　　ボードの状態を表示します。

# 3.4 コネクター・ピン配置

CN-1　拡張コネクター１

| PIN | 機 能 | 入出力 | PIN | 機 能 | 入出力 |
|---|---|---|---|---|---|
| 1 | GND | Out | 2 | GND | Out |
| 3 | N.C. | | 4 | N.C. | |
| 5 | PortA-0 | In/Out | 6 | PortA-1 | In/Out |
| 7 | PortA-2 | In/Out | 8 | PortA-3 | In/Out |
| 9 | PortA-4 | In/Out | 10 | PortA-5 | In/Out |
| 11 | PortA-6 | In/Out | 12 | PortA-7 | In/Out |
| 13 | N.C. | | 14 | N.C. | |
| 15 | N.C. | | 16 | N.C. | |
| 17 | N.C. | | 18 | N.C. | |
| 19 | N.C. | | 20 | N.C. | |
| 21 | N.C. | | 22 | N.C. | |
| 23 | N.C. | | 24 | N.C. | |
| 25 | N.C. | | 26 | +5V | |
| 27 | N.C. | | 28 | N.C. | |
| 29 | N.C. | | 30 | N.C. | |
| 31 | PortB-0 | In/Out | 32 | PortB-1 | In/Out |
| 33 | PortB-2 | In/Out | 34 | PortB-3 | In/Out |
| 35 | PortB-4 | In/Out | 36 | PortB-5 | In/Out |
| 37 | PortB-6 | In/Out | 38 | PortB-7 | In/Out |
| 39 | PortC-0 | In/Out | 40 | PortC-1 | In/Out |
| 41 | PortC-2 | In/Out | 42 | PortC-3 | In/Out |
| 43 | PortC-4 | In/Out | 44 | PortC-5 | In/Out |
| 45 | PortC-6 | In/Out | 46 | PortC-7 | In/Out |
| 47 | +3.3V | Out | 48 | +3.3V | Out |
| 49 | GND | Out | 50 | GND | Out |

品番：ヒロセ A3C-50P-2DSA　(2mm ピッチ 2x25)

注）N.C のピンはどこにも接続せずオープンにしてください。

## CN-2　拡張コネクター２

| PIN | 機 能 | 入出力 | PIN | 機 能 | 入出力 |
|---|---|---|---|---|---|
| 1 | GND | Out | 2 | GND | Out |
| 3 | PortD-0 | In/Out | 4 | PortD-1 | In/Out |
| 5 | PortD-2 | In/Out | 6 | PortD-3 | In/Out |
| 7 | PortD-4 | In/Out | 8 | PortD-5 | In/Out |
| 9 | PortD-6 | In/Out | 10 | PortD-7 | In/Out |
| 11 | N.C. | | 12 | N.C. | |
| 13 | +3.3V | Out | 14 | +3.3V | Out |

品番：ヒロセ A3C-14P-2DSA　(2mm ピッチ 2x7)

注）N.C のピンはどこにも接続せずオープンにしてください。

## CN-3　RS232C コネクター

| PIN | 機 能 | PIN | 機 能 |
|---|---|---|---|
| 1 | | 2 | TxD |
| 3 | RxD | 4 | |
| 5 | GND | 6 | |
| 7 | CTS | 8 | RTS |
| 9 | | | |

## CN-4　PS2 キーボードコネクター

| PIN | 機 能 | PIN | 機 能 |
|---|---|---|---|
| 1 | Data | 2 | TxD |
| 3 | GND | 4 | +5V |
| 5 | Clock | 6 | |

CN-6　ICSP コネクター（ファームウェア書き込み用-未使用）

| PIN | 機 能 | PIN | 機 能 |
|---|---|---|---|
| 1 | MCLR-/Vpp | 2 | VDD |
| 3 | GND | 4 | PGD |
| 5 | PGC | 6 | 未使用 |

CN-9　STN 液晶表示器 コネクター

| PIN | 機 能 | PIN | 機 能 |
|---|---|---|---|
| 1 | GND | 2 | +5V |
| 3 | VO | 4 | D/I |
| 5 | R/W | 6 | E |
| 7 | DB0 | 8 | DB1 |
| 9 | DB2 | 10 | DB3 |
| 11 | DB4 | 12 | DB5 |
| 13 | DB6 | 14 | DB7 |
| 15 | CS1 | 16 | CS2 |
| 17 | RESET | 18 | Vout |
| 19 | A | 20 | K |
| | | | |

CN-10　UART コネクター(5V レベルシリアル-未実装)

| PIN | 機 能 | PIN | 機 能 |
|---|---|---|---|
| 1 | GND | 2 | TxD |
| 3 | RxD | | |

（ 2.54mm ピッチ ）

## 3.5　ＴＴＬレベルでのシリアル通信について

CN-10（UART コネクタ）を実装すれば MAX3232 を介さない TTL レベルでのシリアル通信が可能になります。

この場合は MAX3232 からの Rx 信号と衝突しないようジャンパープラグ J1（ リセットスイッチ SW3 の左下）を必ずはずしてください。

尚、このケースでは CTS, RTS を使ったハードウェアによるハンドシェークは使えません。

詳しくは CD-ROM の Documents フォルダーにある回路図でご確認ください。

# 3.6　汎用入出力ポート

## 3.6.1　ポートの制御

GT100 には A, B, C, D の４組の８ビット汎用入出力ポート( GPIO )を持ちます。
各ポート、各ビットごとに入力または出力をプログラムできます。
電源投入時はすべてのポートが入力にセットされます。
以下の３種類のコマンドによって制御をおこないます。

**GPOT** ・・・ポートに８ビットデータを出力します。
**GPIN** ・・・ポートから８ビットデータを入力します。
**GPDI** ・・・ポートのビットごとの入/出力方向を指定します。

詳細は **4.3** の各コマンドの詳細をご覧ください。
各ポートのビットとコネクターピンの関係は **3.4** の CN-1, CN-2 の表を参照してください。

## 3.6.2　汎用ポートの電気的特性

各ポートは 3.3V CMOS レベルですが 5V までの入力を受け付けることができます。
それ以上の過大電圧では素子が破壊される可能性がありますので、できれば
外部回路でクランプするなどの対策をしてください。

| 項目 | 値 | 条件 |
|---|---|---|
| 入力 Low 電圧 | 0.66V max | 0.2 Vdd |
| 入力 Hi　電圧 | 3.17V min | 0.8Vdd ( 5V トレラント) |
| 出力 Low 電圧 | 0.4V max | IOL =　7mA@3.6V |
| 出力 Hi　電圧 | 2.4V min | IOH = -12.0mA@3.6V |
| 絶対最大シンク電流 | 25mA | |
| 絶対最大ソース電流 | 25mA | |
| | | |

# 4．ソフトウェア仕様

## 4.1 通信緒元

GT100 端末への描画やフォント、色指定などすべての機能は RS-232C で電文（コマンド）をホスト（マイコンや PC)から送ることで実現します。
RS-232C の通信パラメータは以下の通りです。

