ユーザーズマニュアル

Intelligent Graphic Terminal Kit

GT320

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	2011/ 9/ 1	初版リリース
1.01	2011/10/14	LODI, SAVI コマンドの引数間違い修正
1.03	2011/10/21	DOT8, LINE, CIRC コマンド追加、 TPMD07 拡張
1.04	2012/ 1/25	RFIL, CFIL コマンド追加
1.05	2012/ 2/ 9	232C コネクタ記述追加、付属品 SD カード追加

☆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

1. はじめに

このたびはサイバーメロン GT320 をお買いあげいただきまして、誠に有り難うございます。
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. 本製品または付属プログラム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. 本製品に付属のソフトウェア・ライブラリおよび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。
   ソフトウェアの最新版はユーザー登録後、当社ホームページからダウンロードしていただけます。
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。(9 ページ参照)
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　お問い合わせ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. GT320 ボード本体　1 

2. 6.5 インチカラー液晶モジュール　1 

3. 液晶用 FCC ケーブル　1 

4.　インバータ　1 

5. 説明書 CD-ROM　1 

6. DC12V 電源アダプター　1 

7. RS232C ケーブル　1 

8. SD カード　1 

動作チェック

・本ボードの動作および動作チェックには以下の環境が必要です。

Windows パソコン

OS: WindowsXP 以上

RS232C の空きポートまたは仮想 COM ポートの場合は USB-RS232C 変換ケーブル(これは本製品に含まれません)

・**3.7　接続と動作確認**　の手順に従って液晶モジュールとインバータなどの配線と初期設定をしてください。　電源は必ず付属の 12V DC 電源をお使いください。

・付属の電源アダプターを GT-320 本体に接続して通電してください。

液晶のバックライト点灯して電源が入ったことを示します。

数秒して以下のデモ画面が表示されます。

SD/MMC カードにデータがない場合は左のデモ画面（右下の画像なし）のみになります。

・ RS232C ケーブルをパソコンと接続してください。

. **5.2** にしたがってパソコンから RS232C でコマンドを送るテストによって本機の各種機能を学習してください。

・フレームメモリーとカラーパレットの概念については予備知識として 3.5　3.6 をあらかじめお読みいただいた方が理解が早いと思います。

・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

GT320 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、ビットマップ画像などの表示をおこなう QVGA グラフィック端末で液晶パネル＋制御ボード＋インバータ＋電源のセットです。
液晶パネルは明るく発色のきれいな NEC 製の 6.5 インチ TFT カラー液晶モジュールを使用しています。
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。
試作や小ロットの組み込みに最適です。

*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

GT320 は以下の機能と特長を有しています。

- ◆３種類の文字フォントによる漢字を含む文字列の描画。
- ◆ 前景・背景色の描画ごとの設定（256 色、または 64 階調グレースケールの固定パレット）
- ◆ ビットマップ画像の画面全体または一部への表示。また描画画面の一部のビットマップ保存と読み込み。
- ◆ 直線や矩形、塗りつぶしなどの描画。
- ◆ バーグラフの高速描画（メーターなど)。
- ◆ オプションのタッチパネル入力対応 （クリック座標を取得します)。
- ◆ 汎用ポート制御（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)。詳細は 3.10 を参照してください。
- ◆ VB, C 言語によるサンプルソースプログラムの提供により組み込み機器やパソコンから GT320 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 4.2.6 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。

・OA 機器の入出力端末
・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
・博物館、イベント会場などのデータベース端末や案内システム
・通信システムの端末
・各種アミューズメント機器
・教育システム

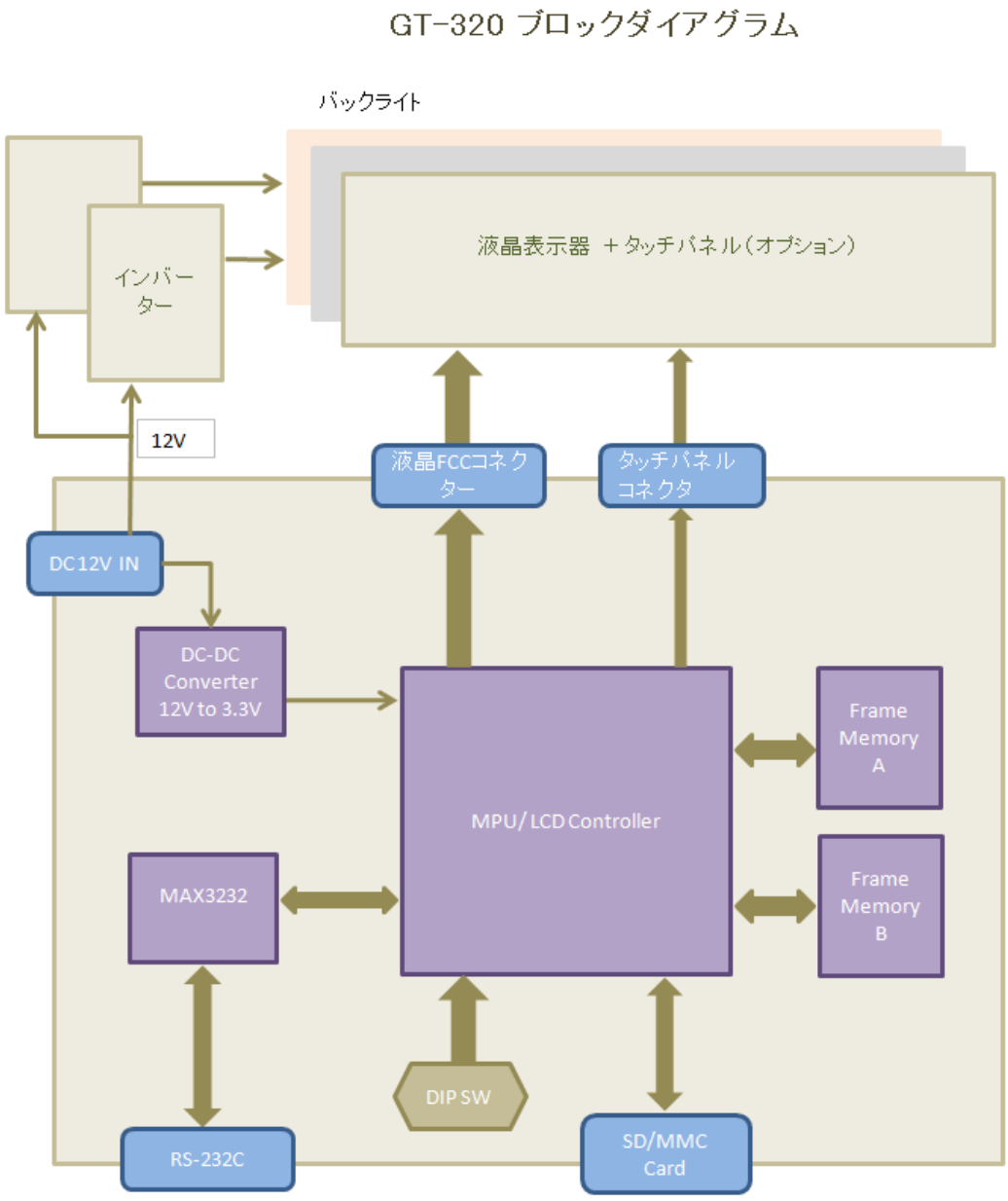
2.3　仕様

項 目	仕　様
CPU	PIC32MX360F512L 動作クロック 75.6MHz、ペリフェラル 18.9MHz
液晶コントローラ	Altera EPM570T144C5（ロジック自社開発 ）
フレームメモリー	128kB x ２面
電源電圧	DC12V 入力　　（ボード内部動作 3.3V ）
電圧出力	DC12V インバータ用出力
入出力	RS232C コネクター
	SD/MMC カードソケット
	DIP スイッチ（４極）
	タッチパネル（オプション）
	汎用出力ポート 8 ビット、入力ポート 6 ビット
液晶パネル	6.5 インチ カラーTFT　　(NEC NL6448BC20)
液晶解像度	QVGA (320 x 240 ピクセル)　*1)
液晶カラー解像度	6 ビット固定パレット 64 色、または 64 階調グレースケール
BL 明るさ	300cd/m2　Typ.
付属 FCC ケーブル	0.5mm ピッチ 36 芯 （基板側コネクタ： Hirose　FH12-36S-0.5SH　）
対応タッチパネル （オプション）	日本開閉器　FT-AS00-6.5AS-4（4 線アナログ方式）
RS-232C コネクタ	D-SUB 9 ピン 基板側メス　インチネジ
タッチパネル コネクタ	CN-2　日圧 S4B-XH-A
BL 電源出力 コネクタ	CN-7　日圧 S3B-XH-A
ボード寸法	100(W) x 104(D)x 22(H) mm（突起を含む）
液晶パネル寸法	179(W) x 127(D)x 12(H) mm
インバータ寸法	150(W) x　36(D)x 11(H) mm
消費電流	1.0A（@12V）（液晶バックライト含む）

