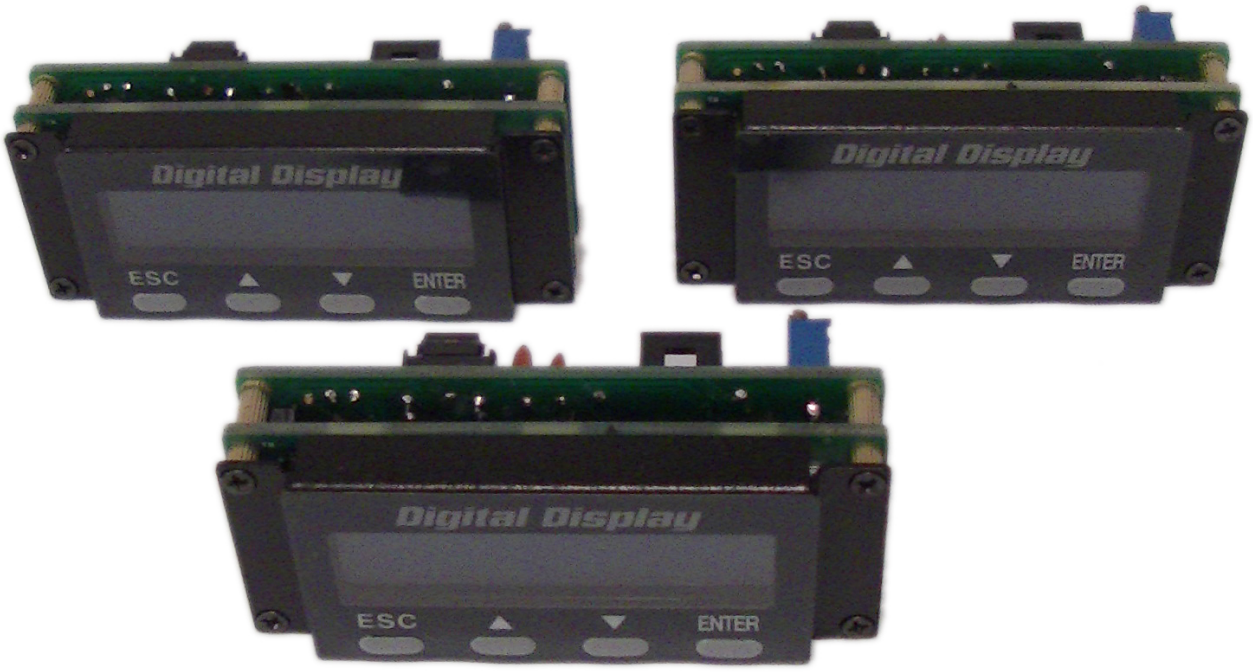
DATASHEET

PART NO: GTLCM2X16e



GTLCM2X16e

Product Overview

The major purpose of this module is to provide an easier man-machine interface for those computing systems in whose applications friendly operation is a “must.” In traditional computing system design, proprietary keypad and LCD display interfaces are implemented and these interfaces are usually different from system to system. The first design goal of this module is to define a single host interface for both LCD display and keypad. The second design goal is that this interface should be available in every computing system. The third design goal requires the implementation to be OS independent.

Our solution is to use “Serial port” as the interface for both LCD display and keypad. A simple protocol is further defined so that applications can directly communicate with this module no matter what the operating system is.

Product Highlights

- Ideal user interface for network appliance
- Pre-defined protocol between host and module
- No driver is required; OS independent
- Can display alphanumeric characters and eight user defined icons
- Four operation keys can be customized for different applications
- Facilitate system installation and operation

Product Specifications

- **Display**

  16x2 characters LCD display

- **Function Key**

  Four keys(up, down, enter and escape)

- **Display Icons**

  Eight self-defined icons

- **Host Interface**

  Interface with mother-board or SBC

  - Via serial port
  - Via pre-defined protocol

- **Dimention**

  68(W)x28(L)x30.3(H)mm

- **Environment**

- Operating Temperature: 0℃ ~ 50℃ (32℉ ~ 122℉)
- Storage Temperature: -20℃ ~ 60℃
- Relative Humidity: 5% ～ 95%, non-condensing

Benefits To Our Customers

- **Faster time-to-market**

  Customer can port/develop their software to/on our ready-to-ship solution to minimize time-to-market

- **Better products scalability**

  Customer can select from our wide range of solutions to scale their products

- **Leading edge hardware innovation**

  Customer can always enjoy the most leading edge product from GTech Corporation.