・非同期（ ASYNC 調歩同期 ）　**データ 8 ビット、パリティーなし、1 Stop ビット**
・端末側ボーレートは下記のとおり DIP スイッチで設定します。
ホスト（制御元）側も同じになるよう設定してください。
設定は電源投入時に一度だけ読み込まれます。変更した場合は電源を入れ直してください。

| DIPSW-1 | DIPSW-2 | DIPSW-3 | DIPSW-4 | ボーレート |
|---|---|---|---|---|
| OFF | OFF | Don't Care | Don't Care | 19200 |
| ON | OFF | Don't Care | Don't Care | 38400 |
| OFF | ON | Don't Care | Don't Care | 57600 |
| ON | ON | Don't Care | Don't Care | 115200 |
| | | | | |

設定はＰＣと接続してターミナル（端末）ソフトで通信を確認していただけます。
詳しくは　**5.1 ターミナルソフトからの描画テスト**を参照してください。

# 4.2 電文コマンドと応答

GT100 は RS-232C により文字列の電文コマンドを受け取って描画などをおこないます。

## 4.2.1　コマンドの形式(チェックサムなし)

・コマンドはすべて ASCII 文字列でパラメータの**数値はすべてヘキサ（１６進数）の文字列にて送ります。**

・コマンドの形式は

---

**コマンド文字列**（４文字）＋**パラメータ**（可変長）＋**ターミネータ**(CR)

---

（CR　= キャリッジ・リターンコード = 0x0d ）

GT100 はターミネータの CR を受け取った時点でコマンドを解析して実行します。

・GT100 はコマンド実行に成功すれば"OK" 失敗すれば "NGxx" の文字列を返します。（xx はエラー番号）応答のターミネータも CR です。

OK　などの応答が返るまでは次のコマンドを送らないようにしてください。

## 4.2.2　コマンドの形式(チェックサムあり)

・通信の信頼性を上げるために電文にチェックサムを付加することができます。

その場合のコマンド形式は

---

**コマンド文字列**（４文字）＋**パラメータ**（可変長）＋**チェックサム**(２文字）＋**ターミネータ**(CR)

---

となります。

チェックサムはチェックサムまでの文字の ASCII 値をすべて合計した数値の下８ビットを２桁ヘキサ文字にしたものです。

電源投入直後はチェックサムなしのモードでスタートしますので、後述の "CSUM" コマンドでチェックサムモードに変更すると、それ以後（CSUM コマンド自体は含まれない）のコマンドはチェックサムあり、として解釈されエラーチェックをおこない、応答も２桁のチェックサムを付加して返されます。

チェックサムの計算方法（計算例）は **4.3 コマンドの詳細仕様**の CSUM コマンドの解説を参照して　ください。

### 4.2.3　応答の種類

・ホストからの電文コマンドに対して GT100 は応答電文を返します。
コマンドには描画など実行を指示する**実行コマンド**とバージョンを取得したりする**問い合わせコマンド**の２種類がありますが、それによって応答形式が変わります。

１．実行コマンドに対する成功応答

---

"OK" + **ターミネータ**(CR)

---

２．実行コマンドに対する失敗応答

---

"NG" + nn（２文字のエラーコード）+ **ターミネータ**(CR)

---

3. 問い合わせコマンドに対する応答

---

**コマンド文字列**（４文字）＋**応答パラメータ**（可変長）＋**ターミネータ**(CR)

---

・コマンドと同様に　チェックサムモードを指定した場合は (CR)の前に２文字のチェックサムが付加されます。
・一定時間以内に応答が返ってこない場合はホスト側で Time Out エラーにしてください。
コマンドによって実行時間は様々ですが、通常１秒～２秒待てば十分です。

・実際の制御においてはコマンドを送ったあと"OK", "NGnn" などの応答を受け取ったあとで次のコマンドを送ってください。
応答を受け取る前に次のコマンドを送ると正しく実行されない可能性があります。

### 4.2.4　数値パラメータの表現

前述のようにコマンドに付随するパラメータの数値はヘキサで表現します。
たとえば 十進数の 30 はヘキサでは 0x1E ですから ４文字にして "1E" の文字列となります。
パラメータによっては１桁で送る場合もあります。

C言語のサンプル)

SendCommand( "LINE0C19481A");

動作：

座標(0C, 09) から 座標( 48,1A) に直線を引く。

十進座標では( 12, 9 ) から （72, 26 )になりますが、左上原点を(0,0) とします。

SendCommand は後述の C 言語サンプルにある電文に(CR)を付加して送信するルーチンです。　今後の C 言語サンプルではすべてこの形式で示します。

実際にはこのあとに 応答を受信する操作( サンプルでは GetReply() 関数) が必要になりますがコマンドの説明では省略されています。

### 4.2.5　コマンドの種類

コマンド文字列は以下の種類があります。　コマンドの基本部分は４文字の文字列でこれに必要に応じて数値や文字列のパラメータが付加されます。

| | |
|---|---|
| "PUTS" | 文字列の描画 |
| "CLRS" | スクリーン全面消去 |
| "DOT1" | １ドットの描画 |
| "DOT0" | １ドットの消去 |
| "DOT8" | 8 ドット分の一括描画 |
| "LINE" | 直線（自由方向）の描画 |
| "CIRC" | 円の描画 |
| "EBOX" | 矩形フレーム（Empty Box）の描画 |
| "RBOX" | 角丸矩形( Round Box)の描画 |
| "FILL" | 矩形の塗りつぶし |
| "RFIL" | 角丸矩形の塗りつぶし |
| "CFIL" | 円の塗りつぶし |
| "ERAS" | 矩形の消去（白塗りつぶし） |
| "CSUM" | チェックサムモード |
| "VERS" | バージョンの取得 |
| "GPOT" | 汎用出力ポートへの出力 |
| "GPIN" | 汎用入力ポートからの入力 |
| "GPDI" | 汎用ポートの方向制御 |
| "KBEN" | PS2 キーボードの入力許可 |
| "SDWR" | SD/MMC カードへのファイル書き込み |
| "SDRD" | SD/MMC カードからのファイル読み込み |
| "SDOP" | SD/MMC カードのファイルオープン |
| "SDCL" | SD/MMC カードのファイルクローズ |

・実際のコマンドを試してみるには　**5.1 ターミナルソフトからの描画テスト**　に飛んでください。

# 4.3 コマンドの詳細仕様

使用例では チェックサムとターミネータ[CR] を省略して表記します。

| コマンド | PUTSfxxyyssssss | |
|---|---|---|
| 機能 | 文字列の描画 | |
| 引数 | 値 | 内 容 |
| f | (1 桁 HEX ) | フォント番号( 0..6 ) |
| xx | (2 桁 HEX ) | X 座標ドット位置 |
| yy | (2 桁 HEX ) | Y 座標ドット位置 |
| ssssss | (任意長文字列) | ASCII または Shift-JIS 文字列 |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | "PUTS10C20Hello!" | (12,32)に 6x8 フォントで "Hello!"を表示 |
| 備考 | フォントについての詳細は 4.4 を参照してください。<br>座標は左上の位置とします。 | |

| コマンド | CLRS | |
|---|---|---|
| 機能 | 画面クリア | |
| 引数 | 値 | 内 容 |
| (なし) | | |
| | | |
| | | |
| | | |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | | |
| 備考 | 画面を全消去します。 | |

| コマンド | DOT1xxyy | |
|---|---|---|
| 機能 | １ドットの描画 | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | X 座標ドット位置 |
| yy | (2 桁 HEX ) | Y 座標ドット位置 |
| 応答 | | “OK”： 成功、”NGxx”： 失敗 |
| 使用例 | | |
| 備考 | 指定した(X,Y)座標位置に１ドットを描画する。 | |