*1)　液晶パネル自体は VGA (640x480)のものを半分の解像度で使用しています。

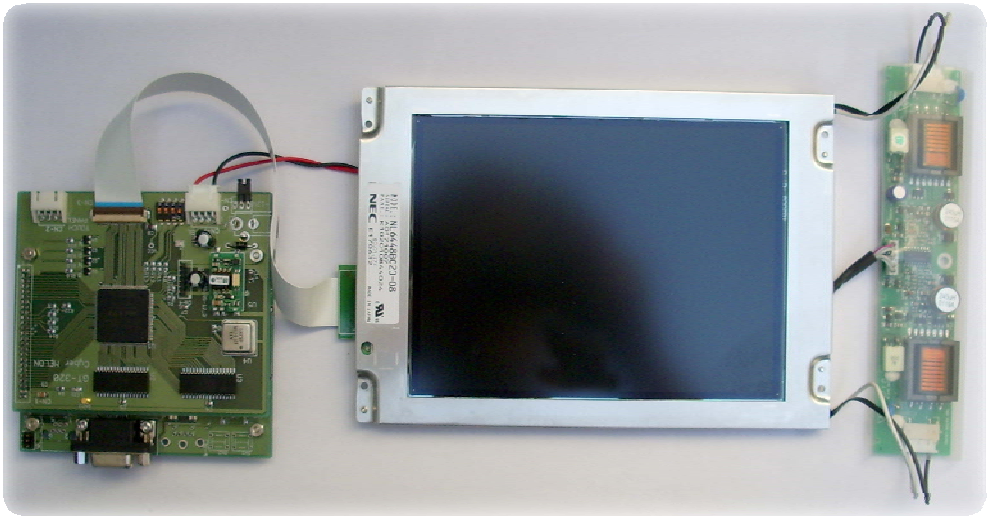
3．ハードウェア

3.1 ブロック図



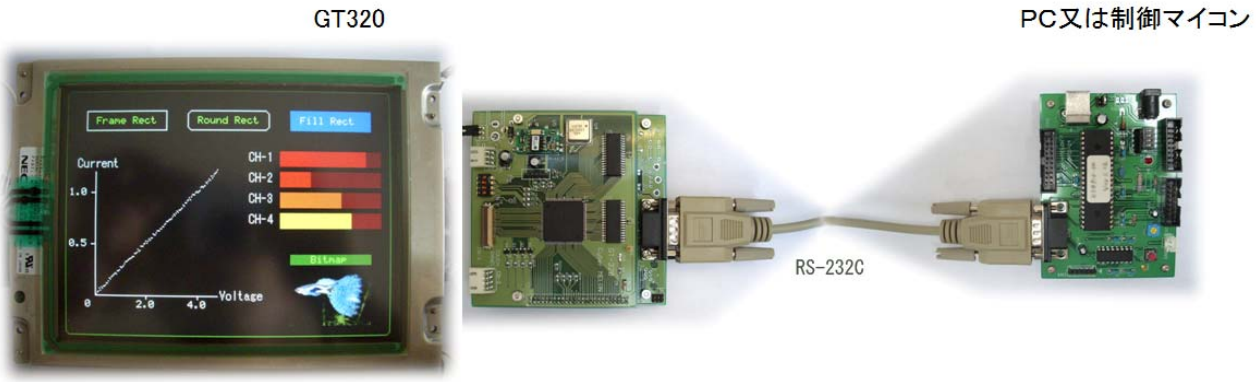
3.2 外観・接続図

GT320 の外観



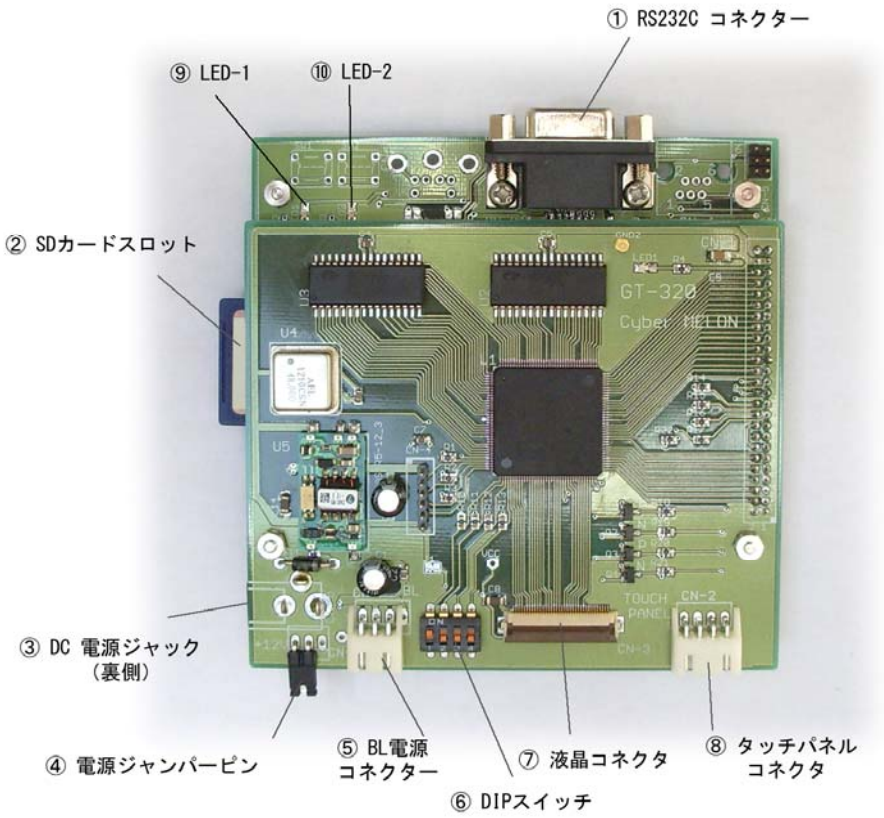
（右側のユニットがバックライト点灯用のインバータです。）

ホストとの接続図



*）RS-232C は付属の 9 ピンストレートケーブルの場合、GT320 側がメス、ホスト側がオスになります。
もし メス-メスの場合はクロスケーブル（オス-オス）を別途お求めください。

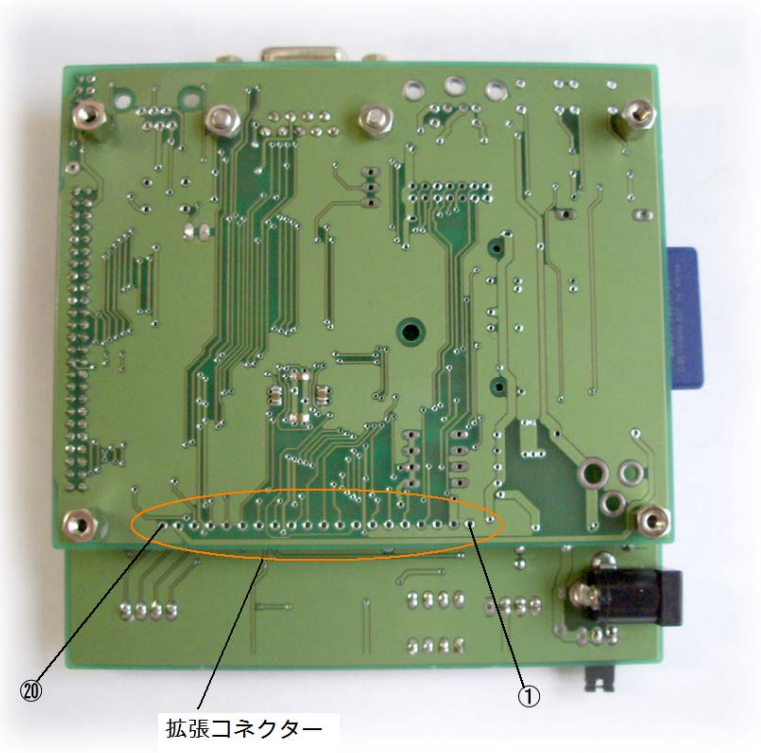
3.3 ボードの各部名称と機能



① RS232C コネクター	D-SUB 9 ピンケーブルでホストと接続します。
② SD カードスロット	SD/MMC カードを挿入します。
③ DC 電源ジャック	付属の 12V スイッチング電源 を接続します。
④ 電源ジャンパーピン	12V 電源をスルーします。 電源スイッチを追加したい場合はプラグをはずしてスイッチを挿入してください。（詳細は **3.8** を参照してください）
⑤ BL 電源コネクター	バックライト用の DC 12V をインバータに供給します。
⑥ DIP スイッチ	RS232C のボーレートや動作モードを指定します。ボーレートの設定は **4.1** 、動作モードは **3.7.4** を参照してください。
⑦ 液晶コネクタ	FCC ケーブルで液晶モジュールと接続します。
⑧ タッチパネルコネクター	タッチパネルのケーブルを接続します。

⑨ LED-1	電源投入時 LED-2 と共に２回点滅して初期化完了を示し、その後２秒周期で点滅してボードが正常に動作していることを示します。
⑩ LED-2	RS232C から受け取ったコマンドがエラーのとき点灯し、次の正常なコマンド終了で消灯します。

ボード裏面



拡張コネクター部（汎用入出力ポート）にはコネクターが実装されておりません。
汎用入出力ポート(GP)をご利用の場合は、2.54mm ピッチのコネクターを付けるか、
直接ボードから線をハンダ付けで引き出してください。
ピンアサインは次章を参照してください。

3.4 コネクター・ピン配置

CN-3　RS232C コネクター

PIN	機 能	PIN	機 能
1	DCD(IN)	2	TxD(OUT)
3	RxD(IN)	4	DSR(IN)
5	GND	6	DTR(OUT)
7	CTS(IN)	8	RTS(OUT)
9	RI		

注）ホストとの接続は付属の 9 ピン オス・メス ストレートケーブルをお使いください。

CN-7　ＢＬ電源コネクター

PIN	機 能	接続先
1	GND	インバータ
2	DC12V OUT	インバータ
3	GND	

CN-2　タッチパネルコネクター

PIN	機 能	接続先
1	YUP	タッチパネル
2	YLO	タッチパネル
3	XLE	タッチパネル
4	XRI	タッチパネル

CN-9　拡張コネクター（ボード裏面）

PIN	方向(R/W)	機　能
1		GND
2		
3		
4	Read	GPIN-0
5	Read	GPIN-1
6	Read	GPIN-2
7	Write	GPOUT-0
8	Write	GPOUT-1
9	Write	GPOUT-2
10	Write	GPOUT-3
11	Write	GPOUT-4
12	Write	GPOUT-5
13	Write	GPOUT-6
14	Write	GPOUT-7
15	Read	GPIN-4
16	Read	GPIN-5
17	Read	GPIN-3
18		
19		
20		GND

汎用の入出力(GPIN/OUT) ポートです。　コマンドで制御できます。

方向は制御ホスト(PC やマイコン) から見た方向です。

ピン配置は 3.3 ボード裏面写真を参照してください。

ポートの絶対最大定格

項目	値	条件
最大出力シンク電流	25mA	注)
最大出力ソース電流	25mA	注)
最大入力電圧	-0.3～5.5V	0～5.0V でお使いください

注) 上記の電流値は CMOS 入力電圧レベルを保証できる電流値ではありません。

3.5 フレームメモリー

GT320 は描画用のバッファ（フレームメモリー）を２面持っています。
一面を表示中、他方の一面に書き込むことができます。(仮にＡ面、Ｂ面と呼びます)
Ａ面を表示中、Ｂ面に書き終わったら表示入れ替え(FLIP) コマンドを発行
することで表示面がＢ面に変わります。
FLIP コマンドを発行するまでは描画コマンドで書き込んだ内容は表示されません。
これは表示中（フレームメモリーを読み出し中）のバッファに書き込むことに
よる画面のちらつきを防止するためです。
同様にＢ面表示中に発行された描画コマンドはＡ面に書き込みがおこなわれます
FLIP コマンドを発行することで表示は垂直同期信号のブランキング期間にＡ面に
切り替わります。
表示中の面に書き込むことはできないので、頻繁に表示を変更したい場合、
（たとえば時計や測定値などの連続表示）は予め両方の面に同じものを描画しておき、
変更したい部分のみを裏面（表示がＡならＢ面）に再描画して FLIP します。
時計表示のサンプルは **5.3 サンプルプログラムの実行**　を参照してください。

両面への書き込みと FLIP を GT320 が自動でおこなわない理由は描画コマンドを
発行するたびに A 面へ書き込み→ FLIP →B 面へ書き込み　という作業を繰り返すと
時間がかかってしまうからです。
FLIP コマンドではフレームメモリー(A 面,　B 面）の入れ替えを垂直ブランキング
期間中におこないますので、最大 17msec を待つ必要があります。
そこで A 面、B 面の管理はユーザーサイドで意識しておこなっていただく必要が
あります。

3.6 カラーパレットと色の指定

3.6.1　カラーモード

GT320 では 1.固定カラーパレット、2.グレースケール（モノクロ 64 階調）の
２種類のカラーモードを CMOD コマンド（後述）で選択できます。
但しビットマップを読み込むとファイルのヘッダー情報を識別して自動的にモードが
切り替わります。
色の指定は COLR コマンド（後述）の８ビットのパラメータでパレットを指定する
ことでおこないます。

3.6.2　グレースケールモード

グレースケールモードを選択した場合の色は 0xFF（255）が白で 00 が黒のモノクロ
グレースケールなります。
パラメータは８ビットですが、実際の液晶パネルのカラー精度が 6 ビットのため
下位の 2 ビットは無視されます。

3.6.3　固定カラーパレットモード

固定カラーパレットは下記の８ ビットで色を表します。

bit	7	6	5	4	3	2	1	0
	R	R	G	G	G	B	B	B

パレットの上２ビットが赤色(R)、真ん中３ビットが緑色(G)、下３ビットが青色(B)
に対応します。　したがって
純粋な青は "０００００１１１" ですので　7
純粋な緑は "００１１１０００" ですのでヘキサで 0x38、十進では 56
純粋な赤は "１１０００００" ですのでヘキサで 0xC0、十進では 192
となります。
赤に半分の青をまぜて赤紫にするには "１１０００１００" = 0xC4 などとします。
0xFF で白、 00 で黒となるのはグレースケールモードと同じです。
代表的な色の一覧です。

色	ヘキサ値	色	ヘキサ値
黒	0x00	シアン	0x3F
青	0x07	黄	0xF8
緑	0x38	マゼンタ	0xC7
赤	0xC0	白	0xFF

3.6.4　前景色と背景色

多くの描画コマンドではそれ以前に COLR コマンドで指定した前景色、または背景色で描画をおこないます。

例えば PUTS コマンド（文字列描画コマンド）では文字の部分を**前景色**で、それ以外のマット部分（背景）を**背景色**で描画します。

マット部分の背景透過（カラーキー合成）はできません。

HLIN, VLIN（直線描画）コマンドや FILL(塗りつぶし）コマンドなどでは以前に指定した**前景色**で描画します。

CLRS(画面消去）コマンドは以前に指定した**背景色**で塗りつぶします。

各コマンドの詳細は 4.3 を参照してください。

3.6.5　パレット変換

付属のビットマップ変換アプリで JPEG, PNG など 標準フォーマットの画像を GT320 の固定８ビット固定パレット画像に変換できますが、変換によってカラー情報が欠落するため、元の色が正確に再現されず、画質はかなり低下するケースがありますのでご了解ください。　モノクロへの変換をすれば汚く変換されることはありません。

元の画像

8bit カラーパレット変換後

左はモノクロ変換

3.7 接続と動作確認

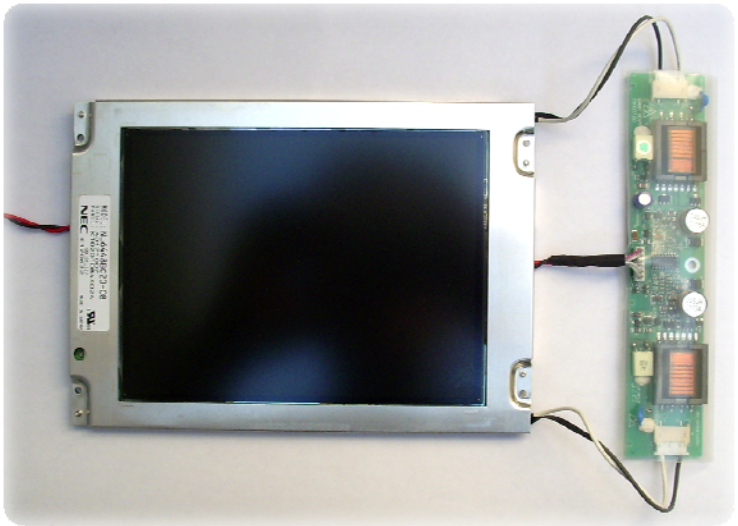
3.7.1　ＳＤカードの準備

付属の SD /MMC カードを Windows パソコンなどで **FAT32** でフォーマットします。
付属 CD-ROM の Data フォルダーの中身一式を SD/MMC カードのルートにすべて
コピーしてください。
内容は

cmfont32.fnt	フォントファイル
Sample01.cbm	サンプルのフル画像(320x240dot)ファイル
〜	
SampleXX.cbm	
Einstein.cbm	サンプルの小画像ファイル
guppy-s.cbm	デモ用の小画像ファイル