- **More focus on value-added software development**

  Independent Software Venders can focus on their value-added software without worrying the hardware platform development

- **Reduced inventory and manufacturing costs**

  Independent Software Venders can team up with GTech Corporation to provide. solutions to System Integrators or end-users

1. Introduction

The major purpose of this module is to provide an easier man-machine interface for those computing systems in whose applications friendly operation is a “must.” In traditional computing system design, proprietary keypad and LCD display interfaces are implemented and these interfaces are usually different from system to system. The design goals of this interface are:

A. A single interface for both LCD display and keypad is required.
B. This interface should be available in every computing system.
C. The communication implementation should be OS independent.

Our solution is to use “Serial port” as the interface for both LCD display and keypad. A simple protocol is further defined so that applications can directly communicate with this module no matter what the operating system is.

There are only two connectors in this module, as shown in Figure 1: power connector and Serial Port connector.

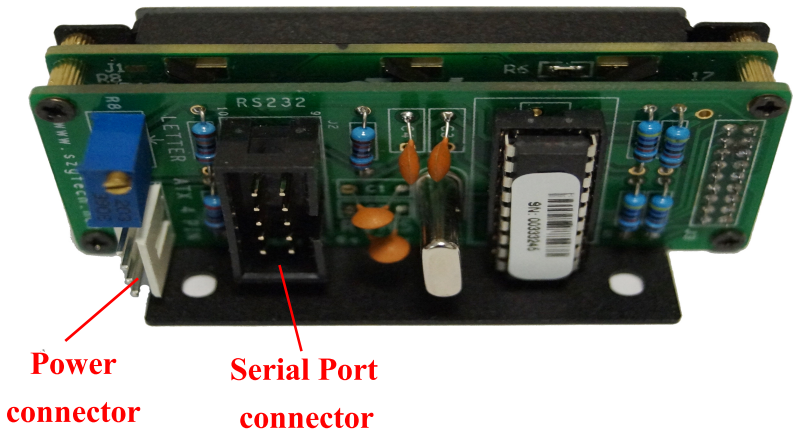


Figure 1.

The power source into this module is 5 volt only. There are only three pins being used in the Serial port interface:

Pin 2:RxD
Pin 3:TxD
Pin 5 : Ground

In another words, this serial port is defined as DCE, therefore, we can use straight-through cable to connect it to Serial port of most computers because they are defined as DTE.

2. Hardware installation

The installation steps are:

A. Connect the power connector to the power connector of this module.
B. Connect the straight-through cable between Serial port of this module and computer.

3. Demo tool

It is a tool for DOS and can be run in Windows environment as well. There are two areas in this Demo/testing tool. The upper area is for editing/sending command/data and lower area is for displaying both out-going and in-coming command/data, as figure 2 shows. The upper area consists of a couple of pages; every page can store up to ten command/data strings. The first byte of every command / data string specify the length of this string. The second byte and those after it are the content to be sent out and are entered in Hexadecimal format. Detailed function of the tool will be shown after pressing "ALT+F1" keys. To exit the demo tool program, "ALT+X" can be pressed.