| コマンド | DOT0xxyy | |
|---|---|---|
| 機能 | １ドットの消去 | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | X 座標ドット位置 |
| yy | (2 桁 HEX ) | Y 座標ドット位置 |
| 応答 | | “OK”： 成功、”NGxx”： 失敗 |
| 使用例 | | |
| 備考 | 指定した(X,Y)座標位置の１ドットを消去する。 | |

| コマンド | DOT8xxyy.....xxyy | |
|---|---|---|
| 機能 | 1 ドットの描画 | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | X 座標ドット位置 |
| yy | (2 桁 HEX ) | Y 座標ドット位置 |
| ・・・ | (2 桁 HEX ) | (上記 X, Y 座標の繰り返しを合計８回) |
| | | |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | | |
| 備考 | 8 ドット分をドットごとに指定した座標に連続描画する。 | |

| コマンド | LINExxyyuuvv | |
|---|---|---|
| 機能 | 始点から終点へラインを描画 | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | 始点 X 座標 |
| yy | (2 桁 HEX ) | 始点 Y 座標 |
| uu | (2 桁 HEX ) | 終点 X 座標 |
| vv | (2 桁 HEX ) | 終点 Y 座標 |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | "LINE40207F3F" | ( 64, 32 )から( 127,63 )に線を引く |
| 備考 | | |

| コマンド | CIRCxxyyrr | |
|---|---|---|
| 機能 | 円を描画 | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | 中心点 X 座標 |
| yy | (2 桁 HEX ) | 中心点 Y 座標 |
| rr | (2 桁 HEX ) | 半径 |
| | | |
| 応答 | | “OK”： 成功、”NGxx”： 失敗 |
| 使用例 | ″CIRC402018″ | ( 64, 32 )を中心に半径 24 の円を描画 |
| 備考 | | |

| コマンド | EBOXxxyywwhh | |
|---|---|---|
| 機能 | 矩形枠( Empty Box )の描画 | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | 左上 X 座標 |
| yy | (2 桁 HEX ) | 左上 Y 座標 |
| ww | (2 桁 HEX ) | 矩形の幅 |
| hh | (2 桁 HEX ) | 矩形の高さ |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | "EBOX321E3C14" | (50,30) から 60x20 の矩形枠を描画 |
| 備考 | | |

| コマンド | RBOXxxyywwhh | |
|---|---|---|
| 機能 | 角丸矩形枠( Round Box )の描画 | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | 左上 X 座標 |
| yy | (2 桁 HEX ) | 左上 Y 座標 |
| ww | (2 桁 HEX ) | 矩形の幅 |
| hh | (2 桁 HEX ) | 矩形の高さ |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | "RBOX321E3C14" | (50,30) から 60x20 の角丸矩形を描画 |
| 備考 | 角の丸い矩形枠を描画する。 | |

| コマンド | FILLxxyywwhh | |
|---|---|---|
| 機能 | 矩形の塗りつぶし | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | 左上 X 座標 |
| yy | (2 桁 HEX ) | 左上 Y 座標 |
| ww | (2 桁 HEX ) | 矩形の幅 |
| hh | (2 桁 HEX ) | 矩形の高さ |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | "FILL321E3C14" | (50,30) から 60x20 の矩形塗りつぶし |
| 備考 | | |

| コマンド | RFILxxyywwhh | |
|---|---|---|
| 機能 | 角丸矩形の塗りつぶし | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | 左上 X 座標 |
| yy | (2 桁 HEX ) | 左上 Y 座標 |
| ww | (2 桁 HEX ) | 矩形の幅 |
| hh | (2 桁 HEX ) | 矩形の高さ |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | "RFIL321E3C14" | (50,30) から 60x20 の角丸矩形塗りつぶし |
| 備考 | | |

| コマンド | ERASxxyywwhh | |
|---|---|---|
| 機能 | 矩形の消去（白塗りつぶし） | |
| 引数 | 値 | 内 容 |
| xx | (2 桁 HEX ) | 左上 X 座標 |
| yy | (2 桁 HEX ) | 左上 Y 座標 |
| ww | (2 桁 HEX ) | 矩形の幅 |
| hh | (2 桁 HEX ) | 矩形の高さ |
| 応答 | | “OK”： 成功、”NGxx”： 失敗 |
| 使用例 | ″ERAS321E3C14″ | (50,30) から 60x20 の矩形を白く塗りつぶす |
| 備考 | | |

| コマンド | KBENv | |
|---|---|---|
| 機能 | PS2 キーボードをイネーブルする | |
| 引数 | 値 | 内 容 |
| v | 1 桁数字 | 1:イネーブル、 0:禁止 |
| | | |
| 応答 | | “OK”： 成功、”NGxx”： 失敗 |
| 使用例 | KBEN1 | PS2 キーボードを使用可能にする |
| 備考 | キーボードが押されると″KBcc″を返す。<br>( cc は２桁 HEX のキーコード ) 4.5 参照 | |

| コマンド | CSUMmm | |
|---|---|---|
| 機能 | チェックサムモードを有効・無効にする | |
| 引数 | 値 | 内 容 |
| mm | 00 | チェックサムモード無効 |
| mm | 01 | チェックサムモード有効 |
| 応答 | | “OK”：成功、”NGxx”：失敗 |
| 使用例 | ″CSUM01″ | チェックサムを有効にする |
| 備考 | 電源投入後はチェックサム無効。 | |

・チェックサムを OFF から ON に変更する場合、この **CSUM** コマンドそのものには
チェックサムをつけてはいけません。このコマンドが解釈されてチェックサムモード
が確定した次のコマンドからチェックサム付きでコマンドを送ってください。
逆にチェックサムを ON から OFF に変更する場合は **CSUM** コマンドそのものにも
チェックサムが必要です。

**・チェックサムの計算法**

**CSUM** コマンドで「チェックサム有効」を指定した場合は、コマンド文字列の末尾
（ターミネータ［CR］の前）にチェックサムを追加します。
チェックサムはコマンド文字列の各文字のアスキー値を合計したものの下
８ビットを２桁のヘキサで表します。
例えばボードに送信するコマンドが ″CLRS″ の場合、各文字の ASCII 値を合計すると
C L R S
0x43 + 0x4c + 0x52 + 0x53 = 0x134 となるので送信する文字列は
チェクサムの２文字を追加して ″CLRS34[CR]″ となります。
（［CR］はターミネータの 0x0d を表す）