3.7.2 インバータの接続

付属の液晶パネルのバックライトは冷陰極管を使用していますのでインバータ
（下写真の右）で駆動します。
先ず液晶モジュールのインバータコネクタ（右２カ所）をインバータに接続します。
コネクタの裏表は下の写真を参照してください。(裏向けでも無理に差し込むと
入ってしまい、故障の原因となりますのでご注意ください)



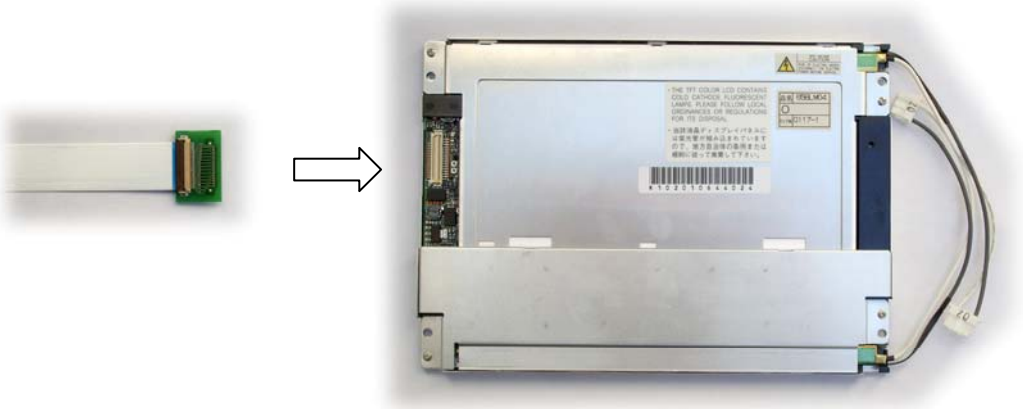
インバータ電源ケーブル（赤黒の線）を GT320 ボードのコネコタ CN-7 に接続します。

インバータは高電圧がかかる部分があって危険ですので、透明のチューブを
はずさないようにしてください。

3.7.3　液晶パネルの接続

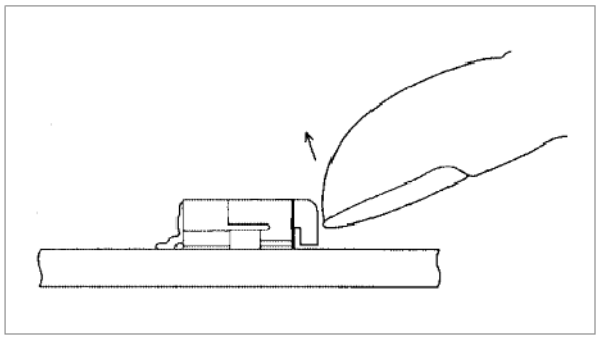
下図のように液晶パネルの信号コネクタ（白）と GT-320 ボードを FCC ケーブル（写真左）で接続します。

先ず FCC ケーブルの小基板を液晶モジュールの裏にある白いコネクターに奥までしっかり差し込みます。

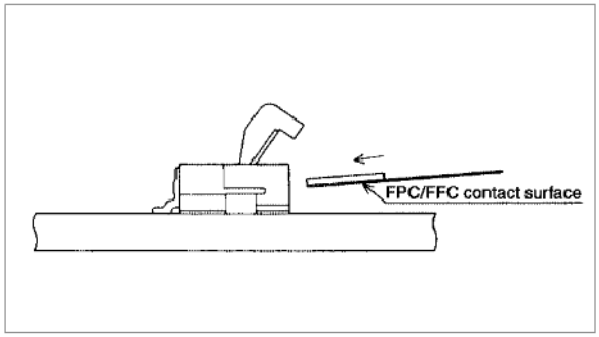


（画像が映らないというトラブルの多くはこのコネクタの接触不良です）
但しあまり力を入れすぎるとコネクターや奥の基板を破損しますので細心の注意を払ってしっかり奥まで差し込んでください。
FCC ケーブルのもう片側を GT320 本体の CN-3 （茶色のコネクタ）に接続します。
特に FCC ケーブルの GT320 ボードへの取り付けは慎重におこなってください。
これを壊すと一切の表示動作が不可能になってしまいます。
基板上の FCC コネクタ(3.3 ボードの各部の名称　を参照）のこげ茶色のアクチュエーター（可動蓋）を指でそっと FCC ケーブルの青い面を上にして差し込み、蓋を閉めます。

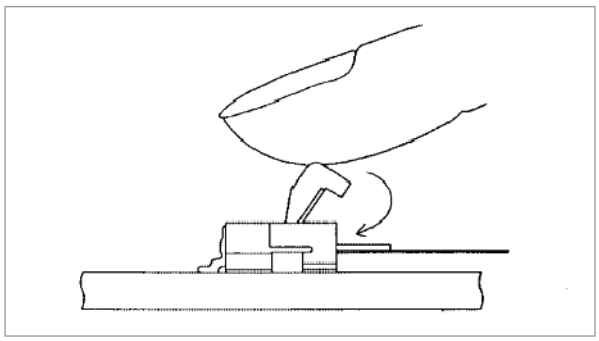
①



②



③



適正にケーブルを奥まで入れて締めるとケーブルの青い部分が２ｍｍほど外に残ります。
FCC ケーブルを差し込むときは力を入れて押し込みすぎないようご注意ください。
力を入れすぎるとアクチュエータ（こげ茶色の蓋部分）が飛んでしまうことがあります。
万一アクチュエータが飛んでしまった場合は、FCC ケーブルを挿入するのと同じ方向からそっと挿入すると奥でカチっと止まります。

3.7.4　デモ画面の表示

SD/MMC カードをカードスロットに挿入し、ボード上のディップスイッチ DIPSW-4 （4 と印刷されたもの）を ON にします（工場出荷設定）。

DIPSW-1	DIPSW-2	DIPSW-3	DIPSW-4	モード
Don't Care	Don't Care	Don't Care	**OFF**	**通常動作**
Don't Care	Don't Care	Don't Care	**ON**	**デモ画面**

この状態で電源を入れる（ DC 12V 電源のプラグを差す）と数秒して ″Loading Sample″ の表示が出たあとグラフのデモ画面と文字のデモ画面が４秒ごとに交互に表示されます。
SD/MMC カードの読み込みに失敗した場合は最初のグラフデモの画面で止まります。
電源投入後、初期化が済むまで一瞬画面ノイズが表示されますが故障ではありません。
（電源投入時の画面ノイズについて詳しくは 3.9 を参照してください）
何も表示されないで真っ黒のままの場合は接続不良の可能性が大ですのでケーブルの接続をご確認ください。
デモを終了するには電源を切って DIPSW-4 を OFF にします。

3.8 電源とスイッチ

本製品には電源スイッチが付いておりません。
組み込みなどで電源スイッチが必要な場合は ボード上の電源ジャンパー(3.3 参照)
をはずして、スイッチを挿入してください。
電源スイッチは DC 20V 3A 以上の耐性のあるものをご使用ください。

尚、電源は付属の +12V スイッチング電源アダプタをお使いください。
リップルのある電源アダプターなどは不可です。
他の電源使用がどうしても必要な場合は 1.5A 以上のとれる安定化電源をお使いください。
その場合、プラグは外側が 0V(GND)、内側が+12V の 2.1mm 標準プラグ
（最大定格 20V　3A 以上）をお使いください。
ボード内の DC-DC コンバータで 12V から内部使用の 3.3V を作っていますが、
12V 電源の立ち上がり特性によっては DC-DC コンバータが起動しなかったり、
起動に数秒かかったり、パネルのリセットがうまくかからない事があります。

3.9 電源立ち上げ時の画面ノイズについて

電源を投入した際に一瞬、画面にノイズが出てから正常表示（黒画面）になりますが故障ではありません。
液晶モジュールの DE （データイネーブル）をオフにしても電源投入時のノイズは出ますので、どうしてもこれを避けたい場合はパネルが安定するまでバックライトの 12V 電源を外部リレーでオフにするなどで対策してください。
汎用の制御出力ポートが使えますのでリレー制御などにお使いいただけます。
汎用ポートについては 次項　3.10 を参照してください。

3.10　汎用ポート入出力

GT320 には 8 ビットの汎用出力と 6 ビットの汎用入力ポートを備えています。
各々 GPOT,　GPIN コマンドでアクセスすることが可能です。
ポートのピンは 3.3 ボード裏面写真の拡張コネクター（オレンジ色で囲った部分）になりますがコネクターが実装されていないため、2.54mm ピッチのコネクタを実装するかボードに直接ハンダ付けして線を引き出す必要があります。
出力ポートのピン配置は 3.4 コネクター・ピン配置の CN-9 の項目を参照してください。
ドライブ可能な電流は 3.4 ポートの絶対最大定格 を参照してください。
出力ポートの利用方法としては
・ブザーの鳴動
・外部リレーの ON/OFF（最大８個）
・音声ボードの制御（当社音声ボード SPICA-V2 に接続すれば 8 種類の音声を出すことができます）
・LED ランプなどの点灯制御

入力ポートの利用方法としては
・スイッチの読み込み
・センサーの状態読み込み

などが考えられます。
GPOT,　GPIN コマンドの詳細は 4.3 コマンドの詳細仕様 でご確認ください。

4. ソフトウェア

4.1 通信緒元

GT320 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　**5.2 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

GT320 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）

GT320 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。

・GT320 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　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。

詳細はサンプルプログラムの項で解説します。

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** の **CSUM** コマンドの詳細仕様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 GT320 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、通常の描画コマンドは１秒以内、漢字の描画とビットマップのロードは最大３秒程度としてください。

C言語のサンプル)

```
SendCommand( "COLRC007");
```

動作：前景色を 赤(0xC0), 背景色を 青(0x07) に設定する。
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 300 はヘキサでは 0x12C ですから ４文字にして "012C" の文字列となります。
パラメータによっては(256 未満の数値) ２桁で送る場合もあります。
たとえば色指定においてはカラーパレットのコードは "00" から "FF" の間で指定します。

C言語のサンプル)

```
SendCommand( "LOCT012C0050");          // カーソルを座標( 300, 80)に移動
```

4.2.5　C 言語サンプルについて

C 言語のサンプルソースコードについては　**5.4**　を参照してください。
C言語のサンプル) で出てくる SendCommand 関数は引数の文字列にターミネータ[CR]コードを付加して RS-232C に送信する関数ですが、実際にはそのあとに 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。

4.2.6　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。

"FONT"	フォントの選択
"LOCT"	描画位置（カーソル）の移動
"PUTS"	文字列の描画
"FLIP"	フレームメモリーの表示面(A 面 or B 面)入れ替え。
"PAGE"	フレームメモリーのページ指定(Page 0=A 面、or 1=B 面)
"COLR"	前景、背景色の指定
"CLRS"	スクリーン全面消去
"CMOD"	カラーモードの設定　(1: カラー、0:モノクロ)
"DOT1"	１ドットの描画
"DOT8"	8 ドット分の一括描画
"HLIN"	水平ラインの描画
"VLIN"	垂直ラインの描画
"LINE"	ライン（自由方向）の描画
"CIRC"	円の描画
"CFIL"	円の塗りつぶし
"FILL"	矩形塗りつぶし
"EBOX"	矩形フレーム（中空）の描画
"RBOX"	角丸矩形の描画
"RFIL"	角丸矩形の塗りつぶし
"REVS"	矩形領域のカラー反転
"LBMP"	ビットマップのカーソル位置ロード
"QBMP"	フルビットマップのロード
"SAVI"	文字列展開バッファのイメージ保存
"LODI"	文字列展開バッファのイメージ読み込み
"SAVS"	スクリーン全体のビットマップ保存
"TPXY"	タッチパネル位置取得
"TPMD"	タッチパネルの割込モード (0: No Int、 1:ON Int、 2:OFF Int etc.)
"CSUM"	チェックサムモード
"VERS"	バージョンの取得
"BARI"	バーグラフの初期設定と描画
"BARS"	バーグラフの長さ変更と描画
"STRI"	文字列構造体の初期設定と描画
"STRS"	文字列構造体の文字列変更と描画