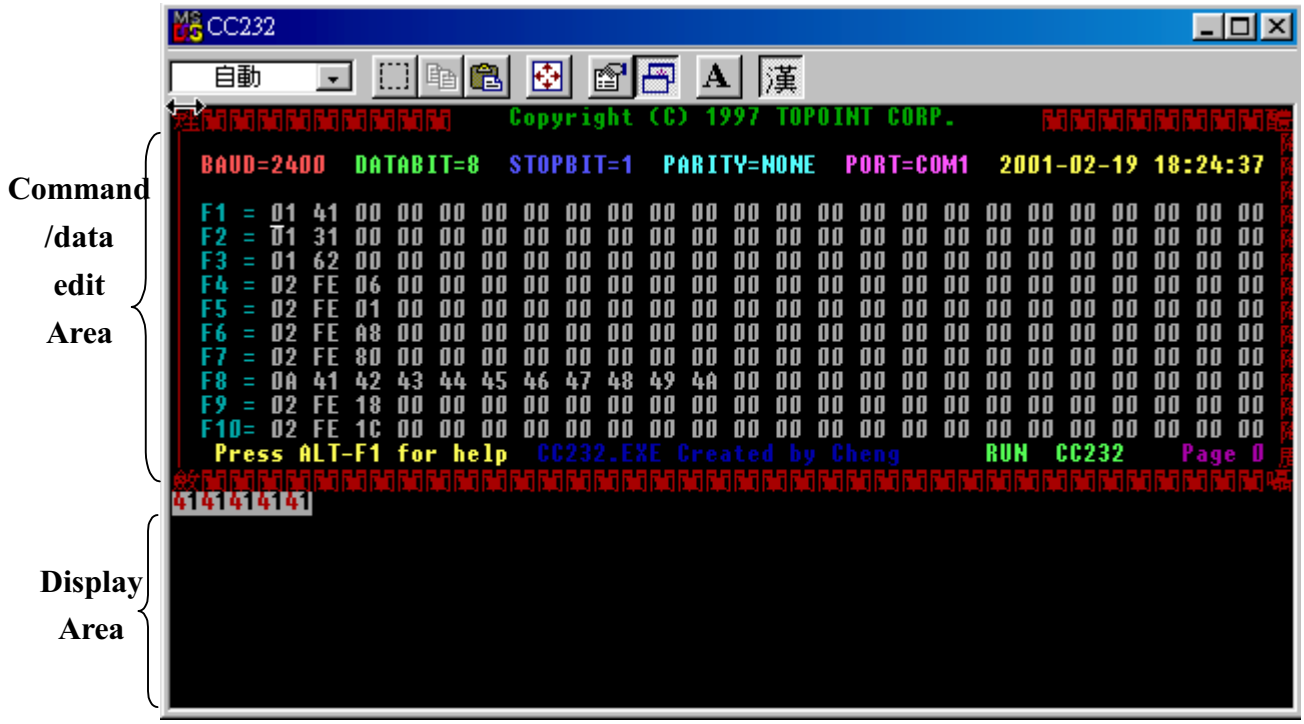


Figure 2

4. Operation Examples

Three pages of examples are stored as default data of this demo tool. The stored contents are as following:

The first page (page 0):

F1: ASCII code of string "ABCD".

F2: Command string to shift the string to right hand side for 3 characters.

F3: Command string to shift the string to right hand side for 3 characters.
F4: Command string to shift the string to right hand side for 3 characters.
F5: Command string to request the key-pad status. A response command will be shown after this request command.
F6: Command string to shift the string to left hand side for 3 characters.
F7: Command to position the cursor to the bottom line, shift the string to right hand side for 2 characters and ASCII code of string “abcd”.
F8: Command string to shift the string to left hand side for 3 characters.
F9: Command string to request the key-pad status.
F10: Command to clear screen.

The first page (page 1):
F1: ASCII code of character “A”.
F2: ASCII code of character “1”.
F3: ASCII code of character “b”.
F4: Command string to request the key-pad status. A response command will be shown after this request command.
F5: Clear screen command.
F6: Command to position the cursor at the beginning of the second column.
F7: Command to position the cursor at the beginning of the first column.
F8: ASCII code of string “ABCDEFGHIJ”.
F9: Command to scroll the displayed string to left hand side for one character.
F10: Command to scroll the displayed string to right hand side for one character.

The second page(page 2):
F1: Command to position the cursor to the upper and left hand side Connor.
F2: Command to hide displayed string.
F3: Command to hide cursor and show hidden string.
F4: Command to blink block cursor.
F5: Command to show underline.
F6: Command to move the cursor to left hand side for one character.
F7: Command to move the cursor to right hand side for one character.
F8: ASCII code of character “B”.
F9: ASCII code of character “C”.
F10: ASCII code of character “3”.

5. Operating procedure

There are two parameters to be changed after entering this tool.

1. Change the operating mode from “monitor” to “CC232” by pressing “ALT + 0”.
2. Change the baud rate from 9600 bps to 2400 bps by pressing “ALT + B” twice.

After these two steps, user is free to select one of the command/data string from the page “0” and “1.” “Page Up” and “Page Down” keys can be used to switch from one page to the other. Once one of “F1” to “F10” key being pressed, the corresponding stored string will be sent immediately which can be verified by checking the out-going string in displaying area. “Page 0” is a demo page to show:

1. Display the string and move it back/forth and up/down.
2. A loop to interrogate the key pressing status.

User can press “Alt” + “F10” so that it will loop between F1 and F10. “Alt” + “F10” can be pressed to stop the looping. “Page 1” and “Page 2” are a list of strings to send out data and major commands.

GTLCM2X16e COMMAND

GTLCM2X16e is an intelligent device which will display those data received from RS232 port and reply key pressing status to polling command from RS232 port. There are command and data from RS232 port. To distinguish between data and commands, the LCD/key-pad Module recognizes a command prefix, 254 (Hex 0FE). The byte following “254” will be processed as a command. For example, to clear the screen, send the command prefix (254) followed by the LCD clear-screen code (1). The valid data range is as following table shows.