電源投入直後はチェックサムなしのモードでスタートします。
「チェックサム有効」を指定すると以後のコマンドにはすべてチェックサムが必要で
GT100 からの応答にもチェックサムの２文字が最後に付加されます。
**例）OK9A**

| コマンド | GPDIppvv | |
|---|---|---|
| 機能 | 汎用ポートの入出力方向を決める | |
| 引数 | 値 | 内 容 |
| pp | （２桁 HEX ） | ポート番号(00=PortA, 01=PortB, 02=PortC・・ ) |
| vv | （２桁 HEX ） | 入力にするビット=0、出力にするビット=1 |
| 応答 | | “OK”： 成功、”NGxx”： 失敗 |
| 使用例 | ″GPDI0085″ | ポート A のビット 7, 2, 0 を出力にする |
| 備考 | | |

| コマンド | GPOTppvv | |
|---|---|---|
| 機能 | 汎用ポートに出力する | |
| 引数 | 値 | 内 容 |
| pp | （２桁 HEX ） | ポート番号(00=PortA, 01=PortB, 02=PortC・・ ) |
| vv | （２桁 HEX ） | 出力する 8 ビット値 |
| 応答 | | “OK”： 成功、”NGxx”： 失敗 |
| 使用例 | ″GPOT0185″ | ポート B のビット 7, 2, 0 を ’1’ にする |
| 備考 | | |

| コマンド | GPINpp | |
|---|---|---|
| 機能 | 汎用ポートの値を読み込む | |
| 引数 | 値 | 内 容 |
| pp | （２桁 HEX ） | ポート番号(00=PortA, 01=PortB, 02=PortC・・ ) |
| | | |
| 応答 | GPINppvv | 例) ″GPIN0221″ ポート C のビット 5, 0 が’1’ |
| 使用例 | ″GPIN02″ | ポート C から入力 |
| 備考 | vv は読み込んだポートの値( 8bit ） | |

汎用入出力ポートの概要については **3.6 汎用入出力ポート** を参照してください。

| コマンド | SDOPmssssss | |
|---|---|---|
| 機能 | SD/MMC カードのファイルオープン | |
| 引数 | 値 | 内 容 |
| m | (1 桁 HEX ) | '0'=Read, 1=Write モード |
| sssss | (任意長 ASCII 文字列) | ファイル名 |
| | | |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | "SDOP1Test.txt" | ファイル"Test.txt" を書き込みモードで作成 |
| 備考 | 制限事項 :<br>ファイル名は最大 32 文字、同時にオープンできるファイルは 1 本 | |

| コマンド | SDCL | |
|---|---|---|
| 機能 | SD/MMC カードのファイルクローズ | |
| 引数 | 値 | 内 容 |
| | | |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | "SDCL" | 現在のファイルをクローズする |
| 備考 | | |

注) GT100 はカレンダークロックを持っていないため、作成されるファイルは
2009 年 1 月 1 日の日付けになります。

| コマンド | SDWRnnssssss | |
|---|---|---|
| 機能 | SD/MMC カードのファイル書き込み | |
| 引数 | 値 | 内 容 |
| nn | (2 桁 HEX ) | 書き込みバイト数 |
| sssss | (任意長 ASCII 文字列) | データ文字列 |
| | | |
| 応答 | | "OK" : 成功、" NGxx" : 失敗 |
| 使用例 | "SDWR05Hello" | "Hello" の文字を作成中のファイルに書き込む |
| 備考 | データ文字列に[CR] (キャリッジリターン) を含むことができる。<br>一度に送れる最大文字数は 256 文字(FF)とする | |

| コマンド | SDRDnn | |
|---|---|---|
| 機能 | SD/MMC カードのファイル書き込み | |
| 引数 | 値 | 内 容 |
| nn | (2 桁 HEX ) | 読み出し要求文字数( 最大 00 = 256 文字) |
| | | |
| 応答 | 読み出し文字列 | "SDRDmmssssss" : 成功、" NGxx" : 失敗 |
| 使用例 | "SDRD80" | ファイルから 128 文字読み出す |
| 備考 | 応答文字列の頭に "SDRDmm" ( mm は実際に読み出せたデータの文字数を 2 桁 HEX で表したもの) が付加される。<br>例) "SDRD0A1234567890"<br>文字数は "00" の場合は 最大値の 256 とみなされる。 | |

注) ファイルを最後まで読み切ると、次のアクセスで NG06 のエラー (Read Error) が返ります。

| コマンド | VERS | |
|---|---|---|
| 機能 | ファームウェアのバージョン取得 | |
| 引数 | 値 | 内 容 |
| | | |
| | | |
| 応答 | 文字列 | 4 文字のバージョン番号 |
| 使用例 | "VERS" | |
| 応答例 | "VERS1.21" | バージョン 1.21 |
| 備考 | | |

### 4.3.2 エラーコード一覧

コマンドにエラーがあった場合（モード設定で「応答なし」を設定いなければ）
"NGxx"（xx は下記のエラー番号）を返します。
同時にボード上の LED-2 が点灯します。(次の正常なコマンド終了で消灯します)

| エラー名 | エラー番号 | 内 容 |
|---|---|---|
| NOERROR | 00 | エラーなし（成功） |
| ERR_ILLEGAL_COMMAND | 01 | 存在しないコマンド |
| ERR_ILLEGAL_PARAM | 02 | 不正なパラメータ |
| ERR_ILLEGAL_FORMAT | 03 | 不正なフォーマット |
| ERR_CHECKSUM | 04 | チェックサムエラー |
| ERR_FILE_OPEN | 05 | オープンファイルエラー |
| ERR_FILE_READ | 06 | ファイルリードエラー |
| ERR_FILE_WRITE | 07 | ファイルライトエラー |
| ERR_TOOLONG_STRING | 08 | 文字列が長すぎ |

# 4.4　フォント

### 4.4.1　フォントの種類

GT100 には以下の３種類のフォントがあります。

**１．3x5 dot 数字フォント**

このフォントは数字(0..9)とアルファベット A～F の 6 文字のみです。
主にグラフの目盛りなど小さい数字が必要な用途が目的のフォントです。
フォントは内蔵 ROM に格納されていますので SD カードなしで表示できます。
また文字の表示位置はドット単位で指定可能です。

**2. 6x8 dot ASCII フォント**

数字とアルファベット（大文字、小文字、記号）から成るフォントです。
フォントは内蔵 ROM に格納されていますので SD カードなしで表示できます。
文字の表示位置はフォントサイズの倍数のドット位置に制約されます。

**3. 12x12 dot 漢字フォント**

漢字の文字列には SHIFT-JIS を用います。
フォントは SD カードから読み込まれるため、漢字を表示するには SD カードにフォントファイルが必要です。
また文字の表示位置はフォントサイズの倍数のドット位置（X 方向は半角サイズの倍数)に制約されます。

### 4.4.2　文字の表示と位置

GT100 に文字を表示するには PUTS コマンドを使います。
コマンドの形式は 4.3 にもありますが PUTSfxxyyssssssss[CR] となっており
f・・・・フォント番号（0..6）
xx・・・X 方向開始位置（2 桁へキサ）
yy・・・Y 方向開始位置（2 桁へキサ）
ssssss・・・表示する文字列( ASCII または SHIFT-JIS 漢字)

の各パラメータでフォントと表示位置を指定します。

フォント番号は以下の割り当てになっています。

| フォント番号 | サイズと種類 | 表示位置の制限 |
|---|---|---|
| 0 (4) | 3x5 dot 数字と A～F のみ | なし |
| 1 (5) | 6x8 dot ASCII | 横は 6dot の倍数に切り捨て<br>縦は 8dot の倍数に切り捨て |
| 2 (6) | 12x12 dot 漢字 | 横は 6dot の倍数に切り捨て<br>縦は 12dot の倍数に切り捨て |

フォント 1,2 では上記の通りに指定した表示位置が修正されます。
たとえばコマンドが

PUTS20F20 こんにちは[CR]

の場合、指定座標は (0FH, 20H ) ==> 十進で( 15, 32 ) ですが、
フォント番号 = 2 ですので、実際の表示開始座標は( 12, 24 ) になります。