"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

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。
すべての描画コマンドは裏バッファ（表示されていない方のフレームメモリー）に
描画されるため、 FLIP コマンドでバッファを入れ替えて始めて表示されます。

コマンド	FONTmm	
機能	フォントを設定する	
引数	値	内 容
mm	00	8x6 dot ASCII フォント
	01	12x6 dot ASCII フォント
	02	20x20 dot 漢字フォント
	03	40x40 dot 倍角漢字フォント
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″FONT02″	20x20 漢字フォントを選択
備考		

コマンド	LOCTxxxxyyyy	
機能	描画位置カーソルを移動する	
引数	値	内 容
xxxx	（４桁 HEX ）	X 座標ドット位置
yyyy	（４桁 HEX ）	Y 座標ドット位置
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″LOCT00C80064″	座標(200, 100)へ移動
備考	左上が(0, 0)原点、右下が （ 319, 239 ） このコマンドでは位置を指定するだけでワープロのような実際のカーソルは表示されません。	

コマンド	PUTSsssss	
機能	文字列の描画	
引数	値	内 容
sssss	(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	″PUTSHello!″	カーソル位置に ″Hello!″を表示
備考	文字の色は″COLR″コマンドで設定した前景色、マットは背景色になる。 ASCII フォントを指定した場合は漢字を表示できない。	

コマンド	FLIP	
機能	描画バッファ(フレームメモリー) の入れ替え	
引数	値	内 容
なし		
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″FLIP″	A面、B面を切り替える(反転する)
備考	描画は表示されていない側のバッファに対しておこなわれる。 表示バッファ面を直接指定したい場合は″PAGE″コマンドを使う。	

コマンド	PAGEpp	
機能	描画バッファのページ指定	
引数	値	内 容
pp	00	フレームバッファA面を選択
	01	フレームバッファB面を選択
応答		“OK”： 成功、”NGxx”： 失敗
使用例		
備考	FLIP コマンドが現在のバッファを反転(A なら B、B なら A)するのに対して PAGE コマンドは直接ページ(0=A 面、1=B 面) を指定する。	

コマンド	COLRffbb	
機能	前景色、背景色の設定	
引数	値	内 容
ff	（２桁 HEX ）	前景色
bb	（２桁 HEX ）	背景色
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″COLR3F07″	前景色をシアン、背景色を青に
備考	カラーパレットの色については 3.6 を参照してください	

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	背景色で画面を塗り潰します。実行時間は約 117msec	

コマンド	CMODmm	
機能		
引数	値	内 容
mm	00	モノクロ（グレースケール）
	01	カラーモード
応答		“OK”：成功、”NGxx”：失敗
使用例		
備考		

コマンド	DOT1xxxxyyyy	
機能	１ドットの描画	
引数	値	内 容
xxxx	（４桁 HEX ）	X 座標ドット位置
yyyy	（４桁 HEX ）	Y 座標ドット位置
応答		“OK”：成功、”NGxx”：失敗
使用例		
備考	COLR コマンドで指定した前景色で１ドットを描画する。	

コマンド	DOT8xxxxyyyy.....xxxxyyyy	
機能	１ドットの描画	
引数	値	内 容
xxxx	（４桁 HEX ）	X 座標ドット位置
yyyy	（４桁 HEX ）	Y 座標ドット位置
・・・	（４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	COLR コマンドで指定した前景色で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	HLINxxxxyyyywwww	
機能	水平ラインを描画	
引数	値	内 容
xxxx	（4 桁 HEX ）	左上 X 座標
yyyy	（4 桁 HEX ）	左上 Y 座標
wwww	（4 桁 HEX ）	直線の長さ
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″HLIN00C8006400A0″	（ 200, 100 ）から右に 160 ドットのライン
備考	COLR コマンドで指定した前景色で左から右に描画する。	

コマンド	VLINxxxxyyyyhhhh	
機能	垂直ラインを描画	
引数	値	内 容
xxxx	（4 桁 HEX ）	左上 X 座標
yyyy	（4 桁 HEX ）	左上 Y 座標
hhhh	（4 桁 HEX ）	直線の長さ
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″VLIN00C800640040″	（ 200, 100 ）から下に 64 ドットのライン
備考	COLR コマンドで指定した前景色で上から下に描画する。	

コマンド	LINExxxxyyyyvvvvwwww	
機能	始点から終点へラインを描画	
引数	値	内 容
xxxx	（４桁 HEX ）	始点 X 座標
yyyy	（４桁 HEX ）	始点 Y 座標
vvvv	（４桁 HEX ）	終点 X 座標
wwww	（４桁 HEX ）	終点 Y 座標
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″LINE0040006401000028)″	(64, 100)から(256, 40)に線を引く
備考	COLR コマンドで指定した前景色で描画する。 （クリッピングは行わない）	

コマンド	CIRCxxxxyyyyrrrr	
機能	円を描画	
引数	値	内 容
xxxx	（４桁 HEX ）	中心点 X 座標
yyyy	（４桁 HEX ）	中心点 Y 座標
rrrr	（４桁 HEX ）	半径
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″CIRC00A000780064″	（ 160, 120 ）を中心に半径 100 の円を描画
備考	COLR コマンドで指定した前景色で描画する。 （クリッピングはおこなわれる）	

コマンド	CFILxxxxyyyyrrrr	
機能	円の塗りつぶし	
引数	値	内 容
xxxx	（４桁 HEX ）	中心点 X 座標
yyyy	（４桁 HEX ）	中心点 Y 座標
rrrr	（４桁 HEX ）	半径
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″CFIL00A000780064″	（ 160, 120 ）を中心に半径 100 の円を塗りつぶし
備考	COLR コマンドで指定した前景色で描画する。 （クリッピングはおこなわれない）	

コマンド	FILLxxxxyyyywwwwhhhh	
機能	矩形の塗りつぶし	
引数	値	内 容
xxxx	（４桁 HEX ）	左上 X 座標
yyyy	（４桁 HEX ）	左上 Y 座標
wwww	（４桁 HEX ）	矩形の幅
hhhh	（４桁 HEX ）	矩形の高さ
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″FILL0032001E00640032″	(50,30) から 100x50 の矩形塗りつぶし
備考	前景色で塗りつぶす。160x120 サイズの実行時間は約 46msec	

コマンド	EBOXxxxxyyyywwwwhhhh	
機能	矩形枠の描画	
引数	値	内 容
xxxx	（４桁 HEX ）	左上 X 座標
yyyy	（４桁 HEX ）	左上 Y 座標
wwww	（４桁 HEX ）	矩形の幅
hhhh	（４桁 HEX ）	矩形の高さ
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″EBOX0032001E00640032″	(50,30) から 100x50 の矩形枠を描画
備考	前景色で矩形枠を描画する。	

コマンド	RBOXxxxxyyyywwwwhhhh	
機能	角丸矩形枠の描画	
引数	値	内 容
xxxx	（４桁 HEX ）	左上 X 座標
yyyy	（４桁 HEX ）	左上 Y 座標
wwww	（４桁 HEX ）	矩形の幅
hhhh	（４桁 HEX ）	矩形の高さ
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″RBOX0032001E00640032″	(50,30) から 100x50 の角丸矩形
備考	前景色で角丸矩形枠を描画する。	

コマンド	RFILxxxxyyyywwwwhhhh	
機能	角丸矩形枠の塗りつぶし	
引数	値	内 容
xxxx	（４桁 HEX ）	左上 X 座標
yyyy	（４桁 HEX ）	左上 Y 座標
wwww	（４桁 HEX ）	矩形の幅
hhhh	（４桁 HEX ）	矩形の高さ
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″RFIL0032001E00640032″	(50,30) から 100x50 の角丸矩形塗りつぶし
備考	前景色で角丸矩形枠を塗りつぶす。	

コマンド	REVSxxxxyyyywwwwhhhh	
機能	角丸矩形枠の描画	
引数	値	内 容
xxxx	（４桁 HEX ）	左上 X 座標
yyyy	（４桁 HEX ）	左上 Y 座標
wwww	（４桁 HEX ）	矩形の幅
hhhh	（４桁 HEX ）	矩形の高さ
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″REVS0032001E00640032″	(50,30) から 100x50 の矩形範囲を反転
備考	矩形部分のカラーを１の補数(ビット反転) で反転する (FF--> 00, C7-->38　etc.)	

コマンド	BARInnddffbbxxxxyyyywwwwhhhhvvvv	
機能	バーグラフの初期設定と描画	
引数	値	内 容
nn	（２桁 HEX ）	バーグラフ番号　00～0F
dd	（２桁 HEX ）	バーグラフの方向　　00:水平、 01:垂直
ff	（２桁 HEX ）	前景色　00～FF
bb	（２桁 HEX ）	背景色　00～FF
xxxx	（４桁 HEX ）	左上 X 座標
yyyy	（４桁 HEX ）	左上 Y 座標
wwww	（４桁 HEX ）	バーグラフの横幅
hhhh	（４桁 HEX ）	バーグラフの高さ
vvvv	（４桁 HEX ）	バーグラフの初期値
応答		“OK”： 成功、”NGxx”： 失敗
使用例	（下記）	
備考	例）″BARI0100C0070032006400A000140050″ 2 番(01)のバーグラフを 左上起点(50,100) から 160(W)x20(H)のサイズで前景色＝赤(C0)、背景色＝青(07) で初期値の長さ＝80dot で水平タイプで描画する。　詳細は **4.6 バーグラフの描画** を参照。	

コマンド	BARSnnvvvv	
機能	バーグラフの長さ変更と描画	
引数	値	内 容
nn	（２桁 HEX ）	バーグラフ番号　00～0F
vvvv	（４桁 HEX ）	バーグラフの初期値
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″BARS020100″	３番目(02)のバーグラフの長さを 256 に
備考	詳細は 4.6 ″バーグラフの描画″ を参照。	

コマンド	STRInnttffbbxxxxyyyy	
機能	文字列構造体の初期設定	
引数	値	内 容
nn	（２桁 HEX ）	文字列構造体番号　00～0F
tt	（２桁 HEX ）	フォント ID　00～03
ff	（２桁 HEX ）	前景色　00～FF
bb	（２桁 HEX ）	背景色　00～FF
xxxx	（４桁 HEX ）	左上 X 座標
yyyy	（４桁 HEX ）	左上 Y 座標
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″STRI0102C00700320064″	2 番目の文字列構造を 20 ドット漢字フォント、前景色＝赤(C0)、背景色＝青(07) 、左上起点(50,100)に設定する
備考	詳細は **4.5 文字列の描画** を参照。	

コマンド	STRSnnssssss	
機能	文字列構造体の文字列描画	
引数	値	内 容
nn	（２桁 HEX ）	文字列構造体番号　00～0F
ssssss	文字列	
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″STRS03Hello!″	4 番目の文字列構造体で ″Hello!″を描画
備考	詳細は **4.5 文字列の描画** を参照。	

コマンド	QBMPsssss	
機能	フル・ビットマップのロード	
引数	値	内 容
sssss	ASCII 文字列	ファイル名
応答		“OK”： 成功、”NGxx”： 失敗
使用例	QBMPSample01	″Sample01.cbm″ 画像をロードする。
備考	320x240 dot のビットマップをロードする。 ファイル名は拡張子(.cbm) を含まない。 必ず原点からロードされるので LOCT コマンドは不要。	

コマンド	LBMPsssss	
機能	任意サイズビットマップのロード	
引数	値	内 容
sssss	ASCII 文字列	ファイル名
応答		“OK”： 成功、”NGxx”： 失敗
使用例	LBMPEinstein	″Einstein.cbm″ 画像をカーソル位置にロード。
備考	任意サイズのビットマップをカーソル位置を左上にロードする。 ファイル名は拡張子(.cbm) を含まない。 直前の LOCT コマンドで指定した位置に読み込まれる。	

コマンド	TPXY	
機能	タッチパネルの最新クリック位置を取得する	
引数	値	内 容
(なし)		
応答	TPXYxxxxyyyy	xxxx, yyyy は４桁 HEX の x, y 座標
使用例	"TPXY"	
応答例	"TPXY00C60036"	座標(198, 54) がクリックされた
備考	左上が(0, 0)原点	

コマンド	TPMDmm	
機能	タッチパネルの割り込みモードの設定	
引数	値	内 容
mm	00	イベントなし
	01	ON イベントを許可
	02	OFF イベントを許可
	03	ON, OFF イベントを許可
	05～07	bit-2 が ON の場合は RAW モード
応答		"OK" : 成功、" NGxx" : 失敗
使用例	"TPMD01"	タッチパネルをクリック ON したときのみ割り込む
備考	「イベントなし」に設定した場合は TPXY コマンドでポーリング可能。 詳細は **6.3 タッチパネルの動作モード** 参照	