Valid data range	Displayed characters
0-7	Customized icon 0-7
48-57 (30-39 Hex)	0-9
65-90 (41-5A Hex)	A-Z
97-122 (61-7A Hex)	a-z

To get the key pressing status, a “read key” command can be issued to this module which will check the key-pressing status and reply accordingly. The following are the command and corresponding Decimal/Hex value:

Functions/commands	Decimal/Hex	comment
Clear screen	1/01	
Home cursor	2/02	
Read key	6/06	See note 1
Blank display (retaining data)	8/08	
Hide cursor & display blanked characters	12/0C	
Turn on (blinking block cursor)	13/0D	
Show underline cursor	14/0E	
Move cursor 1 character left	16/10	
Move cursor 1 character right	20/14	
Scroll 1 character left	24/18	
Scroll 1 character right	28/1C	
Set display address (position the cursor) location	128(Hex080)+ Location	See note 2
Set character-generator address	64(Hex 040)+ address	See note 3

Note 1:

Upon receiving the “read key” command from host computer, the LCD/key-pad module will check the status of every key and reply with status command accordingly. The replied message from LCD/key-pad module consists of a header and a status byte. The header byte is 253 (Hex0FD). The high nibble (with the most significant bit) of the status byte is always “4” and the low nibble (with the least significant bit) of the

status byte is used to indicate key pressing status of the key-pad module. This nibble will be “F” (of four 1s), if there is no key being pressed while the “read key” being received. “0” will be used to indicate key pressing status of corresponding key. There are four keys in this module – upper arrow, down arrow, enter (ENT), and escape (ESC). The relationship between the function key, corresponding status bit, and status byte is as following table.

Function key	Corresponding status bit	status byte
Escape	the fourth bit of lower nibble(the least significant bit) (1110)	4E (H)
Up arrow	the third bit of lower nibble (1101)	4D (H)
Enter	the second bit of lower nibble (1011)	4B (H)
Down arrow	the first bit of lower nibble (0111)	47 (H)

More than one key can be pressed at the same time so that there may be more than one “0”s in the low nibble of status byte. For example, if Up and Down arrow keys are pressed at the same time while “read key” command being received, the replied status will be “Hex045”.

Note 2:

This command can be used to place the cursor at any location. The corresponding address for each character on the screen is as following:

For 16×2 Display Address

Character	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Location (Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

The address of characters at the same row are continuous, so moving cursor commands can be applied to shift the cursor position back and forth. However, the address of characters between upper and lower row are discontinuous. To change cursor position between upper row and lower row, this command will be applied.

Note 3:

This command can be used to create customized icon. The starting address is 64 and every character will take 8 bytes to create a 5 (width) x 7 (height) resolution picture, as shown in following:

CG RAM MAPPING

CG RAM Address							Character Patterns (CG RAM data)								
5	4	3	2	1	0		7	6	5	4	3	2	1	0	
High					Low		High						Low		
0	0	0	0	0	0					0	1	1	0	0	
			0	0	1					1	0	0	1	0	
			0	1	0					0	0	1	0	0	
			0	1	1		*	*	*	0	1	0	0	0	←Character Pattern
			1	0	0					1	1	1	1	0	
			1	0	1					0	0	0	0	0	
			1	1	0					0	0	0	0	0	
			1	1	1					0	0	0	0	0	←Cursor
0	0	1	0	0	0					1	1	1	1	1	
			0	0	1					1	0	0	0	1	
			0	1	0					1	0	1	0	1	
			0	1	1		*	*	*	1	0	1	1	1	←Character Pattern
			1	0	0					1	0	1	0	1	
			1	0	1					1	0	0	0	1	
			1	1	0					1	1	1	1	1	
			1	1	1					0	0	0	0	0	←Cursor
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
1	1	1	0	0	0					1	1	1	1	1	
			0	0	1					1	0	0	0	1	
			0	1	0					1	1	1	0	1	
			0	1	1		*	*	*	1	0	0	0	1	←Character Pattern
			1	0	0					1	0	1	1	1	
			1	0	1					1	0	0	0	1	
			1	1	0					1	1	1	1	1	
			1	1	1					0	0	0	0	0	←Cursor

To show the customized icon, just send the data between “0” to “7” to this module. For example, this module will display the customized icon at location 64 to 71 upon receiving data “0”; it will display the customized icon at location 72 to 79 upon receiving data “1”.

There is a built-in watch dog timer in the module. This module will reset itself and send out “reset packet“ (0FDH, 0EH) thereafter.