表示可能な文字数は LCD のドット数が 128x64 ドットですので
6x8 ドット ASCII フォントの場合は 21 文字 x8 行
12x12 ドット全角漢字フォントの場合は 10.5 文字 x6 行
となります (イメージは **2.4** の表示サンプルを参照してください)
漢字を含む文字列は必ず SHIFT-JIS で送ってください。

### 4.4.3 文字の反転表示

フォント番号が 0, 1, 2 のとき白バックに黒文字で表示されます。
フォント番号に「プラス 4」すると( フォント番号 4, 5, 6)黒バックに白抜きの
文字で描画されます。

# 4.5　PS2 キーボードからの入力

GT100 を立ち上げた直後は PS2 からのキーボード入力 *1)は禁止されています。
"KBEN1[CR]"を GT100 に送ることでキーボード入力がイネーブルされ、それ
以降、押されたキーのキーコード *2)が KBcc の形式で返されます。
cc は押されたキーのキーコードを２桁のヘキサ値で表したものです。
たとえば 'A' のキーを押すとキーコード 0x1C が返され

KB1C[CR]

キーを離すと 0xF0 が返されます。

KBF0[CR]

注 *1) PS2 入力はキーボードのみの対応です。　マウスには対応しません。

注 *2) キーコードは ASCII コードとは異なります。

下の表は当社調べのキーコードの一例（Microsoft 社仕様）ですが、
キーボードによってはこれと異なるケースがありますので、実際にキーを
押したときに返ってくるコードをユーザー様にてご確認ください。
6.1　テストモード のスイッチ操作で簡単に調べることができます。

| キー名 | 押した時のコード | 離した時のコード |
|---|---|---|
| SHIFT | 12 | F0 + 押した時のコード |
| CTRL | 14 | （以下同様） |
| ALT | 11 | |
| TAB | 0D | |
| SPACE | 29 | |
| 0 | 45 | |
| 1 | 16 | |
| 2 | 1E | |
| 3 | 26 | |
| 4 | 25 | |
| 5 | 2E | |
| 6 | 36 | |
| 7 | 3D | |
| 8 | 3E | |

| | | |
|---|---|---|
| 9 | 46 | |
| A | 1C | |
| B | 32 | |
| C | 21 | |
| D | 23 | |
| E | 24 | |
| F | 2B | |
| G | 34 | |
| H | 33 | |
| I | 43 | |
| J | 3B | |
| K | 42 | |
| L | 4B | |
| M | 3A | |
| N | 31 | |
| O | 44 | |
| P | 4D | |
| Q | 15 | |
| R | 2D | |
| S | 1B | |
| T | 2C | |
| U | 3C | |
| V | 2A | |
| W | 1D | |
| X | 22 | |
| Y | 35 | |
| Z | 1A | |
| | | |

# 5．動作テストとサンプルプログラム

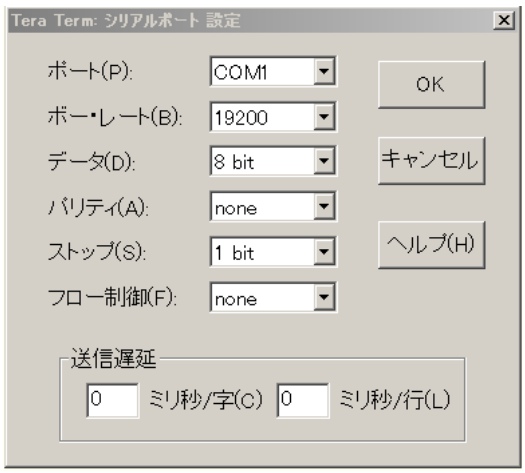
## 5.1 ターミナルソフトからの描画テスト

ターミナルソフトを使ってパソコンの RS232C から GT100 にコマンドを送り、
各種描画機能をテストします。

Windows のターミナルソフト（ ハイパーターミナル、 Tera Term *1)など）から
コマンドを送って GT100 の動作をテストしてみます。

### 5.1.1　ターミナルソフトの設定

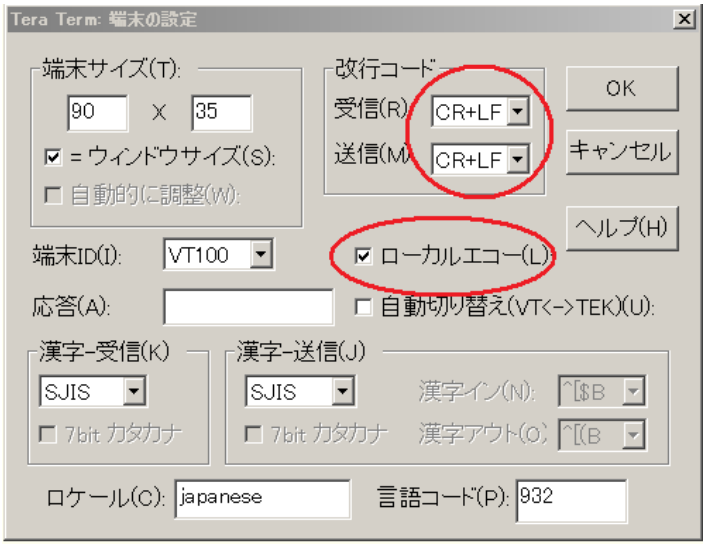
・付属のオス・メス　ストレートケーブルで PC と GT100 を接続してください。
RS-232C 端子のないパソコンの場合、USB の仮想 COM ポートを使います。
USB－RS-232C 変換ケーブルで変換してください。(変換ケーブルは付属しておりません)
・ターミナルソフトを立ち上げ、ボーレートと諸元を 19200 、8N1 、フロー制御なし、
に設定します。
ポートは実際に接続している COM ポートに合わせてください。
以下は TeraTerm の場合の設定画面例です。(″設定″－″シリアルポート″メニュー)




*1) Windows Vista 以降の OS ではハイパーターミナルが標準装備されていないので VECTOR,
窓の杜などで TeraTerm などを入手してお使いください。

TeraTerm　は TeraTerm Project により作成されたターミナルエミュレータで 2012 年 2 月現在もフリーソフトとして公開されているようです。