コマンド	SAVIssss	
機能	文字列描画のイメージ保存	
引数	値	内 容
ssss	４文字文字列	ファイル名（通常は４桁の数字）
応答		“OK”：成功、”NGxx”：失敗
使用例	″SAVI0003″	″$$0003.img″ に保存
備考	最新の PUTS コマンド描画した内容を位置を含めて保存する。	

コマンド	LODIssssxxxxyyyy	
機能	文字列描画のイメージ読み込み	
引数	値	内 容
ssss	４文字文字列	ファイル名
xxxx	（４桁 HEX ）	左上 X 座標
yyyy	（４桁 HEX ）	左上 Y 座標
応答		“OK”：成功、”NGxx”：失敗
使用例	″LODI000300C80064″	″$$0003.img″ から(200,100)に読み込み
備考	イメージファイルから指定座標が左上になるよう読み込む。	

コマンド	SAVSssssss	
機能	スクリーン（全画面）のビットマップ保存	
引数	値	内 容
ssssss	可変長文字列	ファイル名
応答		“OK”：成功、”NGxx”：失敗
使用例	″SAVItest1″	″test1.cbm″ に保存
備考	ファイルネームに拡張子(.cbm)を付けないでください。 (自動付加して保存します。実行完了には数秒を要します)	

コマンド	CSUM	
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 ″FONT02″ の場合、各文字の ASCII 値を合計すると
　F　　O　　N　　T　　0　　2
0x46 + 0x4f + 0x4e + 0x54 + 0x30 + 0x32 = 0x199 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 ″FONT0299[CR]″ となります。
（［CR］はターミネータの 0x0d ）
・サンプルプロジェクトでは xxxx
関数のところでチェックサム計算をおこなっています。
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。

コマンド	GPOTnn	
機能	汎用出力ポートに出力する	
引数	値	内 容
nn	（２桁 HEX ）	出力する８ビット値
応答		“OK”： 成功、”NGxx”： 失敗
使用例	″GPOT85″	出力ポートのビット 7, 2, 0 を ’1’ にする
備考		

コマンド	GPIN	
機能	汎用入力ポートの値を読み込む	
引数	値	内 容
（なし）		
応答	GPINnn	ex.）″GPIN21″ ビット 5, 0 が’1’
使用例	″GPIN″	ポート入力
備考	nn（２桁ヘキサ値）の下６ビットが有効	

汎用入出力ポートについては **3.10 汎用ポート入出力**　を参照してください。

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	"VERS1.02"	バージョン 1.02
備考		

4.2.7　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　ビットマップの形式

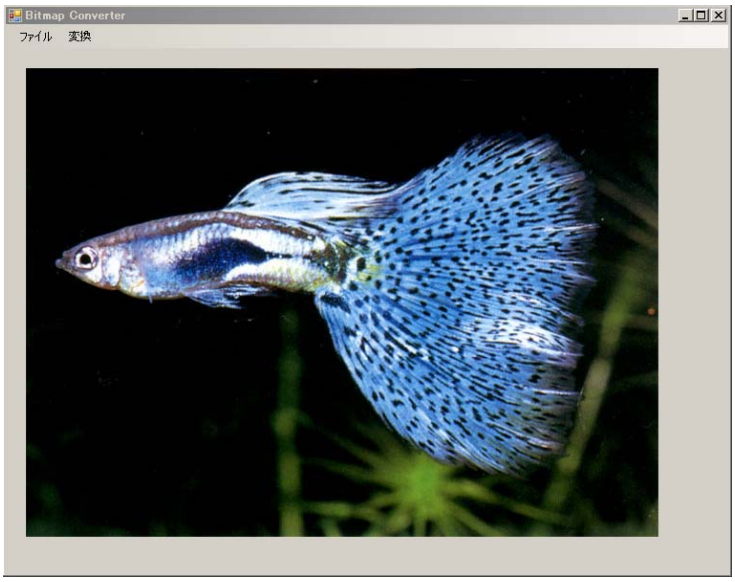
4.4.1　GT320 で扱えるビットマップ形式

GT320 では拡張子 **.cbm** のサイバーメロン独自形式のビットマップを扱います。
Windows の DIB 形式（左下原点）と異なり、左上原点でパレットは 8bit 固定パレットのみサポートします。
付属の BitmapConv8 で JPEG , Bitmap(Windows) などの画像を GT320 専用の画像形式（拡張子 .cbm)に変換できますので変換してから SD/MMC カードにコピーしてご利用ください。
固定パレットに色変換した時点で色の精度が落ちるため同じ色は再現できませんが青、緑が多い画像では赤の多い画像に比べて近い色が出ます。
変換でモノクロの 6bit グレースケール（白黒の濃淡）を選ぶこともできます。

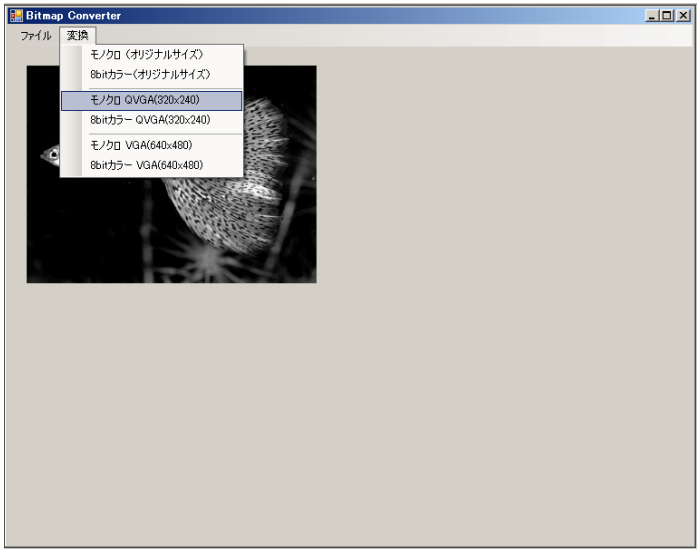
4.4.2　BitmapConv8　によるビットマップの作成

GT320 では **QBMP**, **LBMP**　コマンドによりファイル名を指定することでこの形式の画像を SD/MMC カードから読み込んで表示します。
画像のサイズやカラー、モノクロ変換にも対応します。
BitmapConv8 のインストール方法については **5.3.1** を参照してください。

・BitmapConv8 を立ち上げて[ファイル]メニューの[画像を開く]で任意サイズの画像を開きます。



・[変換]メニューで希望の形式（カラー、モノクロ、各サイズ）に変換します。



「オリジナルサイズ」を選ぶとサイズの変更はされません。
GT320 では QVGA(320x240dot)以上の画像を表示できませんので、VGA (640x480) は選択しないでください。

・変換ができましたら[ファイル]メニューの[Bitmap の保存]で名前をつけて拡張子 .cbm で保存し、 SD/MMC カードにコピーしてください。

・GT320 で SD/MMC カードに保存したビットマップを表示させるには **QBMP**, **LBMP** コマンドでファイル名を指定します。
QBMP コマンドは QVGA(320x240dot)サイズの画像を原点から画面全体に読み込んで表示します。 **LBMP** コマンドはより小さい画像を直前の LOCT コマンドで指定した位置が画像の左上になるよう読み込んで表示します。

C 言語のサンプル)

```
SendCommand("QBMPSample01");     // "Sample01.cbm" ファイルを裏バッファに
                                 // 読み込む
SendCommand("FLIP");             // 裏バッファを表に入れ替えて表示
```

4.5 文字列の描画

4.5.1　フォントの種類

GT320 では ASCII 文字（ANK 半角カタカナは含まず）と JIS 第二水準の漢字を表示可能です。　ファントは以下の４種類あり FONT コマンドの引数のフォント ID で指定します。

FONT ID	サイズ	種類
0	8 x 6dot	ASCII のみ
1	12 x 6dot	ASCII のみ
2	20 x 20dot	ASCII(20x10), 漢字(20x20)
3	40 x 40dot	20dot フォントの倍角

C 言語のサンプル)

```
SendCommand("LOCT00280064");     // カーソルを( 40, 100 )へ
SendCommand("FONT02");           // 20x20 フォントを選択
SendCommand("COLR3F07");         // 文字＝シアン色、背景＝青
SendCommand("PUTS 明けましておめでとう");  // 文字の描画
```

漢字を含む文字列は必ず SHIFT-JIS で送ってください。

ASCII 文字はフォントを Flash ROM から読み出すので高速描画されますが漢字は SD/MMC カードのフォントファイルからフォントを読み出すため、１文字の描画に数十ミリ秒の時間がかかります。
漢字の描画を高速化するための方法として SAVE, LOAD コマンドが用意されています。

4.5.2　SAVI, LODI コマンド

SAVI (Save Image)コマンドは直前にメモリーに展開した文字列をビットマップとして SD/MMC カードに保存します。

SAVI コマンドでは引数に４桁の番号か文字列を渡すことでファイル名を指定します。例えば "1234" を指定した場合は SD/MMC カード上に "$$1234.img" という名前のファイルが作成されます。
このファイルには展開された画像だけでなく、位置情報も含まれるためビットマップと区別してファイル名の拡張子は .img になります。
LODI (Load Image)コマンドで同様に４桁の番号を指定することで保存したイメージ

を読み込んで描画します。
漢字フォントを展開する時間に比べてイメージを読み込むのは場合によりますが約 1/3 程度の実行時間で済みます。

C 言語のサンプル)

```
SendCommand("LOCT00100020");      // カーソルを( 16, 32 )へ
SendCommand("PUTS あいうえお");   // 文字列描画
SendCommand("SAVI0003");          // $$0003.img ファイルに保存
・・・・
SendCommand("FLIP");              // 面を入れ替え
SendCommand("LODI0003006400C8"); // $$0003.img ファイルから
                                  // 座標(100,200)に読み込み
```

4.5.3　STRI, STRS コマンド

前述の通り、ASCII 文字は ROM から読み出すため、描画は高速ですが、たとえばタイマーで時間とバーグラフを同時に高速表示したい場合など、COLR コマンドによる色の切替、LOCT コマンドによる位置の設定などの併用が必要になり実行に時間がかかってしまいます。
そこで予め文字色や表示位置を GT320 内部の**文字列構造体**に STRI コマンド(String Initialize)で登録しておけば その後 STRS コマンド(String Set)で文字列を指定するだけで、その文字列を予め登録した属性（位置や文字色）で高速描画することが可能になります。
文字列構造体は最大 16 個のいずれかを指定して使うことができます。
詳細は 4.3 のコマンド表と VB サンプルプログラム(GT320Test) のソースを参照してください。
注意点としては文字列が可変長のため、後から描画する文字列の方が短い場合、先に描画した文字列の末尾が残りますので、スペースを追加するなどして文字列の長さを合わせてください。

4.6 バーグラフの描画

・バーグラフ（棒グラフ）を矩形塗りつぶしなどの基本コマンドの組み合わせで描画しようとした場合、以下の複数の動作が必要になって実行に時間がかかってしまいます。

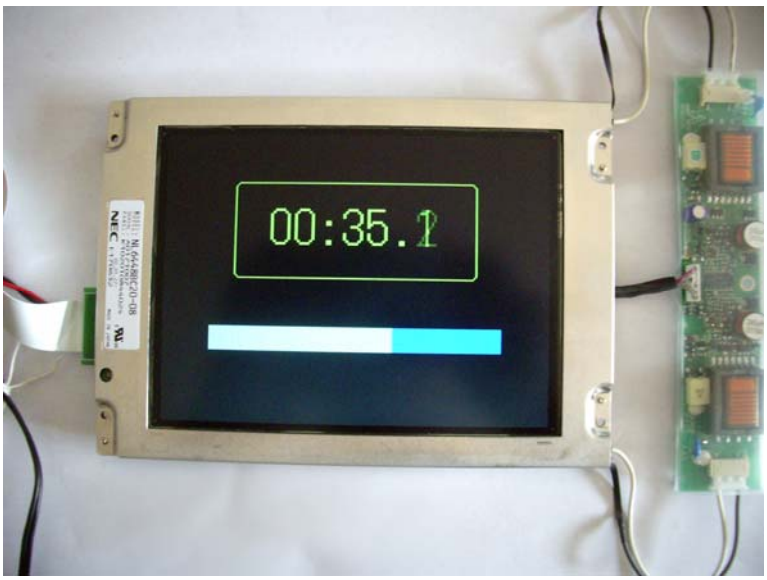
1. バーグラフの背景の色を COLR コマンドで前景色に設定し、その応答を待つ。
2. 描画位置を LOCT コマンドで指定して、その応答を待つ。
3. バーグラフ領域全体の背景を FILL コマンドで塗って、その応答を待つ。
4. バーグラフの前景色を COLR コマンドで設定し、その応答を待つ。
5. バーの長さ分の領域を FILL コマンドで塗って、その応答を待つ。