The input must be a standard RS232 or inverted TTL signal. The RS232 setup is:

Baud rate: 2400 bps

Parity: None

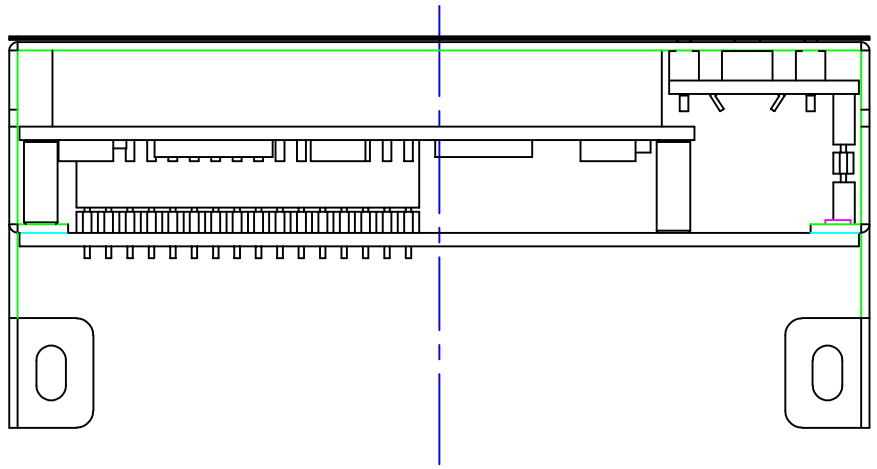
Data bits: 8

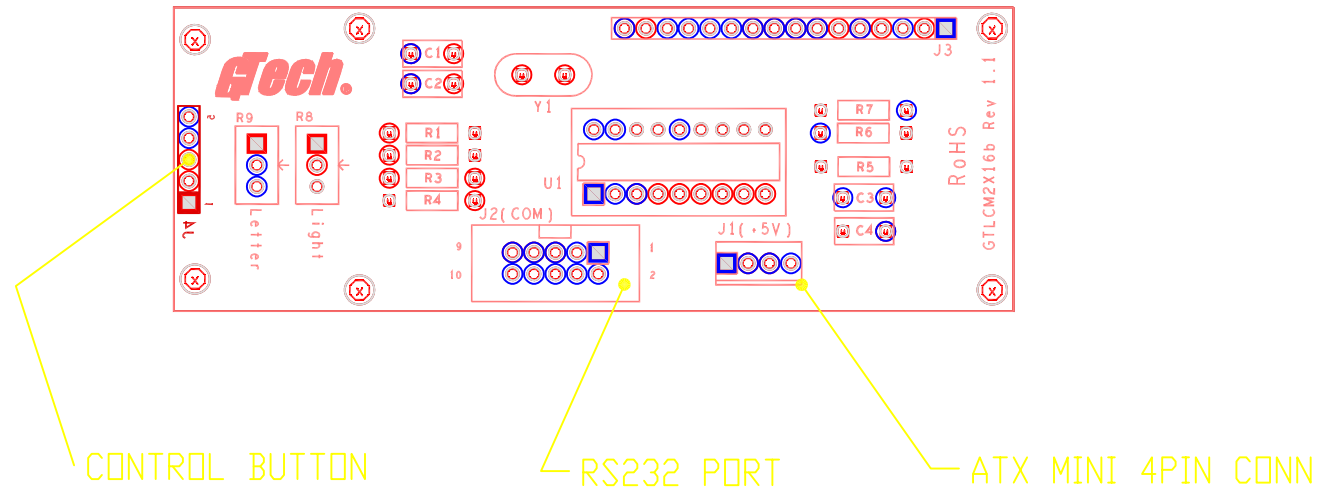
Stop bit: 1

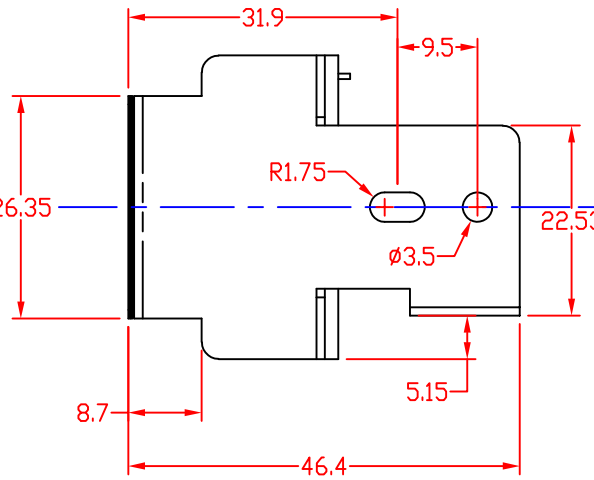
The following are default setup after LCD module be initiated:

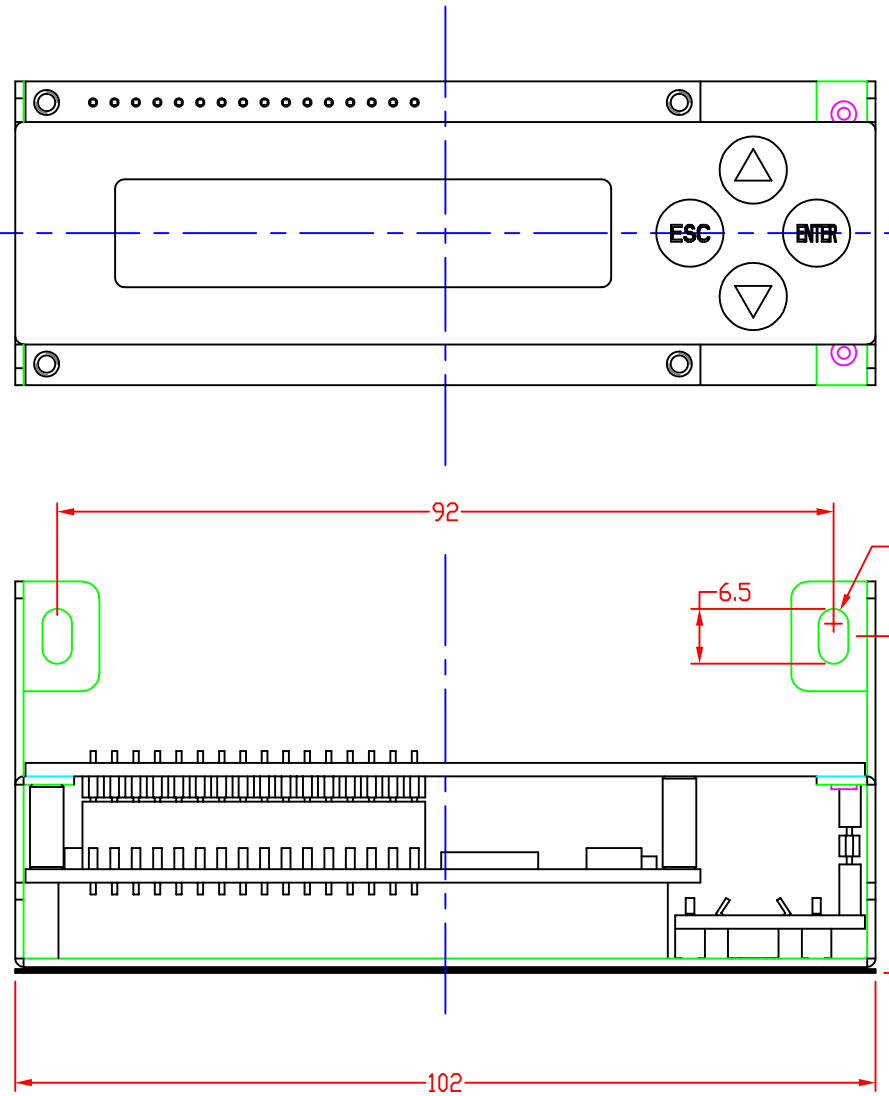
1. 2-line display mode; every character is 5 x 8 dots.
2. Display on; cursor off; cursor blink off.
3. Display will be cleared.
4. Shift right for entry mode.
5. Set address counter to “00”(cursor position to 0)
6. In entry mode.

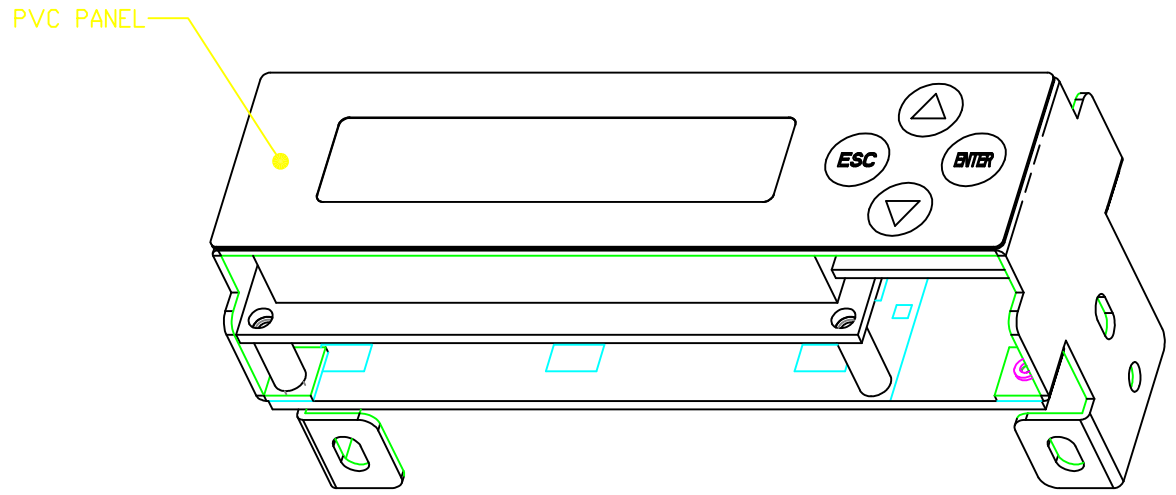
REVISIONS						
REV.	POS.	DESCRIPTION	DATE	DRW	APP	ECN#
A		INITIAL CREATION	2004/02/01	JASON	JUNSON	