・打ち込んだ文字列がフィードバック表示されるよう、[設定]－[端末]　メニューでローカルエコーを ON にし、送信コードを CR+LF にします。




送信コードが CR だけの場合、[CR]（キャリッジリターン）キーを打ってもカーソルが先頭に戻るだけで改行がおこなわれません。

### 5.1.2　GT100 本体の設定

ボード上のディップスイッチをすべて OFF にしてください。

19200 以外のボーレートを使う場合は 4.1 を参照して設定してください。

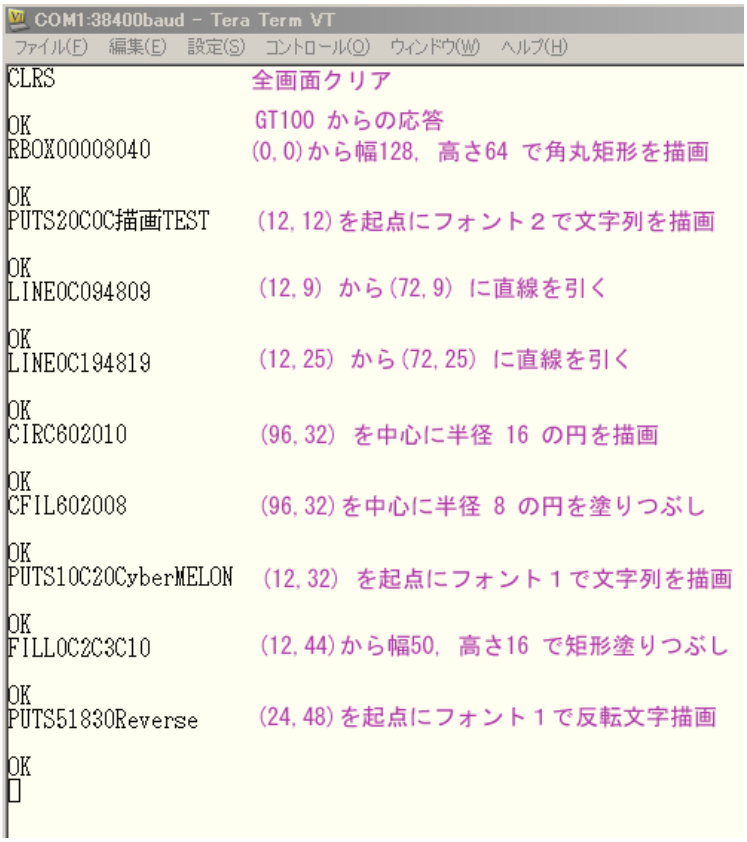
以下のサンプルは 38400 baud での実行結果です。

DIP SW-4 を OFF にした場合は電源を入れてもデモ画面は表示されません。

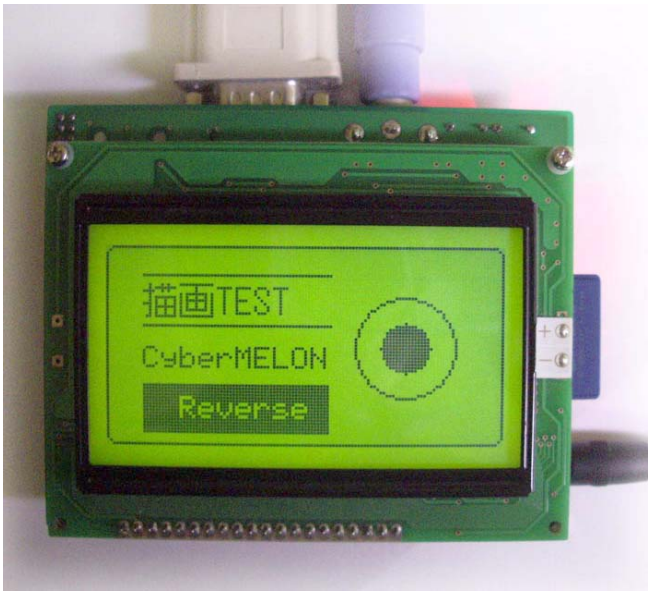
### 5.1.3　コマンドの実行

準備ができましたらターミナルソフトにコマンドを打ち込んでみます。

**各コマンドの詳細は 4.3 を参照してください。**




結果はこのようになります。



# 5.2　C言語によるマイコンからのGT100制御

### 5.2.1　接続

GT100 ボードのRS-232C コネクタは2番ピンが Tx(送信)、3番ピンがRx(受信)の DCE ですので、ホスト（制御マイコンなど）側も同じ 構成の場合はクロスケーブルが必要です。

付属のオス・メス９ピンケーブルは「ストレート」タイプですのでご注意ください。

### 5.2.2　コマンド送信手続き

コマンドを送るＣ言語サンプルソースを示します。　ボーレートやパリティーの設定などSCI の初期化は既にされているものとします。

```
//-------------------------------------------------------------
//    RS-232C へ１文字送信する
//    Input:  c      送信する文字
//-------------------------------------------------------------
void putc( char c )
{
    // CPU によって異なるので省略
}

//-------------------------------------------------------------
//    文字列を送信する
//    Input:   str        '0' ターミネート文字列
//-------------------------------------------------------------
void puts( char* str )
{
    char  *p;
    p = str;
    while( *p )
    {
        putc( *p++ );
    }
}
```

```
//-------------------------------------------------------
//    BYTE の数値を２桁へキサ文字列に変換する
//    Input:  buf     文字列受け取り用バッファ(3 文字以上)
//            num    数値
//    Note:   sprintf( buf, "%02X", num )  相当
//-------------------------------------------------------
void format_hex2( char* buf, unsigned char num )
{
    char    c;
    c = ( num >> 4 ) & 0x0f;
    if ( c <= 9 )
            c += '0';
    else
            c += ('A' - 10 );
    buf[0] = c;
    c = num & 0x0f;
    if ( c <= 9 )
            c += '0';
    else
            c += ('A' - 10 );
    buf[1] = c;
    buf[2] = 0;
}
//-------------------------------------------------------
//    WORD の数値を 4 桁 へキサ文字列に変換する
//    Input:  buf     文字列受け取り用バッファ(5 文字以上)
//            num    数値
//    Note:   sprintf( buf, "%02X", num ) 相当
//-------------------------------------------------------
void format_hex4( char* buf, unsigned short num )
{
    char    t[4];
    format_hex2 ( buf, ( num >> 8 ) & 0x00ff );
    format_hex2 ( t, num & 0x00ff );
    strcat( buf, t );
```

```
}
//-------------------------------------------------------------
//    コマンド文字列を送信する
//    Input:   command      コマンド文字列
//    Note:  command に（必要なら）チェックサム、ターミネータ
//           [CR] を付加して送信する
//           元の command バッファに追加分の余裕があること
//-------------------------------------------------------------
void SendCommand( char* command )
{
    unsigned short   i, len, sum;
    char    t[6];
    // bCheckSumMode はどこかで設定されているとして
    if (bCheckSumMode )     // チェックサムモードでなければ以下は無視
    {
        len = strlen( command );
        sum = 0;
        for( i = 0; i < len; i++ )
        {
            sum += (unsigned short)command[i];
        }
        format_hex2( t, sum & 0x00ff );
        strcat( command, t );       // チェックサムを付加
    }
    len = strlen( command );
    command[len] = 0x0d;            // ターミネータ[CR] を付加
    command[len+1] = 0;
    puts( command );                // 送信
 }
```