そこで位置やサイズ、色などを予め登録しておいてバーグラフの長さの変化だけ送ることでバーグラフを高速で表示させるコマンドが **BARI**, **BARS** コマンドです。
GT320 ではバーグラフを最大１６個登録することが可能です。
あらかじめ **BARI**（BarGraph Initialize)コマンドでバーグラフの タイプ(水平、垂直)、前景、背景色　位置、サイズ、初期値（バーグラフの長さ）を登録し描画します。
あとはバーグラフの長さを変化させたいときに **BARS**（ BarGraph Set)コマンドでグラフの長さを送るだけで簡単に変化させることができます。

・バーグラフには**水平タイプ**と**垂直タイプ**の２種類があり、 BARI コマンドの引数で指定します。

・水平タイプのバーグラフは左から右へ、垂直タイプは下から上へバーが伸びます。
詳細は **4.3** のコマンド表と VB サンプルプログラム(GT320Test) のソースを参照してください。

GT320Test で 0.1 秒ごとに数値とバーグラフが更新される



5. コマンドの実行

5.1 GT320 の利用形態

GT320 の利用方法には

１．パソコンからの制御

２．組み込みマイコンからの制御

などが考えられますが、機能と制御の基本原理を理解するためにはパソコンの
ターミナルソフトから手動でコマンドを打ち込んでみるのが近道です。

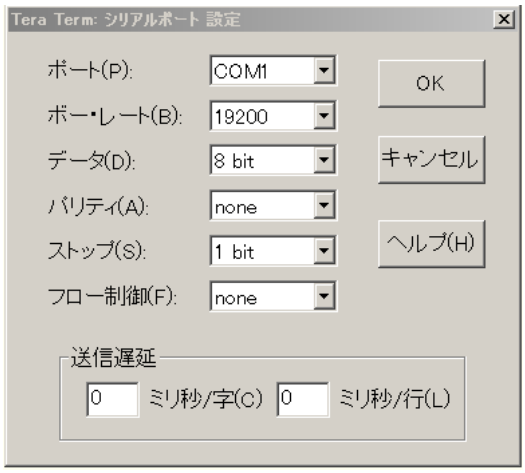
5.2 ターミナルソフトからの描画テスト

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 GT320 の動作をテストしてみます。

5.2.1　ターミナルソフトの設定

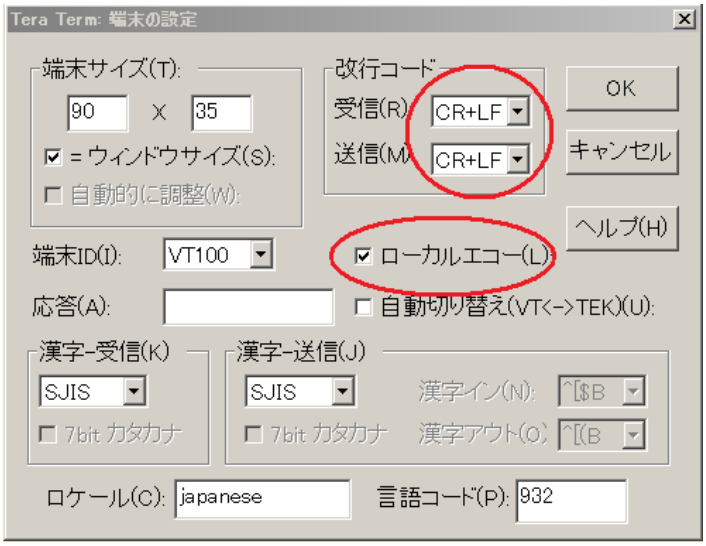
・付属のオス・メス　ストレートケーブルで PC と GT320 を接続してください。
RS-232C 端子のないパソコンの場合、USB の仮想 COM ポートを使います。
USB－RS-232C 変換ケーブルで変換してください。(変換ケーブルは付属しておりません)

・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 2011 年 9 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。

5.2.2　GT320 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ず、最初ノイズを一瞬表示したあと黒画面になります。

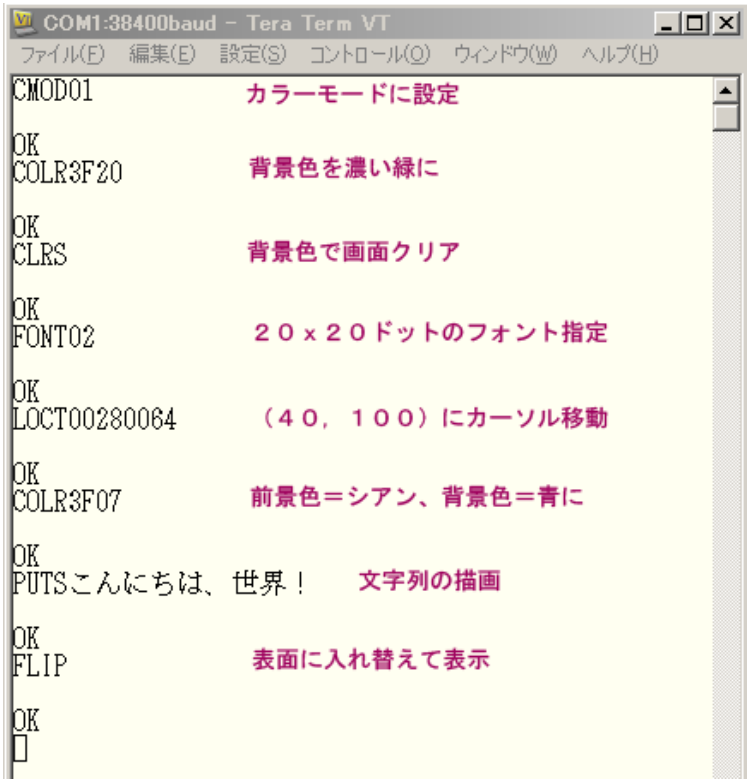
5.2.3　コマンドの実行

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。

先ずは色の設定と文字列の描画を実行してみましょう。

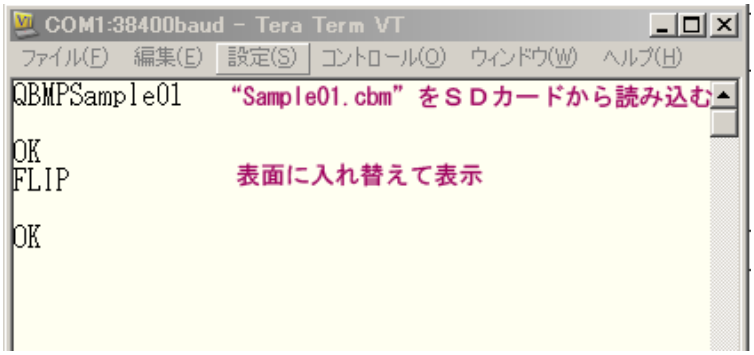
FLIP コマンドを打ち込むまでは裏側（表示されていない側）のフレームメモリーに描画されるので描画結果は見えません。

各コマンドの詳細は 4.3 を参照してください。

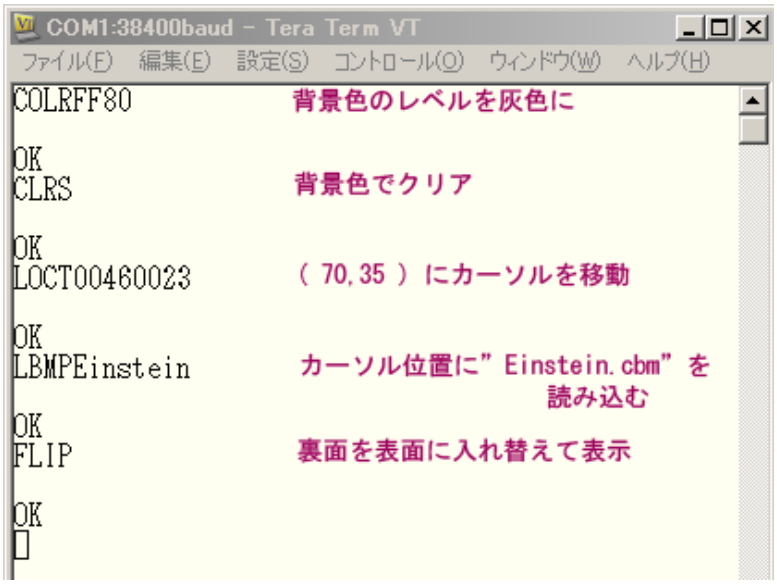


結果はこのようになります。

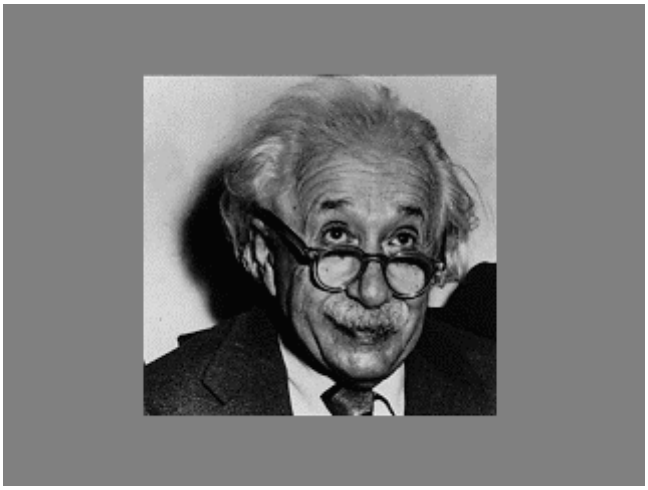
次に QBMP コマンドでフルサイズ(320x240)のビットマップを読み込んでみます。



小さいサイズのビットマップを LBMP コマンドでカーソル位置に読み込みます。



結果は画像がカーソル位置が左上にくるように読み込まれます。

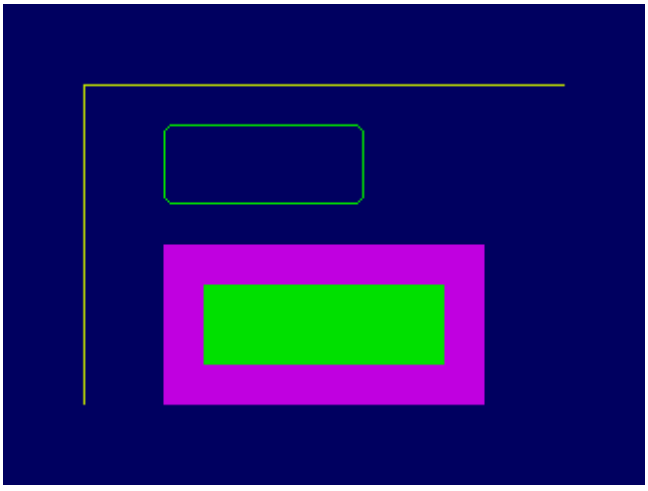


ビットマップがモノクロだったのでカラーモードも自動的にモノクロ
グレースケールに変更されました。
COLR コマンドで前景を 0xFF（白） 背景を 0x80 にしたのでビットマップの外側の
背景色はグレースケールの中間値（灰色）になっているのがわかります。

グラフィックコマンドをテストします。

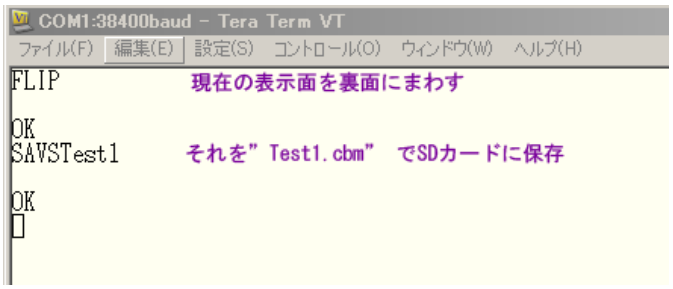
```
COM1:38400baud - Tera Term VT
ファイル(F)  編集(E)  設定(S)  コントロール(O)  ウィンドウ(W)  ヘルプ(H)
COLRF803                        前景色を黄色、背景色を暗い青に

OK
CLRS                            背景色で全面塗りつぶし

OK
HLIN0028002800F0                ( 40,40 ) を起点に長さ240 の水平ラインを引く

OK
VLIN0028002800A0                ( 40,40 )を起点に長さ 160 の縦ラインを引く

OK
COLR3803                        前景色を緑に

OK
RBOX0050003C00640028            ( 80,60 )を左上に 100x40 の角丸矩形を描画

OK
COLRC703                        前景色をマゼンタに

OK
FILL0050007800A00050            ( 80,120 )を左上に 160x80 の矩形領域を塗りつぶす

OK
REVS0064008C00780028            ( 100,140 )を左上に 120x40 の矩形領域をカラー反転

OK
FLIP

OK
```