PART NO	DESCRIPTION
GTLCM2X16b-RD	RED PVC PANEL
GTLCM2X16b-BL	BLACK PVC PANEL
GTLCM2X16b-BU	BLUE PVC PANEL
GTLCM2X16b-SL	SILVER PVC PANEL

1. UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE MM
2 TOLERANCE ARE AS FOLLOWS:
0 < X < 2 ± 0.06
2 < X < 10 ± 0.08
10 < X < 50 ± 0.12
50 < X < 100 ± 0.16
100 < X < 200 ± 0.20
200 < X < 300 ± 0.30
ANGLES ± 0.5°

MATERIAL:

Part NO.	GTLCM2X16b serial	DESCRIPTION	硬件防火墙串口显示屏

SIGNATURCE		DATE
DRAWN BY	JASON	2004/02/01
CHECKED BY	LILY	2004/02/01
ENGR	GARY	2004/02/01
APPROVED BY	VIVI	2004/02/01
ISSUED BY	JACK	2004/02/01

GTECH CORPORATION
www.sz-gtech.cn gtech@sz-gtech.cn

CAD MODLE:	SCALE: 1:1	REV: A
CAD DWG:	SIZE: A3	SHEET: 1 OF 1

*DO NOT SCALE DRAWING

THIRD ANGLE PROJECTION

■使用举例(仅供参考)

```
#include <reg51.h>
#include <intrins.h>
sbit dc=0xa0;        /*P2.0  LCD 的 RS  21*/
sbit rw=0xa1;        /*P2.1  LCD 的 R/W 22*/
sbit cs=0xa4;        /*P2.4  LCD 的 E   25*/
sfr  lcdbus=0x80;    /*p0LCD 数据 D0=P0.0*/
unsigned int sys10mscounter;
unsigned char syslimitcounter;
char path1[8]={0x00,0x1f,0x00,0x1f,0x00,0x1f,0x00,0x1f};/*自定义符号：横 1*/
char path2[8]={0x1f,0x00,0x1f,0x00,0x1f,0x00,0x1f,0x00};/*自定义符号：横 2*/
char pats1[8]={0x15,0x15,0x15,0x15,0x15,0x15,0x15,0x15};/*自定义符号：竖 1*/
char pats2[8]={0x0a,0x0a,0x0a,0x0a,0x0a,0x0a,0x0a,0x0a};/*自定义符号：竖 2*/
void soft_nop(){}
void soft_10ms()/***********12MHZ 提供 10MS 软件延时***********/
{   register int i;
    for(i=0;i<711;i++);
}
void soft_20ms()/***********12MHZ 提供 20MS 软件延时***********/
{   soft_10ms();
    soft_10ms();
}
void hard_10ms(unsigned int delaytime) /*基于 10MS 的硬件延时*/
{   sys10mscounter=delaytime;
    while(sys10mscounter);
}
unsigned char data lcdcounter;
bit lcdusing1,lcdusing2;
bit lcd_checkbusy()/*检查 LCD 忙*/
{   register lcdstate;
    dc=0;                /*dc=1 为数据,=0 为命令.*/
    rw=1;                /*rw=1 为读,=0 为写.*/
    cs=1;                /*cs=1 选通.*/
    soft_nop();
    lcdstate=lcdbus;
    cs=0;
    return((bit)(lcdstate&0x80));
}
void lcd_wrcmd(unsigned char lcdcmd) /*写 LCD 命令*/
{   lcdusing1=1;
    while(lcd_checkbusy());
    lcdbus=lcdcmd;
    dc=0;                /*dc=1 为数据,=0 为命令.*/
    rw=0;                /*rw=1 为读,=0 为写.*/
    cs=1;                /*cs=1 选通.*/
    soft_nop();
```

```
    cs=0;
    lcdbus=0xff;
    lcdusing1=0;
}
void lcd_moveto(char position) /*移动光标到指定位.0-79*/
{   register cmd=0x80;
    lcdcounter=position;
        if (position > 59)
            position += 0x18;
        else
        {   if (position > 39)position -= 0x14;
            else
            {       if (position > 19)position += 0x2c;
            }
        }
    cmd=cmd|position;
    lcd_wrcmd(cmd);
}
void lcd_wrdata(char lcddata) /*在当前显示位置显示数据*/