### 5.2.3 応答の受信

実際にはRTOS のメッセージ通知などを使うことが多いと思いますが、ここでは簡単にするためポーリングでの受信サンプルを示します。

```
//-------------------------------------------------------
//    RS-232C の受信チェック
//    Return:  1:受信文字あり、 0:受信文字なし
//-------------------------------------------------------
unsigned char is_ready()
{
    // CPU によって異なるので省略
    if ( ready )
        return  1;
    else
        return  0;
}
//-------------------------------------------------------
//    RS-232C の１文字受信
//    Return:   受信文字コード
//-------------------------------------------------------
unsigned char getc()
{
    unsigned char   c;
    // CPU によって異なるので省略
    return  c;
}
//-------------------------------------------------------
//    文字列受信
//    Return:  0 : 受信文字列なし、1: 受信文字列あり
//-------------------------------------------------------
static unsigned char rxbuf[32];
unsigned char gets()
{
    int  i, retry;
    char    c;
```

```
    for( i = 0; i < 31; i++ )
    {
        retry = 0;
        while( ++retry < 5000 )
        {
            if ( is_ready())
            {
                c = getc();
                break;
            }
            sleep( 1 );             // 1msec 待つ
        }
        if ( retry >= 5000 )
            return  0;              // 5秒タイムアウト
        rxbuf[i] = c;               // 受信バッファに1文字追加
        if ( c == 0x0d )
        {
            rxbuf[i+1] = 0;
            return  1;              // [CR]ターミネータを検出
        }
    }
    return  0;      // Rx バッファ・オーバーフロー
}
//-------------------------------------------------------
//    応答の受信
//    Return:  0 : 受信エラー、1: 受信OK
//-------------------------------------------------------
unsigned char GetReply()
{
    if ( gets() == 0 )
        return    0;        // タイムアウトかオーバーフロー
    if ( rxbuf[0] == 'O' && rxbuf[1] == 'K' )
        return    1;        // OK
    return    0;            // その他のエラー
}
```

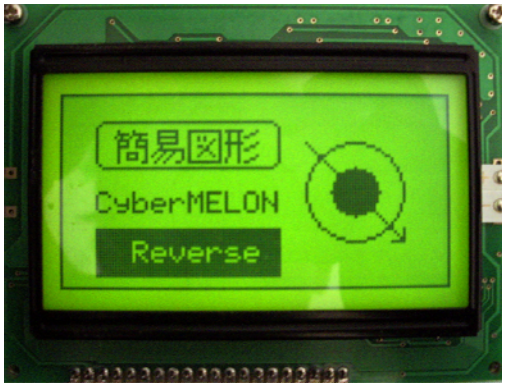
### 5.2.4 簡単な描画サンプル

```
//-------------------------------------------------------
//    文字と図形の簡単な描画サンプル    (メイン)
//    Note: 実際には各 SendCommand のあとに GetReply で応答を
//          待ってから次のコマンドを送る必要がありますが
//          流れを見やすくする為に省略してあります。
//-------------------------------------------------------
main()
{
    char    cmd[32], t[6];
    //  SCI (RS232C)の初期化
    init_sci();
    //  全画面消去
    SendCommand("CLRS");    // 背景色で塗りつぶし

    // 画面全体を枠で囲む
    sprintf( cmd, "EBOX%02X%02X%02X%02X", 0, 0, 128, 64 );
    /*-----------------------------------------------
    // sprintf が使えない時は以下のルーチンで代用
      strcpy( cmd, "RBOX" );
      format_hex2( t, 0 );          // 2桁ヘキサに( sprintf  の代用 )
      strcat( cmd, t );
      format_hex2( t, 0 );
      strcat( cmd, t );
      format_hex2( t, 128 );
      strcat( cmd, t );
      format_hex2( t, 64 );
      strcat( cmd, t );
    -----------------------------------------------*/
    SendCommand( cmd );
    // 角丸矩形を  始点=(12,9)   サイズ=(60,16) で描画
    sprintf( cmd, "RBOX%02X%02X%02X%02X", 12, 9, 60, 16);
    SendCommand( cmd );
    // 全角文字を (18,12)を起点に描画
    sprintf( cmd, "PUTS2%02X%02X 簡易図形", 18, 12 );
```

```
    SendCommand( cmd );
    // 円の描画    中心=(96, 32) 半径=16
    sprintf( cmd, "CIRC%02X%02X%02X%02X", 96, 32, 16 );
    SendCommand( cmd );
    // 円の塗りつぶし    中心=(96, 32) 半径=8
    sprintf( cmd, "CFIL%02X%02X%02X%02X", 96, 32, 8 );
    SendCommand( cmd );
    //  半角文字で"CyberMELON" を (12,32)を起点に描画
    sprintf( cmd, "PUTS1%02X%02XCyberMELON", 12, 32 );
    SendCommand( cmd );
    //  矩形領域を黒塗り  (12,44)を起点に幅 60, 高さ 16
    sprintf( cmd, "FILL%02X%02X%02X %02X ", 12, 44, 60, 16 );
    SendCommand( cmd );
    // 反転半角文字の描画
    sprintf( cmd, "PUTS5%02X%02XReverse", 24, 48 );
    SendCommand( cmd );
    // 斜め直線の描画    始点(80,16) 終点(112,48)
    sprintf( cmd, "LINE%02X%02X%02X%02X", 80, 16, 112, 48 );
    SendCommand( cmd );
    // 矢印の先
    sprintf( cmd, "LINE%02X%02X%02X%02X", 108, 48, 112, 48 );
    SendCommand( cmd );
    sprintf( cmd, "LINE%02X%02X%02X%02X", 112, 44, 112, 48 );
    SendCommand( cmd );
}
```

上記プログラムの実行結果は 以下のようになります。



### 5.2.5 デモ画面 y = x * sin(x) のグラフ 描画

```
//-------------------------------------------------------
//    文字列の描画
//    Input:  pstr          文字列( '0'終端 )
//            x, y          ドット座標
//-------------------------------------------------------
void glcdPutString( char* pstr, int x, int y )
{
    char    cmd[32];
    sprintf( cmd, "PUTS1%02X%02X", x, y );
    strcat( cmd, pstr );
    SendCommand( cmd );
}

void glcdPutSmallString ( char* pstr, int x, int y )
{
    char    cmd[32];
    sprintf( cmd, "PUTS0%02X%02X", x, y );
    strcat( cmd, pstr );
    SendCommand( cmd );
}

//-------------------------------------------------------
//    直線の描画
//    Input:  sx, sy        始点ドット座標
//            ex, ey        終点ドット座標
//-------------------------------------------------------
void glcdDrawLine( int sx, int sy, int ex, int ey )
{
    char    cmd[16];
    sprintf( cmd, "LINE%02X%02X%02X %02X ", sx, sy, ex, ey );
    SendCommand( cmd );
}
```

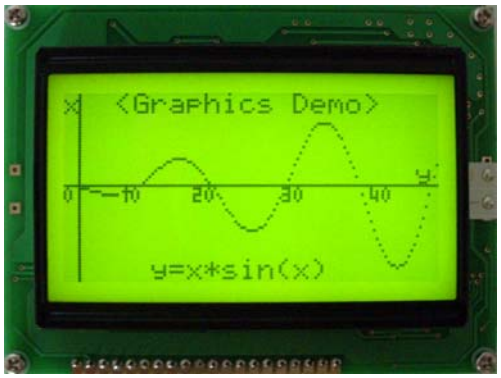
```
//-------------------------------------------------------
//    点の描画
//    Input:  x,sy          ドット座標
//-------------------------------------------------------
void glcdPutDot( int x, int y )
{
    char    cmd[16];
    sprintf( cmd, "DOT1%02X%02X ", x, y );
    SendCommand( cmd );
}


#define     M_PI    3.141592
//-------------------------------------------------------
//    y = x * sin(x) のグラフ    (メイン)
//
//    Note: 実際には各 SendCommand のあとに GetReply で応答を
//          待ってから次のコマンドを送る必要がありますが
//          流れを見やすくする為に省略してあります。
//-------------------------------------------------------
main()
{
    int     i, ix, iy;
    float   fx, fy;

    SendCommand("CLRS");            // 画面消去
    glcdPutString( "<Graphics Demo>", 18, 0 );
    glcdPutString( "y=x*sin(x)", 30, 56 );
    glcdPutString ( "x", 0, 0 );
    glcdPutString ( "y", 120, 24 );
    //      座標軸
    glcdDrawLine( 0, 31, 127, 31 );
    glcdDrawLine( 5, 0, 5, 63 );
```

```
    // y = x * sin(x) のグラフ
    for( ix = 0; ix < 128; ix++ )
    {
            fx = ( 2 * M_PI / 50) * (float)ix;
            fy = 50.0 / ( 8 * M_PI ) * fx * sin( fx );
            glcdPutDot( ix, (int)fy + 31 );
    }
    glcdPutSmallString( "0", 0, 32 );
    glcdPutSmallString ( "10", 19, 32 );
    glcdPutSmallString ( "20", 44, 32 );
    glcdPutSmallString ( "30", 74, 32 );
    glcdPutSmallString ( "40", 104, 32 );
}
```