上記の実行結果

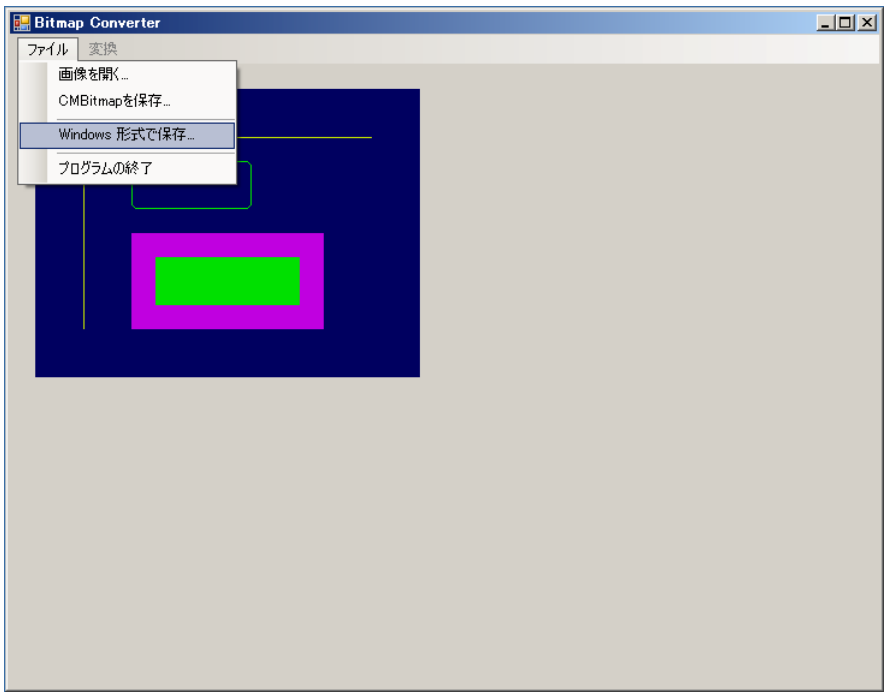


5.2.4　画像の保存

描画コマンドで作成した画面は SAVS コマンドで SD/MMC カードにビットマップ(.cbm 形式)保存することができます。
これによって GT320 を組み込んだ製品のカタログや取説などの作成が容易になります。



画像が保存されるのは表示されていない方の面ですので、必ず FLIP コマンドで面を入れ替えてから実行してください。
4.4 の BitmapConv8 アプリで.cmb ファイルを読み込んで Windows 形式の画像ファイル(Bitmap, JPEG, PNG など) に変換することが可能です。



5.3 サンプルプログラムの実行

・サンプルプログラムは付属 CD-ROM の"Sample Programs" フォルダーに以下の
プロジェクト　**GT320Test**　が入っています。
これは GT320 をパソコンから制御する VB.net のサンプルプロジェクトですが
VB.net の構文がおわかりのプログラマーなら C 言語などで制御する場合の参考
にもなると思います。

・VB プログラムの実行には .NET Framework 2.0 以上の環境が必要ですが付属の
インストーラでアプリをインストールした場合は Windows XP でも必要な
DLL は自動的にインストールされます。
もしサンプルプログラムを改造する場合はコンパイルに VisualStudio 2005
以上の開発環境が必要です。

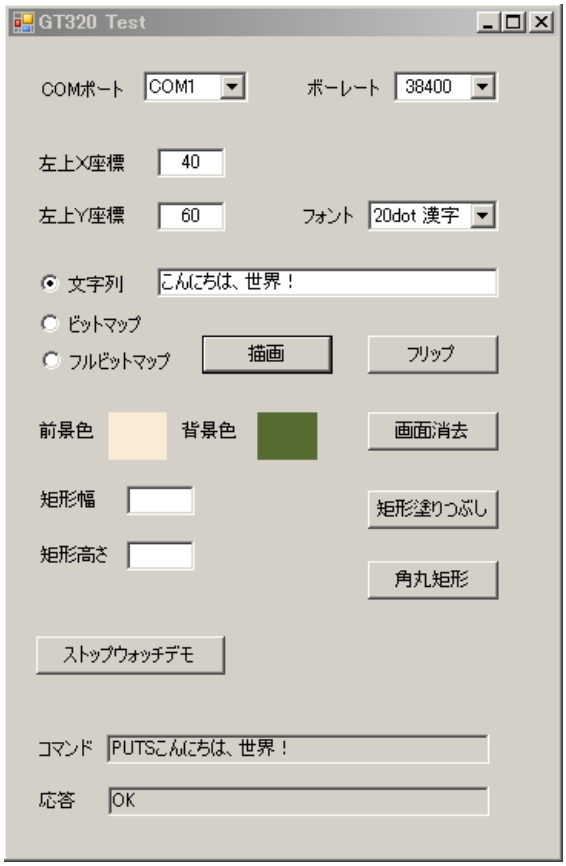
5.3.1　プログラムのインストール

Setup フォルダーの Setup.msi を起動して指示に従ってインストールしてください。
これで GT320Test と BitmapConv8 アプリおよびそれに必要な .NET Framework
環境がインストールされます。

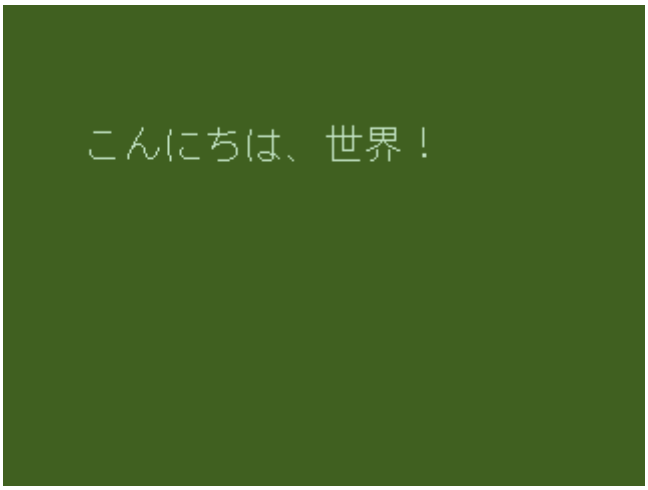
5.3.2　GT320Test

GT320Test は PC から GT320 を制御する簡単なサンプルプログラムです。

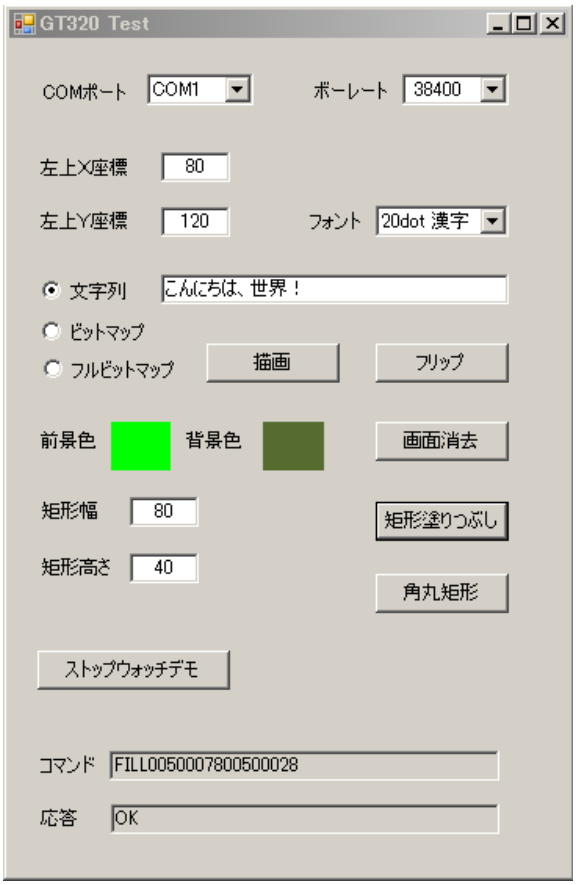
・GT320 をパソコンと RS-232C ケーブルで接続し、GT320 本体に 準備のできた SD/MMC
カードを挿入して電源を入れます。
・PC から GT320Test.exe を実行すると下記の画面になります。
・先ずは背景色を設定して [画面消去]ボタンを押してください。
これで「応答」のところに "OK" が返ってくれば画面が背景色で塗り潰されます。
（[フリップ] をクリックするまでは表示されません）
・左上 X 座標と左上 Y 座標を入力して「文字列」のところに短い文章を打ち込んで
[描画]ボタンをクリックします。
・続いて[フリップ]をクリックすると今まで書き込んだ面が表に入れ替わり表示
されます。



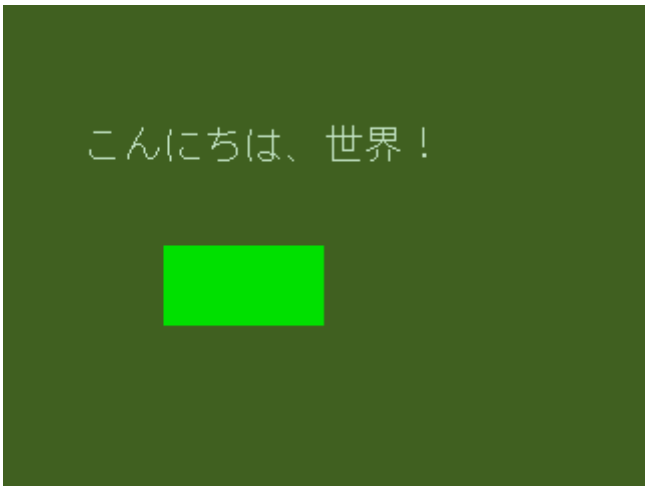
[フリップ]実行結果



・この文字の下に緑の矩形を描画してみます。
・[フリップ]をクリックして再び面を入れ替えます。
・前景色を緑に変えて左上座標を変更します。
・「矩形幅」「矩形高さ」を入力して[矩形塗りつぶし]ボタンをクリックします。



［フリップ］で表に表示させます。



・次に[ストップウォッチデモ]をクリックします。
これは高速で画面をアップデート（書き換え）するデモです。
右端の数字は 0.1 秒単位で更新され、同時にバーグラフが延びていきます。

高速で描画を更新するには A, B 両面に同じものを描画しておき、その後生じた差分を追加描画しては FLIP で画面を入れ替えることで実現します。
尚、このデモは高速動作のため GT320Test で応答文字列の表示はおこなっておりません。

プログラムの詳細は ソースで Form1.vb のタイマールーチン Timer1_Tick() を参照してください。
0.1 秒ごとにタイマーで時間をカウントアップして **STRS** コマンドで文字を高速描画し、バーグラフの長さを **BARS** コマンドで設定したあと **FLIP** しています。
文字列、バーグラフの位置や色はあらかじめ DrawStopWatch() 関数で設定しています。

詳細は
3.5 フレームメモリー
4.5.3　STRI, STRS コマンド
4.6 バーグラフの描画
を参照してください。

5.3.3　GT320Test のコンパイル

・CD-ROM の “Sample Programs” フォルダーをどこかハードディスクの適当な場所にコピーしてください。

・2005 以降の環境でコンパイルする場合はプロジェクトの変換を尋ねてきますので新しいバージョンにプロジェクトを変換してください。
コンパイル結果は GT320Test¥bin¥Debug （Debug モードの場合）などに作成されますのでスタートメニューから実行される GT320Test.exe とは別物である点をご注意ください。

・このプログラムを参考にボタンの機能など追加して各コマンドの動作を学習してください。

・尚、当サンプルプログラムに関して開発環境の使い方や言語仕様に関する内容、コンパイルが通らない、等のご質問はサポート外となりますのでご了承ください。

5.4　C 言語によるマイコンからの GT320 制御

5.4.1　接続

GT320 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。

5.4.2　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
 }
```

5.4.3　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
```

```
}
//-------------------------------------------------------
//    文字と図形の簡単な描画サンプル
//
//    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();
    //  前景色を黄色、背景色を暗い青に
    SendCommand("COLRF803");        // F8:黄色, 03:ダークブルー
    // 背景色で塗りつぶし
    SendCommand("CLRS");
    // (40, 40)を起点に長さ 240 の水平ラインを引く
    sprintf( cmd, "HLIN%04X%04X%04X", 40, 40, 240 );
    /*-----------------------------------------------
    // sprintf が使えない時は以下のルーチンで代用
      strcpy( cmd, "HLIN" );
      format_hex4( t, 40 );         // 4 桁へキサに( sprintf  の代用 )
      strcat( cmd, t );
      format_hex4( t, 40 );
      strcat( cmd, t );
      format_hex4( t, 240 );
      strcat( cmd, t );
    -----------------------------------------------*/
      SendCommand( cmd );
    // 前景色を緑に
    SendCommand("COLR3803");        // 38:緑, 03:ダークブルー
    // (40, 60) を左上に 240 x 60 の角丸矩形を描画
    sprintf( cmd, "RBOX%04X%04X %04X %04X ", 40, 60, 240, 60 );
    SendCommand( cmd );
    // 前景色を水色 (シアン) に
```