{   char i;
    lcdusing2=1;
    while(lcd_checkbusy());
    if(lcdcounter==20){
        lcd_moveto(20);
        while(lcd_checkbusy());
        }
    if(lcdcounter==40){
        lcd_moveto(40);
        while(lcd_checkbusy());
        }
    if(lcdcounter==60){
        lcd_moveto(60);
        while(lcd_checkbusy());
        }
    if(lcdcounter==80){
        lcd_moveto(0);
        while(lcd_checkbusy());
        lcdcounter=0;
        }                       /*为通用而如此*/
    lcdcounter++;
    lcdbus=lcddata;
    dc=1;               /*dc=1 为数据,=0 为命令.*/
    rw=0;               /*rw=1 为读,=0 为写.*/
    cs=1;               /*cs=1 选通.*/
    soft_nop();
    cs=0;
```

```
    lcdbus=0xff;
    lcdusing2=0;
}
void lcd_string(char *strpoint) /*在当前显示位置显示 LCD 字符串*/
{   register i=0;
    while(strpoint[i]!=0){
        lcd_wrdata(strpoint[i]);
        i++;
        }
}
void lcd_init()/*初始化*/
{   lcd_wrcmd(0x38);     /*设置 8 位格式,2 行,5*7*/
    lcd_wrcmd(0x0c);     /*整体显示,关光标,不闪烁*/
    lcd_wrcmd(0x06);     /*设定输入方式,增量不移位*/
    lcd_wrcmd(0x01);     /*清除显示*/
    lcdcounter=0;
}
void lcd_cls()/*清除显示*/
{   lcd_wrcmd(0x01);
    lcdcounter=0;
}
void timer0(void) interrupt 1        /*T0 中断*/
{   TH0=0xd8;            /*12M,10ms*/
    TL0=0xf6;
    TR0=1;
    if(sys10mscounter!=0)sys10mscounter--;       /*定时器 10ms*/
    if(syslimitcounter!=0)syslimitcounter--;/*定时器 10ms*/
}
main()
{
unsigned char j;
IE=0;P0=0xff;P1=0xff;P2=0xff;P3=0xff; /*初始化 T*/
lcd_init();soft_20ms();
TMOD=0x51;
TH0=0xd8;            /*12M,10ms*/
TL0=0xf6;
TR0=1;ET0=1;EA=1;
    while(1)
    {
    /*全黑、横一、横二、竖一、竖二、U、Q、ABCD...，*/
    lcd_init(); /*全黑*/
    for(j=0;j<80;j++){lcd_wrdata(0xff);}
    hard_10ms(50);
    lcd_init(); /*横一，可参考自行设计符号*/
    lcd_wrcmd(0x40);
    for(j=0;j<8;j++)lcd_wrdata(path1[j]);
```

```
    for(j=0;j<100;j++)lcd_wrdata(0);
    hard_10ms(50);
    lcd_init(); /*横二*/
    lcd_wrcmd(0x40);
    for(j=0;j<8;j++)lcd_wrdata(path2[j]);
    for(j=0;j<100;j++)lcd_wrdata(0);
    hard_10ms(50);
    lcd_init(); /*竖一*/
    lcd_wrcmd(0x40);
    for(j=0;j<8;j++)lcd_wrdata(pats1[j]);
    for(j=0;j<100;j++)lcd_wrdata(0);
    hard_10ms(50);
    lcd_init(); /*竖二*/
    lcd_wrcmd(0x40);
    for(j=0;j<8;j++)lcd_wrdata(pats2[j]);
    for(j=0;j<100;j++)lcd_wrdata(0);
    hard_10ms(50);
    lcd_init();
    lcd_string("UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUU");
    hard_10ms(50);
    lcd_init();
    lcd_string("QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
QQQQQ");
    hard_10ms(50);
    lcd_init();
    lcd_string("ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789+-!
#$%&?");
    hard_10ms(50);
    }
}
```

*以上 C51 程序仅为测试使用，仅供参考，请自行断定。不同的驱动 CPU 和不同的语言，参考时序图请自行编写程式