上記プログラムの実行結果は以下のようになります。



### 5.2.6　デモ画面　ボタンによる選択メニュー

```
//-------------------------------------------------------
//    文字列の描画
//    Input:  pstr          文字列( '0'終端 )
//            x, y          ドット座標
//-------------------------------------------------------
void glcdPutKanjiString( char* pStr, int x, int y )
{
    char    cmd[32];
    sprintf( cmd, "PUTS2%02X%02X", x, y ); // フォント=2 を指定
    strcat( cmd, pstr );
    SendCommand( cmd );
}
//-------------------------------------------------------
//    矩形領域の消去（白塗り）
//    Input:  x, y       ドット座標
//            w, h       幅と高さ
//-------------------------------------------------------
void glcdEraseRect( int x, int y, int w, int h )
{
    char    cmd[16];
    sprintf( cmd, "ERAS%02X%02X%02X %02X", x, y, w, h );
    SendCommand( cmd );
}
//-------------------------------------------------------
//    円の描画
//    Input:  x, y      中心ドット座標
//            r         半径
//-------------------------------------------------------
void glcdDrawCircle( int x, int y, int r )
{
    char    cmd[16];
    sprintf( cmd, "CIRC%02X%02X%02X", x, y, r );
    SendCommand( cmd );
}
```

```
//-------------------------------------------------------
//    円の塗り潰し
//    Input:  x, y      中心ドット座標
//            r         半径
//-------------------------------------------------------
void glcdFillCircle ( int x, int y, int r )
{
    char    cmd[16];
    sprintf( cmd, "CFIL%02X%02X%02X", x, y, r );
    SendCommand( cmd );
}
```

```
//-------------------------------------------------------
//    選択メニューとボタンの移動描画
//
//    Note: 実際には各 SendCommand のあとに GetReply で応答を
//          待ってから次のコマンドを送る必要がありますが
//          流れを見やすくする為に省略してあります。
//-------------------------------------------------------
main()
{
    int  i, j;
    SendCommand("CLRS");            // 画面消去
    glcdClearBackgroundBuf( 0 );
    glcdPutKanjiString ( " ファイルを開く", 12, 0, 0 );
    glcdPutKanjiString ( " ファイルを保存", 12, 12, 0 );
    glcdPutKanjiString ( " 動作の取り消し", 12, 24, 0 );
    glcdPutKanjiString ( " 先頭に戻る", 12, 36, 0 );
    glcdPutKanjiString ( " キャンセル", 12, 48, 0 );
```

```
for( i = 0; i < 5; i++ )
{
    for( j = 0; j < 5; j++ )
    {
        glcdEraseRect( 0, j*12, 12, 12 );
        glcdDrawCircle( 6, j*12+6, 4 );
    }
    glcdFillCircle( 6, i*12+6, 2 );
    //  0.5秒待ってボタンを下に移動させる
    sleep( 500 );                   // 時間待ち関数
}
```

上記プログラムの実行結果は以下のようになります。

# 5.3　SD/MMC カードの読み書き

## 5.3.1　SD/MMC カードへのファイル書き込み

(1) ファイル名を指定してファイルを書き込みモードでオープンする。

例) "SDOP1Test.txt"

4 文字のコマンドに続く'1' は書き込みモードを指定し、続く"Test.txt" がファイル名になります。

(2) 何か書き込む。

例) "SDWR0A1234567890"

この例では "1234567890" の 10 文字( 0A )が書き込まれます。

(3) 必要なだけ書き込みを繰り返す。　データに[CR] キャリッジリターンなど制御コードが含まれていても問題ありません。 SDWR に続いて文字数が送られるので GT100 ではそれを見て電文の終わりの位置を計算します。

(4) ファイルをクローズする。

例) "SDCL"

書き込みファイルは最後に必ずクローズ処理をしないと作成できません。

注) GT100 のファイルシステムは Windows 互換ですが、FAT32 のみをサポートします。　カードは必ず FAT32 でフォーマットしてご利用ください。

## 5.3.2　SD/MMC カードからのファイル読み込み

(1) ファイル名を指定してファイルを読み込みモードでオープンする。

例) "SDOP0Test.txt"

4 文字のコマンドに続く'0' は書き込みモードを指定し、続く"Test.txt" がファイル名になります。

(2) バイト数を指定して読み出しをおこなう。

例) "SDRD0A"

この例では 10 バイト( 0A )のデータを要求しています。

GT100 では応答として以下のデータを返します。

"SDRD0A1234567890"

"SDRD" に続く 2 桁の数値 ( 0A ) は実際に読み出すことのできたバイト数になります。

(3) 必要なだけ読み出しを繰り返す。

途中でファイルの最後に到達した場合は、要求のバイト数より小さい値が返されて、それ以後の SDRD コマンドでは NG06 (ファイル読み出しエラー)

が返されます。

### 5.3.3 制限事項

GT100 の MMC/SD カードはカードの抜き差し検出をおこなっておりません。
従いましてホットプラグ（電源投入中のカードの抜き差し）には対応できません。
カードを差し替えた場合はリセットボタンを押すか、電源を入れ直してください。

# 6. その他

## 6.1 テストモード

タクトスイッチ SW1（緑）と SW2（青）は社内テスト用です。

・SW1 を２～３秒押した状態のままでリセットボタンを押すか、電源を入れ直すとポートチェックモードに入ります。
ポートチェックモードでは 0.1 秒ごとに各ポートのビットを順に ON/OFF します。
SW1 と同時に SW2 を押すと間隔が 0.5 秒になります。

・SW2 のみを押しながらリセットボタンを押した場合はキーボードテストモードになります。
キーボードテストモードでは PS2 キーボードから送られてくるコードをヘキサで 液晶と RS232C 端末に表示します。

・テストモードから抜け出すにはリセットボタンを押すか、電源を入れ直してください。

## 6.2 特注仕様

・本ボードを利用した特注仕様ソフトウェアなどの開発請負も可能です。

詳細は当社ホームページ http://www.cyber-melon.com お問い合わせコーナーよりご連絡ください。

本書の改訂版は当社ホームページの該当製品コーナーよりダウンロードしてください。

Cyber MELON

株式会社インターネット

〒665-0841

兵庫県宝塚市御殿山 2-25-39

http://www.cyber-melon.com

e-mail: info#cyber-melon.com

（ # を @ に置き換えてください ）