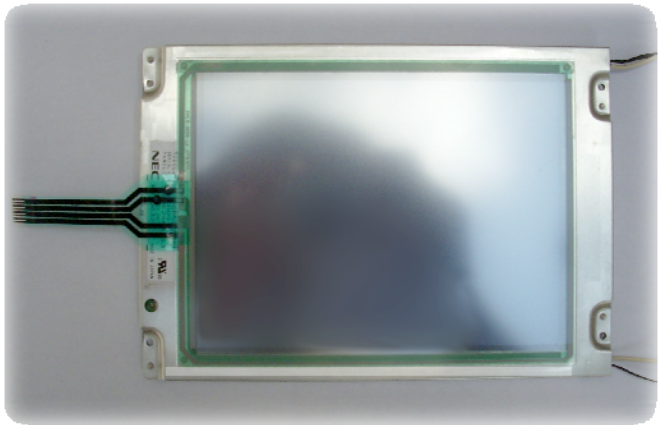
```
    SendCommand("COLR3F03");          // 3F:シアン, 03:ダークブルー
    // フォントを 20x20dot に
    SendCommand("FONT02");
    // カーソルを (60, 80)に
    sprintf( cmd, "LOCT%04X%04X", 60, 80 );
    SendCommand( cmd );
    // 文字列を描画
    SendCommand("PUTS こんにちは、世界！");
    // 前景色をマゼンタに
    SendCommand("COLRC703");          // C7:マゼンタ, 03:ダークブルー
    // (80, 140)を左上に 160x60 の矩形領域を前景色で塗りつぶす
    sprintf( cmd, "FILL%04X %04X%04X%04X", 80, 140, 160, 60 );
    SendCommand( cmd );
    // 前景色を暗い赤、背景色をマゼンタに
    SendCommand("COLR80C7");          // 80:ダークレッド、C7:マゼンタ
    // カーソルを (100, 150)に
    sprintf( cmd, "LOCT%04X%04X", 100, 150 );
    // フォントを倍角に
    SendCommand("FONT03");
    // 数字を描画
    SendCommand("PUTS123456");
    // 今まで描画した裏面を表に入れ替えて表示させる
    SendCommand("FLIP");
}
```

実行結果

６．タッチパネル

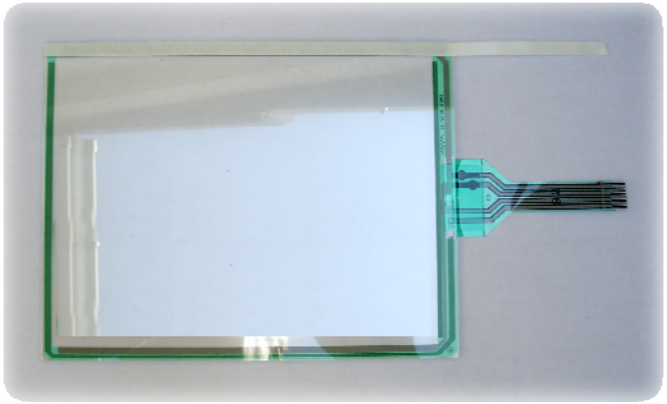
6.1　タッチパネル（オプション）

GT320 にオプションのタッチパネルを付加することで柔軟なユーザーインターフェースを構築することができます。ノングレア表面処理により蛍光灯等の映り込みが緩和されています。　タッチパネルは下の写真の向きに取り付けます。

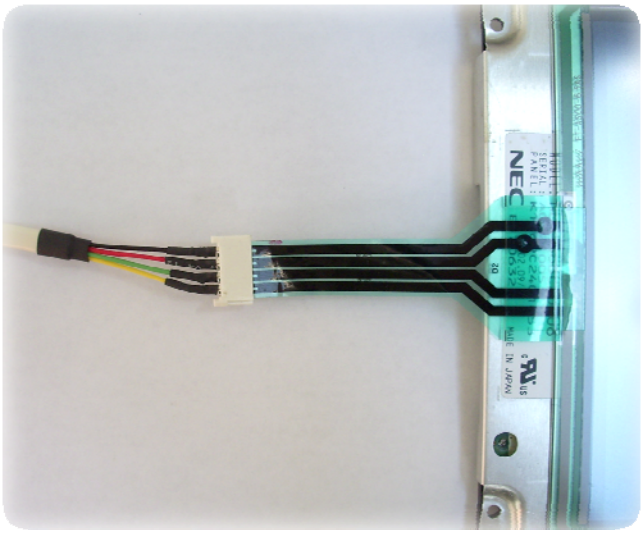


6.2　タッチパネルの接続

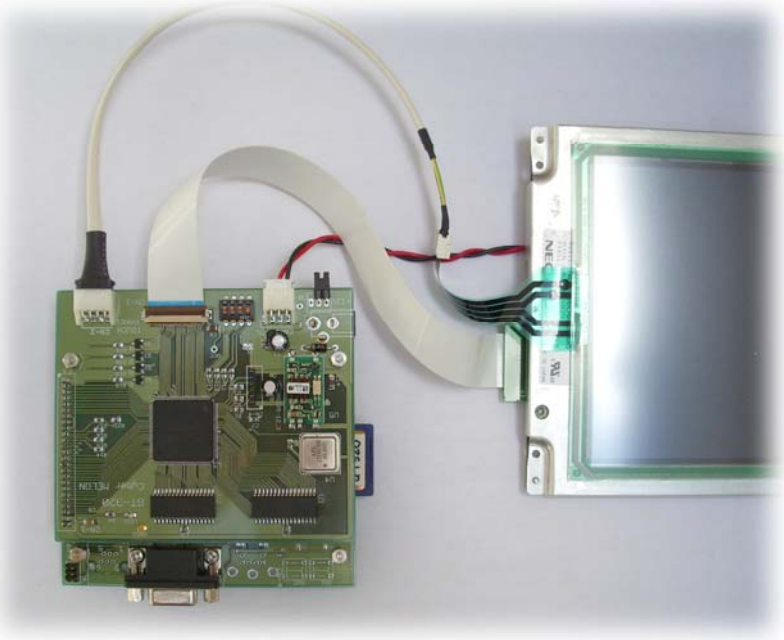
タッチパネルを液晶に両面テープで貼り付けます。　表裏に注意してください。
タッチパネル裏返して 5mm 幅の両面テープを外周に合わせて周囲に貼り付けます。



液晶パネルの表示範囲にピッタリ位置を合わせて貼り付けます。
液晶パネルを点灯した状態の方が位置合わせがしやすいかもしれません。
貼り付けが終われば４芯のコネクターで GT320 ボードと接続します。
下の写真のようにケーブルの黄色線が下に来るよう差し込んでください。



ケーブルのもう片方をボードの CN-2 に接続します。(写真でボードの左上)



6.3　タッチパネルの動作モード

GT320 は初期状態ではタッチパネルのイベント発生が禁止になっていますので、
TPMD （タッチパネルモード）コマンドを送ってイネーブルする必要があります。

C 言語のサンプル)　　SendCommand("TPMD03");

引数の 03 の部分の意味は

01	クリック ON をイネーブル（指などを押しつけたときに発生）
02	クリック OFF をイネーブル（指を離したときに発生するイベント）
03	ON/OFF 両方のイベントをイネーブル
05	クリック ON を RAW モードでイネーブル
06	クリック OFF を RAW モードでイネーブル
07	ON/OFF 両方のイベントを RAW モードでイネーブル

上記からわかるように
bit-0 が ON イベントのイネーブル
bit-1 が OFF イベントのイネーブル
bit-2 が立った場合(05～07)は RAW モードとなります。
GT320 のタッチパネルはアナログ方式のため、電圧を A/D 変換したものを計算で X, Y 座標に変換しております。
RAW モードでは GT-320 が計算した X,Y 座標のかわりに X, Y の生の A/D 変換値を返します。　ユーザープログラムで校正をおこない場合は RAW モードで A/D 変換値を取得して座標に変換してください。

イネーブルした項目はタッチパネルを操作するごとに RS232C へイベント応答を返します。　イベント応答の形式はクリック ON の場合

　"TPONxxxxyyyy [CR]"

xxxx はクリックされたＸ座標のヘキサ値、yyyy はＹ座標のヘキサ値です。

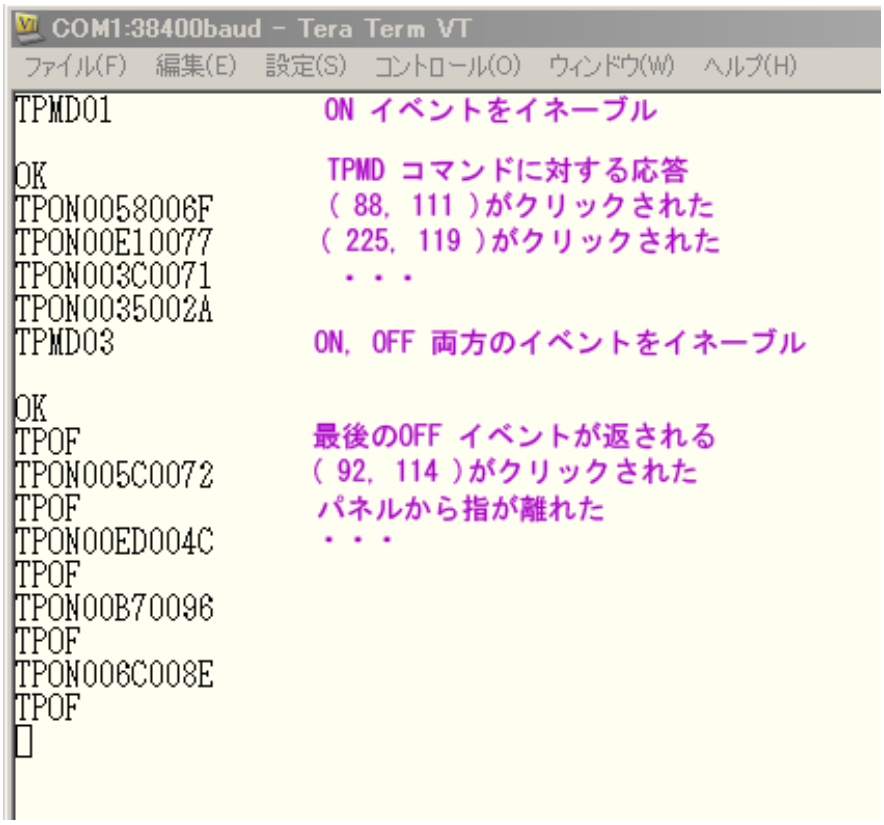
RAW モードの場合は xxxx, yyyy 座標の代わりに A/D 変換値になります。

座標は画面左上を原点(0, 0)とし、右下端が (319, 239)となります。

クリック OFF イベントの応答形式は　"TPOF[CR]"　となります。

"TPMD00" でイベント発生を禁止した場合でも TPXY コマンドによるポーリングで最後にクリックされた情報を取得することができます。

以下はＰＣの端末からコマンドを送ってテストした結果です。



6.4　タッチパネルの動作範囲と使用条件

本タッチパネルは 2 点押しには対応しません。
またクリック ON したままドラッグしても OFF までの間にイベントは発生しませんので、 ON/OFF 以外の使い方はできません。

より正確な座標が必要な場合は前述の RAW モードで値を取得してユーザーアプリケーション側のソフトで校正するようにしてください。
アナログ方式の欠点として周辺の座標が正確に出にくいことがありますが、これは保証外とさせていただきます。
周辺部で正確な座標取得が必要なアプリケーションには向いておりません。
ボタンのクリックなど大凡の位置を知る手段としてお使いください。

メーカー公表の寿命と操作条件は以下の通りです。
・5 万回往復以上（ポリアセタールペンにて約 30mm 移動）
・100 万回以上　（シリコンゴム 60°にて押下荷重約 4.9N）
・操作力 1.47N 以下

＊ ポリアセタールペンなどは当社では販売しておりません。

7. その他

7.1　液晶パネルの検査基準

液晶パネルのデフェクト（ドット不良）は下記の基準で点灯検査・選別しております。

- **輝点、暗点 周辺で４個まで、中央付近では３個まで**
- **隣接輝点（白など目立つ輝点）は１箇所まで**

この範囲のデフェクトにつきましてはクレームとして返品交換をお受けできませんので予めご了承ください。

上記の基準よりシビアな仕様をご希望の場合は別途選別品をお見積もり致します。

7.2　特注仕様など

・数量によりましてはボードのみ、液晶パネルのみの販売もいたします。
・本製品を利用した特注ソフトウェアの開発請負や量産設計も致します。

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 ）